

Министерство науки и высшего образования Российской Федерации
Суперкомпьютерный консорциум университетов России

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ТЕХНОЛОГИИ (ПаВТ'2025)

Короткие статьи и описания плакатов

г. Москва, 8–10 апреля 2025 г.

Челябинск,
Издательский центр ЮУрГУ
2025

УДК 004.75

П 18

Параллельные вычислительные технологии – XIX всероссийская научная конференция с международным участием, ПаВТ'2025, г. Москва, 8–10 апреля 2025 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2025. 344 с.

ISBN 978-5-696-05572-5

<https://doi.org/10.14529/pct2025>

Данный сборник содержит статьи, включенные в программу всероссийской научной конференции с международным участием «Параллельные вычислительные технологии 2025». Конференция проходила 8–10 апреля 2025 года в национальном исследовательском университете «Высшая школа экономики» (г. Москва). Подробную информацию о конференции можно найти в сети Интернет по адресу <https://павт.рф>.

Отпечатано с авторских оригиналов.

Одобрено Советом Высшей школы электроники и компьютерных наук ЮУрГУ

Рецензент:

Вл.В. Воеводин, член-корреспондент РАН

Ответственные за выпуск:

Л.Б. Соколинский, доктор физ.-мат. наук,

Я.А. Краева, кандидат физ.-мат. наук

ПЛАТИНОВЫЕ СПОНСОРЫ

Компания Е-Флопс, Компания АМДтехнологии
Группа компаний РСК

ИНФОРМАЦИОННАЯ ПОДДЕРЖКА

Информационно-аналитический центр Parallel.ru
Газета «Поиск»

НАБЛЮДАТЕЛЬНЫЙ СОВЕТ

Бердышев В.И., академик РАН, научный руководитель ИММ УрО РАН, г. Екатеринбург
Ершов Ю.Л., академик РАН, научный руководитель ИМ СО РАН, г. Новосибирск
Минкин В.И., академик РАН, научный руководитель ЮФУ, г. Ростов-на-Дону
Савин Г.И., академик РАН, научный руководитель ФНЦ НИИСИ РАН, г. Москва
Садовничий В.А., академик РАН, ректор МГУ, г. Москва
Четверушкин Б.Н., академик РАН, научный руководитель ИПМ РАН, г. Москва

ПРОГРАММНЫЙ КОМИТЕТ

Сопредседатели программного комитета:

Воеводин Вл.В., чл.-корр. РАН, НИВЦ МГУ, г. Москва
Донгарра Дж., PhD, Университет Теннесси, США
Соколинский Л.Б., д.ф.-м.н., НИУ ЮУрГУ, г. Челябинск

Зам. председателя программного комитета:

Цымблер М.Л., д.ф.-м.н., НИУ ЮУрГУ, г. Челябинск

Ученый секретарь программного комитета:

Краева Я.А., к.ф.-м.н., НИУ ЮУрГУ, г. Челябинск

Члены программного комитета:

Абламейко С.В., академик НАН РБ, БГУ, г. Минск, Республика Беларусь
Акимова Е.Н., д.ф.-м.н., ИММ УрО РАН, УрФУ, г. Екатеринбург
Андреяк А., PhD, Гейдельбергский университет, Германия
Афанасьев А.П., д.ф.-м.н., МГУ, г. Москва
Балайи П., PhD, Аргоннская национальная лаборатория, США
Болдырев Ю.Я., д.т.н., НИУ СПбГПУ, г. Санкт-Петербург
Виржиковский Р., PhD, Ченстоховский технологический университет, Польша
Газизов Р.К., д.ф.-м.н., ООО «РН-БашНИПИнефть», г. Уфа
Горячев В.Д., д.т.н., ТГТУ, г. Тверь
Губайдуллин И.М., д.ф.-м.н., ИНК УФИЦ РАН, г. Уфа
Ильин В.П., д.ф.-м.н., ИВМиМГ СО РАН, г. Новосибирск
Карретеро Х., PhD, Мадридский университет имени Карлоса III, Испания
Кобаяши Х., PhD, Университет Тохоку, Япония
Кумар С., PhD, Американский университет Армении, Армения
Кюнсел Й., PhD, Гамбургский университет, Германия
Лабарта Х., PhD, Суперкомпьютерный центр Барселоны, Испания
Ластовецкий А.Л., д.ф.-м.н., Университетский колледж Дублина, Ирландия
Лиходед Н.А., д.ф.-м.н., БГУ, г. Минск, Республика Беларусь
Людвиг Т., PhD, Германский вычислительный центр климата, Германия
Маллманн Д., Исследовательский центр г. Юлиха, Германия
Малышкин В.Э., д.т.н., ИВМиМГ СО РАН, г. Новосибирск
Михалевич М., PhD, Вычислительный центр Агентства по научно-технологическим исследованиям, Сингапур
Модорский В.Я., д.т.н., ПНИПУ, г. Пермь
Пан К.С., к.ф.-м.н., Cloudflare, Великобритания
Продан Р., PhD, Альпийско-Адриатический университет Клагенфурта, Австрия
Радченко Г.И., к.ф.-м.н., Silicon Austria Labs, Австрия
Ситоле Х., PhD, Центр НРС, ЮАР
Старченко А.В., д.ф.-м.н., НИУ ТГУ, г. Томск
Стерлинг Т., PhD, Индианский университет, США
Сухинов А.И., чл.-корр. РАН, ДГТУ, г. Ростов-на-Дону
Тауфер М., PhD, Университет Теннесси, США
Турлапов В.Е., д.т.н., НИУ ННГУ, г. Нижний Новгород

Черных А., PhD, Центр научных исследований и высшего образования г. Энсенада, Мексика
Шамакина А.В., к.ф.-м.н., Суперкомпьютерный центр Штутгарта, ФРГ
Шумяцкий П., PhD, Университет г. Бразилиа, Бразилия
Якобовский М.В., чл.-корр. РАН, ИПМ РАН, г. Москва
Ямазаки Е., PhD, Университет г. Пелотас, Бразилия

ОРГАНИЗАЦИОННЫЙ КОМИТЕТ

Председатель организационного комитета:

Одоевская Е.В., проректор НИУ ВШЭ, г. Москва

Ответственный секретарь организационного комитета:

Костенецкий П.С., начальник Отдела суперкомпьютерного моделирования НИУ ВШЭ, г. Москва

Члены организационного комитета:

Антонов А.С., в.н.с. НИВЦ МГУ, г. Москва
Антонова А.П., программист НИВЦ МГУ, г. Москва
Ахметов Ш.К., проректор НИУ ВШЭ, г. Москва
Бондарь Д.П., старший директор по цифровой трансформации НИУ ВШЭ, г. Москва
Валиулин А.А., преподаватель кафедры СП ЮУрГУ, г. Челябинск
Воеводин Вад.В., зав. лабораторией НИВЦ МГУ, г. Москва
Времеш И.М., директор Культурного центра НИУ ВШЭ, г. Москва
Гоглачев А.И., программист НОЦ ИИКТ ЮУрГУ, г. Челябинск
Гохберг Л.М., первый проректор НИУ ВШЭ, г. Москва
Григорьева Е.А., начальник Управления по организации и сопровождению мероприятий НИУ ВШЭ, г. Москва
Клычихина О.В., руководитель проектной группы Дирекции программы развития НИУ ВШЭ, г. Москва
Козырев В.И., ведущий инженер Отдела суперкомпьютерного моделирования НИУ ВШЭ, г. Москва
Краева Я.А., нач. отдела НОЦ ИИКТ ЮУрГУ, г. Челябинск
Курличенко Е.В., директор по информационным технологиям НИУ ВШЭ, г. Москва
Лавров А.С., старший директор НИУ ВШЭ, г. Москва
Левкович Р.В., директор по связям с общественностью НИУ ВШЭ, г. Москва
Масютин А.А., директор Института искусственного интеллекта и цифровых наук ФКН НИУ ВШЭ, г. Москва
Мишенин Р.М., инженер Отдела суперкомпьютерного моделирования НИУ ВШЭ, г. Москва
Нарчук В.Н., нач. отдела планирования и контроля цифровой трансформации Дирекции по обеспечению цифровой трансформации НИУ ВШЭ, г. Москва
Никитенко Д.А., в.н.с. НИВЦ МГУ, г. Москва
Рогачев И.Н., нач. Управления по безопасности и режиму Дирекции по безопасности НИУ ВШЭ, г. Москва
Сидоров И.Ю., программист НИВЦ МГУ, г. Москва
Соколинская И.М., доцент кафедры ПМиП ЮУрГУ, г. Челябинск
Чулкевич Р.А., ведущий инженер Отдела суперкомпьютерного моделирования НИУ ВШЭ, г. Москва
Щур Л.Н., зав. базовой кафедрой «Прикладные информационно-коммуникационные средства и системы» ВЦ РАН и Лабораторией вычислительной физики НИУ ВШЭ, г. Москва
Юртин А.А., программист ЛБДМО ЮУрГУ, г. Челябинск

ЭКСПЕРТЫ ПаВТ

Аксенов А.А., ТЕСИС, г. Москва
Алеева В.Н., ЮУрГУ, г. Челябинск
Андреев А.Е., ВолгГТУ, г. Волгоград
Антонов А.С., НИВЦ МГУ, г. Москва
Бабушкина А.В., ПНИПУ, г. Пермь

Баландин Д.В., ННГУ, г. Нижний Новгород
Баркалов К.А., ННГУ, г. Нижний Новгород
Беляев Е.А., ИТМО, г. С.-Петербург
Болдарев А.С., ИПМ РАН, г. Москва
Будков Ю.А., НИУ ВШЭ, г. Москва
Буша Я., ОИЯИ, г. Дубна
Ваганова Н.А., ИММ УрО РАН, г. Екатеринбург
Васенко А.С., НИУ ВШЭ, г. Москва
Воеводин Вад.В., НИВЦ МГУ, г. Москва
Волохов В.М., ФИЦ ПХФ и МХ РАН, г. Черноголовка
Вшивков В.А., ИВМиМГ СО РАН, г. Новосибирск
Громов В.А., НИУ ВШЭ, г. Москва
Гущин М.И., НИУ ВШЭ, г. Москва
Дацюк О.В., ЮГИНФО ЮФУ, г. Ростов-на-Дону
Деркач Д.А., НИУ ВШЭ, г. Москва
Дордопуло А.И., НИЦ СЭ и НК, г. Таганрог
Духанов А.В., ИТМО, г. С.-Петербург
Завьялов Д.В., ВолгГТУ, г. Волгоград
Иванова Е.В., ЮУрГУ, г. Челябинск
Иванова О.Н., ЮУрГУ, г. Челябинск
Игнатов Д.И., НИУ ВШЭ, г. Москва
Игумнов А.С., ИММ УрО РАН, г. Екатеринбург
Ихсанов Р.Ш., НИУ ВШЭ, г. Москва
Калюлин С.Л., ПНИПУ, г. Пермь
Карпенко А.П., МГТУ, г. Москва
Козинов Е.А., ННГУ, г. Нижний Новгород
Конченков В.И., ВолгГТУ, г. Волгоград
Конюхов В.М., КФУ, г. Казань
Кравец А.Г., ВолгГТУ, г. Волгоград
Куликов И.М., ИВМиМГ СО РАН, г. Новосибирск
Левин И.И., НИИ МВС ЮФУ, г. Таганрог
Легалов А.И., НИУ ВШЭ, г. Москва
Линёв А.В., ННГУ, г. Нижний Новгород
Лукашук С.Ю., УУНиТ, г. Уфа
Марчевский И.К., МГТУ, г. Москва
Медведев М.Г., НИУ ВШЭ, г. Москва
Мееров И.Б., ННГУ, г. Нижний Новгород
Мисилов В.Е., ИММ УрО РАН, г. Екатеринбург
Муратова Г.В., ЮФУ, г. Ростов-на-Дону
Нечаевский А.В., ОИЯИ, г. Дубна
Никитенко Д.А., МГУ, г. Москва
Никитина А.В., ЮФУ, г. Ростов-на-Дону
Петрова Е.Н., ПНИПУ, г. Пермь
Петросян А.Ш., ОИЯИ, г. Дубна
Писарев П.В., ПНИПУ, г. Пермь
Половинкин А.Н., ННГУ, г. Нижний Новгород
Проценко Е.А., ТГПИ (филиал РГЭУ (РИНХ)), г. Таганрог
Ратников Ф.Д., НИУ ВШЭ, г. Москва
Розенберг В.Л., ИММ УрО РАН, г. Екатеринбург
Романов С.Ю., НИВЦ МГУ, г. Москва
Рябченко С.В., САФУ, г. Архангельск
Садовникова Н.П., ВолгГТУ, г. Волгоград
Сальников А.Н., ВМК МГУ, г. Москва

Свешников В.М., ИВМиМГ СО РАН, г. Новосибирск
Семёнов А.А., ИДСТУ СО РАН, г. Иркутск
Сидорякина В.В., ТГПИ (филиал РГЭУ (РИНХ)), г. Таганрог
Сластников С.А., НИУ ВШЭ, г. Москва
Созыкин А.В., МФТИ (НИУ), г. Екатеринбург
Стегайлов В.В., НИУ ВШЭ, г. Москва
Сухорослов О.В., ИППИ РАН, г. Москва
Тимофеев А.В., НИУ ВШЭ, г. Москва
Топорков В.В., НИУ МЭИ, г. Москва
Хведелидзе А.М., ОИЯИ, г. Дубна
Чистяков А.Е., ДГТУ, г. Ростов-на-Дону
Шишени А.В., Яндекс, г. Москва
Штейнберг Б.Я., ИММКН ЮФУ, г. Ростов-на-Дону
Щур В.Л., НИУ ВШЭ, г. Москва
Щур Л.Н., НИУ ВШЭ, г. Москва

Predicting Specific Heat Capacity of Nanofluids Using Machine Learning Methods: Applications in Thermophysics

Y.E. Barablina, V.A. Raudsep

ITMO University

This study presents the development of a predictive model for the specific heat capacity of nanofluids based on the Random Forest Regressor method, optimized using GridSearchCV. The final model parameters demonstrated high prediction accuracy ($MSE = 4.16$, $R^2 = 0.99999$), as confirmed by residual analysis and comparison between actual and predicted values. The model was successfully tested on data for isopropyl alcohol with nanoparticles, showing minimal deviations from the experimental values. Despite limitations associated with the clustered structure of the data, the model exhibits potential for application to other base fluids and nanoparticles, making it a valuable tool for studying the thermophysical properties of nanofluids and developing new materials.

Keywords: nanofluids, heat capacity, RandomForestRegressor, machine learning, thermophysical properties.

1. Introduction

Nanofluids, consisting of nanoparticles suspended in base fluids, have garnered significant attention due to their enhanced thermophysical properties [1]. These fluids exhibit improved thermal conductivity and altered viscosity, making them promising for heat transfer applications [2]. Nanoparticle characteristics, such as concentration, size, and shape, significantly impact nanofluid properties and performance [3]. Metallic oxide nanoparticles like MgO, TiO₂, and ZnO can increase heat transfer efficiency by approximately 30% compared to base fluids [3]. Stability is crucial for nanofluid usability, with various methods employed to enhance it, including surfactant addition and ultrasonic mixing [3]. While increasing nanoparticle concentration can improve heat transfer efficiency, exceeding optimal levels may reduce Brownian motions due to higher viscosity and density [3]. Density, thermal conductivity, viscosity, and heat capacity are key parameters influencing nanofluid performance in heat transfer applications [4].

Recent studies highlight the growing importance of machine learning (ML) techniques in predicting thermophysical properties of nanofluids. These advanced computational methods have proven superior to traditional approaches in handling complex, non-linear relationships between nanofluid parameters [5, 6]. Various ML algorithms, including Artificial Neural Networks (ANNs), Support Vector Regression (SVR), and genetic algorithms, have been employed to accurately predict properties such as thermal conductivity, viscosity, and specific heat capacity [6–8]. The input parameters typically include temperature, concentration, nanoparticle size, and base fluid type [7, 6]. Researchers have found that extra trees and decision trees often provide the best results for estimating thermal conductivity and viscosity, respectively [7]. These ML methods offer efficient, rapid, and practical alternatives to experimental approaches, making them valuable tools for researchers and engineers in the field of nanofluids [8–11].

Machine learning techniques have emerged as powerful tools for predicting and optimizing nanofluid thermophysical properties, offering superior accuracy compared to traditional methods [12, 13]. Various approaches, including Artificial Neural Networks (ANNs), Support Vector Regression (SVR), and genetic algorithms, have demonstrated effectiveness in capturing the complex dynamics of nanofluids [12, 14]. These AI-based models excel at processing large datasets and identifying intricate patterns, making them particularly suitable for predicting properties such as thermal conductivity, viscosity, and specific heat capacity [14, 15]. The integration of machine learning with empirical data has shown promise in enhancing the efficiency and accuracy of predictions, outperforming classical approaches [12, 14]. This advancement in predictive capabilities facilitates the optimization of nanofluid

compositions for specific technical applications, marking a significant step forward in thermal engineering and management [12, 15].

Recent research highlights the potential of machine learning in nanofluid design and analysis. A novel machine learning framework coupled with molecular dynamics methods has been proposed to model multi-component mixing nanofluidic systems, significantly reducing computational costs and improving prediction accuracy for surfactant adsorption properties and heat transfer performance [16]. Machine learning techniques have been shown to efficiently predict thermophysical properties of nanofluids, offering advantages over experimental methods [13]. Various machine learning algorithms, including artificial neural networks, genetic algorithms, and ensemble techniques like Boosted regression and XGBoost, have been applied to nanofluid-based heat transfer studies in renewable energy systems [17]. In a specific study on ethylene glycol- and glycerol-based SiO₂ nanofluids, decision tree-based models performed well in predicting thermal performance, with random forest and extreme gradient boosting models also showing high accuracy [18]. These advancements demonstrate the growing importance of machine learning in nanofluid research and applications.

The aim of this study is to develop and optimize a predictive model for estimating the specific heat capacity of nanofluids using the Random Forest Regressor method. The relevance of this work is driven by the widespread application of nanofluids in various fields, including heat exchange systems, energy technologies, and cooling systems, where accurate knowledge of their thermophysical properties plays a critical role. However, the experimental determination of specific heat capacity is often associated with significant time and resource costs, making predictive models a promising tool for addressing this challenge.

The research addresses the following objectives: analyzing data on the specific heat capacity of various nanofluids, developing a predictive model based on the Random Forest Regressor method, optimizing its hyperparameters using the GridSearchCV method, and evaluating the model's accuracy using standard metrics (MSE, R²). Additionally, the model is tested on new data, including experimental values for isopropyl alcohol with Al₂O₃ nanoparticles, to assess its applicability and generalizability. Particular attention is given to analyzing the model's limitations related to data clustering and its potential adaptation for other base fluids and types of nanoparticles.

The findings of this study may contribute to accelerating the development of new nanofluids with desired thermophysical properties and enhancing the understanding of heat transfer mechanisms in such systems.

2. Methods

In the study, data on nanofluids were used, which are available in the dataset on the Kaggle platform [19]. This dataset contains information on various properties of nanofluids, including parameters such as nanoparticle size, nanoparticle composition, concentration, temperature, type of base fluid, and specific heat capacity. The objective of this research is to predict the specific heat capacity of nanofluids based on these properties.

At the first stage, data preprocessing was performed. The first two columns, containing categorical data unrelated to numerical values, were removed from the dataset, and all missing commas were replaced with periods. This resulted in a cleaned sample suitable for further analysis. Outliers were also removed. To analyze the distribution of the data, plots were created to display the distribution of various characteristics of the nanofluids (Fig. 1).

At the first stage, data preprocessing was performed. The first two columns, containing categorical data unrelated to numerical values, were removed from the dataset, and all missing commas were replaced with periods. This resulted in a cleaned sample suitable for further analysis. Outliers were also removed. To analyze the distribution of the data, plots were created to display the distribution of various characteristics of the nanofluids (Fig. 1).

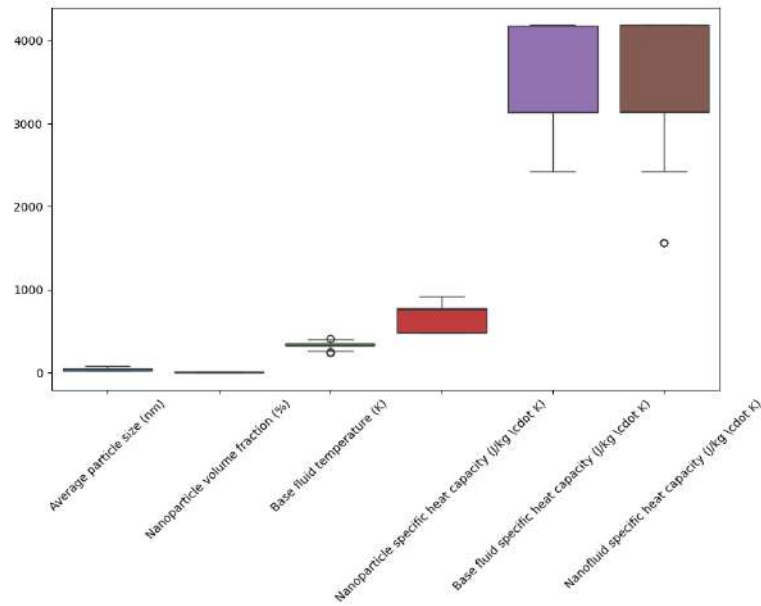


Fig. 1. Distribution of variables in the processed dataset

To gain a deeper understanding of the relationships between the different characteristics of nanofluids, a correlation analysis between the features was conducted. To assess the degree of association between the variables, a Pearson correlation analysis was performed, which quantitatively evaluates the linear relationship between numerical features. As a result, correlation matrices were obtained, clearly illustrating the degree of interconnection between all the variables. For a more visual representation of the correlations, a heatmap was created, displaying the correlation coefficients between all features in the dataset. The most highly correlated features are the heat capacity of the nanofluid and the heat capacity of the base fluid (Fig. 2).

To train the model and assess its performance, the data was split into two parts: a training set and a test set. The standard `train_test_split` function from the `sklearn` library was used to randomly divide the data, with 80% of the data allocated for training the model and the remaining 20% for testing. The target variable selected was the nanofluid specific heat capacity, which is predicted based on other features. To ensure that the variables were evenly distributed across the samples, distribution plots were created (Fig. 3).

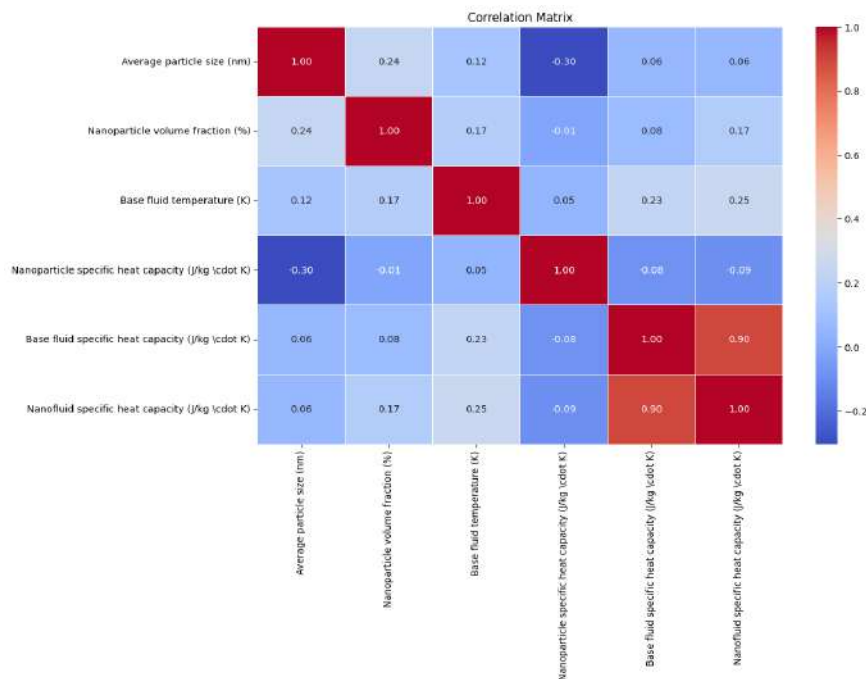


Fig. 2. Correlation matrix of variables in the studied dataset

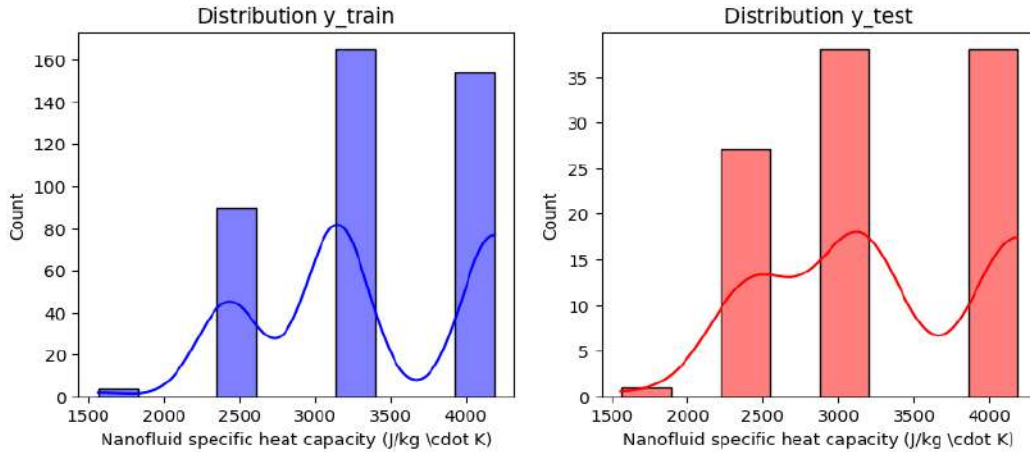


Fig. 3. Distribution plots for the test and training datasets

For the correct functioning of the model, all numerical features were standardized using the StandardScaler method. This step was necessary because different features may have varying scales (for example, temperature and density), which could impact the model's performance.

To predict the specific heat capacity, the RandomForestRegressor method was chosen. This algorithm is well-suited for regression tasks as it can model complex relationships between features and the target variable without making assumptions about the data's distribution. The Random Forest model constructs an ensemble of decision trees and uses the average of their predictions to produce a more accurate result.

To improve model performance, hyperparameter tuning was performed using GridSearchCV. This technique helps find the optimal parameters for the model, such as the number of trees in the forest (`n_estimators`), the maximum depth of the trees (`max_depth`), the minimum number of samples required to split a node (`min_samples_split`), and others. This enhances the model's accuracy and prevents overfitting.

During the training phase, cross-validation with 3 folds was employed for hyperparameter optimization, running the model 3600 times for 1200 different hyperparameter combinations. The best hyperparameters obtained after optimization were: `max_depth=None`, `max_features=None`, `min_samples_leaf=1`, `min_samples_split=2`, and `n_estimators=42`.

After optimizing the hyperparameters, the Random Forest model showed excellent results on the test set, with a mean squared error (MSE) of 4.16 and a coefficient of determination (R^2) of 0.99999, indicating extremely high prediction accuracy and the closeness of the predicted values to the actual data.

Mean Squared Error (MSE):

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2,$$

where y_i — are the actual values, $\hat{y}_i$ — are the predicted values, and n is the number of observations.

Coefficient of Determination (R^2):

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2},$$

where $\bar{y}$ — is the mean of the actual values.

These results indicate that the selected model with optimized parameters is capable of predicting the specific heat capacity of nanofluids with high accuracy based on the input features.

To further improve the results and assess the possibility of applying other methods, a Gradient Boosting model based on the XGBoost algorithm was used. XGBoost is one of the most efficient methods for regression and classification tasks. Gradient Boosting helps reduce overfitting and improve prediction accuracy by sequentially training weak models, where each model corrects the errors of the previous one. For model tuning, the following parameter set was chosen: number of trees (`n_estimators=1000`), learning rate (`learning_rate=0.01`), and a fixed random state (`random_state=80`).

3. Results

The final parameters of the model's learning quality are as follows: the mean squared error (MSE) was 0.0138, the mean absolute error (MAE) was 0.0855, and the coefficient of determination (R^2) reached a value of 0.999999734. These results confirm the high accuracy of the heat capacity prediction for nanofluids, indicating the model's ability to precisely predict the system's behavior, even with minimal errors.

To evaluate the prediction quality, several graphs were constructed.

1. Comparison of actual and predicted values (Fig. 4): The first graph presents the real and predicted values of heat capacity. The line of perfect agreement, which runs diagonally, represents the theoretical accuracy of predictions. The graph demonstrates that the model provides predictions that are close to the actual values, which is further supported by the model's high accuracy.
2. Residual distribution in Fig. 5: The second graph illustrates the distribution of residuals (the difference between actual and predicted values). The distribution is close to normal, indicating good model quality and the absence of systematic errors. The small magnitude of the standard deviation of the residuals further confirms this.

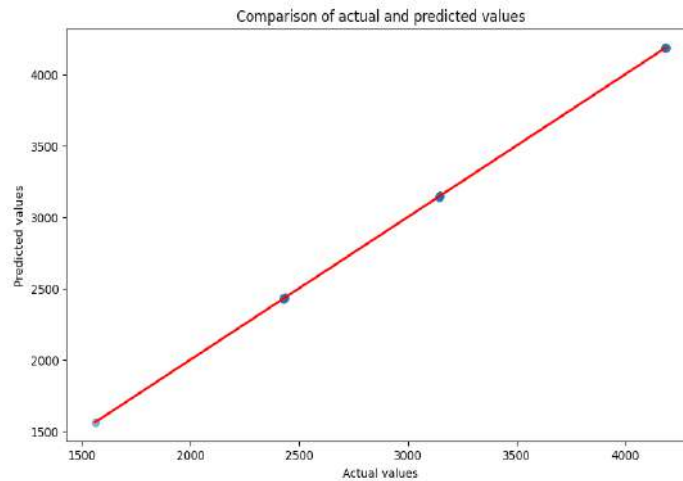


Fig. 4. Visualization of Actual vs Predicted Values

3. Residuals vs Predicted Values (Fig. 6): The third graph displays the distribution of residuals relative to the predicted values. The line positioned at zero helps to visually demonstrate that the residuals barely deviate from zero, further confirming the high accuracy of the model.

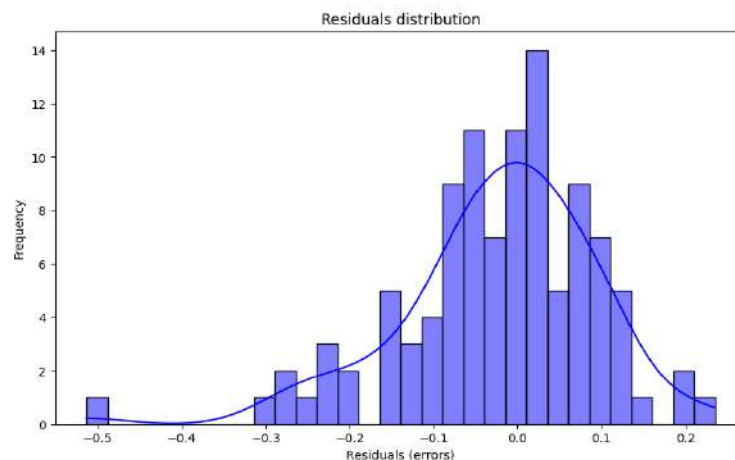


Fig. 5. Visualization of Residuals

4. Residuals vs Predicted Values (Fig. 6): The third graph displays the distribution of residuals relative to the predicted values. The line positioned at zero helps to visually demonstrate that the residuals barely deviate from zero, further confirming the high accuracy of the model.

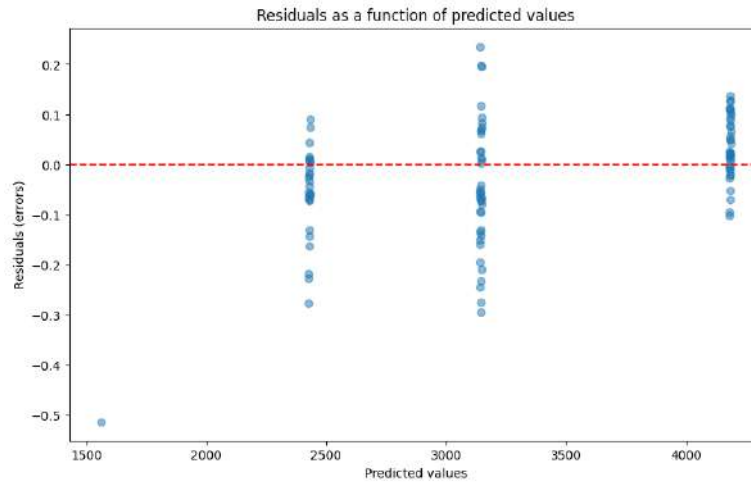


Fig. 6. Residuals vs Predicted Values Plot

Mean and Standard Deviation of Residuals: The mean value of the residuals was close to zero, and the standard deviation was very low, further confirming the high accuracy of the model's performance.

Additionally, the model was tested on random experimental data obtained from the study “Effect of Al_2O_3 Nanoparticle Impurities on the Heat Capacity of Isopropyl Alcohol” [20]. The experimental data included various concentrations of nanoparticles in isopropyl alcohol, allowing for further validation of the model's applicability in real-world chemical systems. The following sample characteristics were used during the validation:

- average particle size: 53 nm;
- volume fraction of nanoparticles: 2%;
- base fluid temperature: 293 K;
- specific heat capacity of nanoparticles: 773 J/kg·K;
- specific heat capacity of base fluid: 2400 J/kg·K.

Using these data, the model predicted the specific heat capacity of the nanofluid to be 2428.32 J/kg·K. This value was found to be close to the actual experimental data, confirming the high accuracy of the model's predictions.

The results demonstrated that the model effectively predicts the heat capacity of isopropyl alcohol with Al_2O_3 nanoparticles, confirming its universality and precision.

4. Conclusion

The proposed solution for predicting the specific heat capacity of nanofluids, despite its high accuracy, still demonstrates some deviation from ideal results. This can be attributed to the fact that the initial dataset exhibits a clustered structure, which may affect the model's ability to generalize to new, previously unseen data. Nevertheless, the model achieves excellent performance on both test and experimental datasets, confirming its effectiveness in predicting the thermophysical properties of nanofluids.

It is worth noting that the developed model for predicting the specific heat capacity of nanofluids, based on the current dataset, can be successfully applied to other fluids and nanoparticles not included in the original dataset. This opens up opportunities for its broader application in various fields, including the study of new materials and systems where accurate predictions of thermophysical properties are required.

Future work will focus on expanding the current approach by increasing the size of the dataset and incorporating a wider range of input parameters. This is expected to enhance the model's quality and improve its generalization capabilities. Additionally, further research will involve comparing different

predictive models, including neural networks, which may potentially yield better results, particularly when working with larger and more diverse datasets.

Thus, further improvement of the model and its adaptation to new types of data present promising prospects for developing even more accurate and versatile tools for predicting the thermophysical properties of nanofluids.

References

1. Kalsi S., Kumar S., Kumar A. et al. Thermophysical properties of nanofluids and their potential applications in heat transfer enhancement: A review // *Arabian Journal of Chemistry*. 2023. Vol. 16, no. 11. P. 105272. DOI: 10.1016/j.arabjc.2023.105272.
2. Ali A.R.I., Salam B. A review on nanofluid: preparation, stability, thermophysical properties, heat transfer characteristics and application // *SN Applied Sciences*. 2020. Vol. 2. P. 1636. DOI: 10.1007/s42452-020-03427-1.
3. Rahman M.A., Hasnain S.M.M., Pandey S. et al. Review on Nanofluids: Preparation, Properties, Stability, and Thermal Performance Augmentation in Heat Transfer Applications // *ACS Omega*. 2024. Vol. 9, no. 30. P. 328–339. DOI: 10.1021/acsomega.4c03279.
4. Arslan R., Özdemir V.A., Akyol E. et al. Thermophysical Properties of Nanofluids // *Current Nanoscience*. 2021. Vol. 17, no. 5. DOI: 10.2174/1573413716999201228130519.
5. Basu A., Saha A., Banerjee S. et al. A Review of Artificial Intelligence Methods in Predicting Thermophysical Properties of Nanofluids for Heat Transfer Applications // *Energies*. 2024. P. 56–89.
6. Ganga S., Uddin Z., Asthana R. et al. Modelling of Viscosity and Thermal Conductivity of Water-Based Nanofluids using Machine-Learning Techniques // *International Journal of Mathematical, Engineering and Management Sciences*. 2023. P. 24–27.
7. Wang H.B., Chen X. A Comprehensive Review of Predicting the Thermophysical Properties of Nanofluids Using Machine Learning Methods // *Industrial Engineering & Chemistry Research*. 2022. Vol. 61, no. 40. P. 14711–14730. DOI: 10.1021/acs.iecr.2c02059.
8. Apmann K., Fulmer R., Elias Soto A.A., Vafaei S. Thermal Conductivity and Viscosity: Review and Optimization of Effects of Nanoparticles // *Materials*. 2021. Vol. 14, no. 5. P. 78. DOI: 10.3390/ma14051291.
9. Urmi W.T., Rahman M.M., Kadirgama K. et al. A Comprehensive Review on Thermal Conductivity and Viscosity of Nanofluids // *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences*. 2022. P. 5.
10. Arslan R., Özdemir V.A., Akyol E. et al. Thermophysical Properties of Nanofluids // *Current Nanoscience*. 2020. P. 16–18.
11. Ali A.R.I., Salam B. A review on nanofluid: preparation, stability, thermophysical properties, heat transfer characteristics and application // *SN Applied Sciences*. 2020. Vol. 2. P. 78–91. DOI: 10.1007/s42452-020-03427-1.
12. Saha A., Basu A., Banerjee S. Enhancing thermal management systems: a machine learning and metaheuristic approach for predicting thermophysical properties of nanofluids // *Engineering Research Express*. 2024. P. 89–97.
13. Wang H.B., Chen X.Y. A Comprehensive Review of Predicting the Thermophysical Properties of Nanofluids Using Machine Learning Methods // *Industrial Engineering & Chemistry Research*. 2022. Vol. 61, no. 40. P. 101–107. DOI: 10.1021/acs.iecr.2c02059.
14. Maleki A., Haghighi A., Mahariq I. Machine learning-based approaches for modeling thermophysical properties of hybrid nanofluids: A comprehensive review // *Journal of Molecular Liquids*. 2021. Vol. 322. P. 114843. DOI: 10.1016/j.molliq.2020.114843.

15. Saha A., Basu A., Banerjee S. Enhancing thermal management systems: a machine learning and metaheuristic approach for predicting thermophysical properties of nanofluids // *Engineering Research Express*. 2024. Vol. 6, no. 4. P. 7–23. DOI: 10.1088/2631-8695/ad8536.
16. Li G., Sun H., Han D. et al. Applying machine learning to reveal the microscopic heat transfer mechanism of nanofluids as coolants // *Thermochimica Acta*. 2024. P. 1–9. DOI: 10.1016/j.tca.2024.179788.
17. Sharma P., Said Z., Kumar A. et al. Recent Advances in Machine Learning Research for Nanofluid-Based Heat Transfer in Renewable Energy System // *Energy & Fuels*. 2022. Vol. 36, no. 13. P. 6626–6658. DOI: 10.1021/acs.energyfuels.2c01006.
18. Akilu S., Sharma K.V., Baheta A.T. et al. Machine learning analysis of thermophysical and thermohydraulic properties in ethylene glycol- and glycerol-based SiO₂ nanofluids // *Scientific Reports*. 2024. Vol. 14. Article 14829. P. 21–27. DOI: 10.1038/s41598-024-65411-8.
19. NanoFluid Dynamics. Revolutionizing Thermal Management: Advanced Insights from Nanofluid Experiment. URL: <https://www.kaggle.com/datasets/kumaraugustya/nanoparticle-basefluid> (дата обращения: 05.01.2025).
20. Лозовский Т., Шимчук Н., Мотовой И., Железный В. Влияние примесей наночастиц Al₂O₃ на теплоемкость изопропилового спирта // *Холодильна техніка та технологія*. 2016. № 52. С. 52–79.

Application of Physics-Informed Neural Networks for Solving the Inverse Advection-Diffusion Problem to Localize Pollution Sources

Ivan Chuprov^(✉)[0000–0002–1091–4442], Denis Derkach^[0000–0001–5871–0628],
Dmitry Efremenko, and Aleksei Kychkin

HSE University, Moscow, Russia
ichuprov@hse.ru

Abstract. This paper investigates the application of Physics-Informed Neural Networks (PINNs) for solving the inverse advection-diffusion problem to localize pollution sources. The study focuses on optimizing neural network architectures to accurately model pollutant dispersion dynamics under diverse conditions, including scenarios with weak and strong winds and multiple pollution sources. Various PINN configurations are evaluated, showing the strong dependence of solution accuracy on hyperparameter selection. Recommendations for efficient PINN configurations are provided based on these comparisons. The approach is tested across multiple scenarios and validated using real-world data that accounts for atmospheric variability. The results demonstrate that the proposed methodology achieves high accuracy in source localization, showcasing the stability and potential of PINNs for addressing environmental monitoring and pollution management challenges under complex weather conditions.

Keywords: Physics-Informed Neural Networks · Inverse problem · Advection-diffusion process

1 Introduction

Weather forecasting and solving related challenges represent some of the most complex areas in meteorology. Numerical models, such as global circulation models [1], require significant computational resources to accurately describe atmospheric processes. However, these approaches often struggle with predicting local weather parameters, particularly in regions with limited observational data or sparse meteorological station networks. This limitation is especially critical when describing the migration of pollution in the atmosphere, where accurate modeling is essential for understanding pollutant transport and devising effective air quality management strategies [2].

Currently, numerous technology companies are leading the development of local and regional air quality monitoring systems. In industrial emission monitoring, well-established solutions exist that enable automatic control directly at

the emission source, such as installations on smokestacks. However, several challenges remain unresolved, including predictive modeling and continuous analysis of pollutant dispersion, identification of emission sources and causes of maximum allowable concentration exceedances, as well as proactive monitoring and reduction of harmful industrial emissions into the atmosphere.

In recent years, machine learning (ML) methods have emerged as promising alternatives to numerical weather prediction [3], particularly in scenarios where data-driven approaches can accelerate computationally expensive components. For instance, physics-incorporated deep-learning frameworks have been developed to parameterize atmospheric radiative transfer, enforcing energy conservation laws while improving prediction accuracy [4]. However, a major limitation of ML models is their dependency on large volumes of high-quality training data. This challenge is particularly pronounced in regions with sparse meteorological station networks or when predicting rare events.

A relatively new and innovative approach, Physics-Informed Neural Networks (PINNs), addresses this limitation by incorporating physical laws directly into the model training process [5]. PINNs demand substantial computational resources, but leveraging parallel computing on GPUs can significantly enhance their training efficiency and scalability. Unlike traditional neural networks, PINNs embed fundamental principles, such as mass, energy, and momentum conservation, into their loss functions. This allows them to use mathematical representations of physical laws, making them more reliable and accurate.

PINNs have already been successfully applied in fields such as optics, fluid dynamics, and material science, where they have proven effective in solving both forward and inverse problems. For example, they have been used to address the inverse Navier-Stokes problems [6], groundwater modeling [7], and advection-diffusion equations [8], although the latter focused on water bodies rather than atmospheric processes. The ability of PINNs to generalize across parameter ranges has been particularly demonstrated in nonlinear Schrödinger equations [9], showing their potential for efficient modeling and prediction across a broad spectrum of conditions.

Despite these achievements, the application of PINNs to inverse problems in atmospheric physics, such as pollution source localization, remains under-explored. While recent studies have attempted to estimate pollutant sources in rivers using PINNs [8], their performance in turbulent atmospheric environments — where transport dynamics are significantly more complex — requires further investigation. The integration of sparse ground-based measurements with physical constraints offers a unique opportunity to address this gap.

The localization of the pollution source and the modeling of the dispersion of pollutants are critical to the development of strategies to improve air quality and mitigate the environmental impact of anthropogenic activities. PINNs, by integrating data-driven learning with physical modeling, offer a unique opportunity to tackle this challenge more effectively, even in scenarios with limited observational data.

This study focuses on using PINNs to address challenges related to atmospheric pollution, specifically in modeling pollutant dispersion and identifying pollution sources. By exploring the capabilities of PINNs in this context, we aim to develop a computational framework that accurately retrieves the coordinates of pollution sources using ground-based measurements, providing a robust tool for air quality management and environmental monitoring. Section 2 presents the theoretical foundations, introducing the governing equation and PINN methodology, with particular focus on implementation aspects critical for computational efficiency. Section 3 addresses the practical implementation, examining neural network architecture choices that ensure training stability and convergence speed. Section 4 provides a comparative analysis of the method's performance in solving both forward and inverse problems using test cases and real observational data, demonstrating the practical effectiveness of the approach.

2 Theory

2.1 Problem formulation

The modeling of atmospheric pollution involves understanding the dispersion of pollutants and identifying their sources. This process is governed by the advection-diffusion equation, which describes the transport of pollutant concentration c in the atmosphere:

$$\frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c - \nabla \cdot (k \nabla c) = s, \quad (1)$$

where $\mathbf{u}$ is the wind velocity vector, k is the diffusion coefficient, while s is the pollutant source term.

Assume that the pollutant is released instantaneously at $t = 0$ from the source location (x_0, y_0) . The initial concentration field is zero everywhere except at the source, where it follows a Gaussian distribution:

$$c(\mathbf{x}, 0) = c_0 \exp \left(-\frac{(x - x_0)^2 + (y - y_0)^2}{2r^2} \right), \quad \mathbf{x} \in \Omega, \quad (2)$$

where c_0 is the peak concentration and r controls the initial spread radius.

Homogeneous Dirichlet boundary conditions are imposed:

$$c(\mathbf{x}, t) = 0, \quad \mathbf{x} \in \partial\Omega, \quad t \geq 0. \quad (3)$$

For computational simplicity, the wind velocity field $\mathbf{u}$ is treated as a, obtained either through observations or as the solution to the Navier-Stokes equations. The solution of the Navier-Stokes equations using PINNs has been explored in [10] and is beyond the scope of this study. By treating $\mathbf{u}$ as known, we decouple the problem and focus exclusively on solving the advection-diffusion equation to model pollutant dispersion.

The forward problem consists of solving the advection-diffusion equation given the pollutant source term s , the wind velocity field $\mathbf{u}$, and other parameters

such as k . The goal is to determine the pollutant concentration c as a function of space and time. Mathematically, the forward problem can be expressed as:

Given $\mathbf{u}, k$, and s , solve for $c(x, y, t)$.

The inverse problem involves determining the unknown source term s , which describes the location and intensity of pollutant emissions, using discrete ground-based measurements of the pollutant concentration c . The objective is to identify s such that the advection-diffusion equation is satisfied for the observed data. Mathematically, the inverse problem can be formulated as:

Given $c(x, y, t)$ (measured data), solve for $s(x, y)$.

Nondimensionalization of equations is a common procedure to convert them into dimensionless variables. This approach simplifies the analysis of the problem and ensures that all terms in the equation are of the same order of magnitude. As shown in [9,11], nondimensionalization can have a positive impact on the convergence of PINNs. By ensuring that the terms in the equation are balanced, the sensitivity of the loss function becomes equal for all PDE terms, leading to more stable and efficient training. When scaling the advection-diffusion equation, we use characteristic values for concentration C , length L , and velocity U . The dimensionless variables are defined as follows:

$$c^* = \frac{c}{C}, \quad \mathbf{u}^* = \frac{\mathbf{u}}{U}, \quad t^* = \frac{tU}{L}. \quad (4)$$

Substituting these dimensionless variables into Eq. (1) yields:

$$\frac{\partial c^*}{\partial t^*} + \mathbf{u}^* \cdot \nabla c^* - \frac{1}{\text{Pe}} \nabla^2 c^* = s, \quad (5)$$

where Pe is the Peclet number, given by

$$\text{Pe} = \frac{LU}{k}. \quad (6)$$

2.2 PINN methodology

Here we briefly summarize the principles of PINN. PINN is a machine learning approach that combines data-driven learning with physical laws expressed as PDEs. Unlike traditional neural networks, which rely solely on data for training, PINNs embed governing physical equations directly into their loss function. This ensures that the solutions not only fit the observed data but also satisfy fundamental physical principles. In this study, we apply the PINN framework to solve the advection-diffusion equation (1). In this case, PINN takes spatial coordinates (x, y) and time t as input and predicts the pollutant concentration $c(x, y, t)$ as output. In this way, PINN approximates the solution to the advection-diffusion equation:

$$c_{\text{PINN}}(x, y, t; \theta),$$

where θ represents the trainable parameters (weights and biases) of the neural network.

The loss function in a PINN framework is designed to enforce both data consistency and physical law adherence. It consists of three primary components listed below.

1. PDE residual loss ensures that the predicted solution satisfies the advection-diffusion equation:

$$\mathcal{E}_{\text{PDE}} = \frac{1}{N_{\text{collocation}}} \sum_{i=1}^{N_{\text{collocation}}} \left| \frac{\partial c_{\text{PINN}}}{\partial t} + \mathbf{u} \cdot \nabla c_{\text{PINN}} - \nabla \cdot (k \nabla c_{\text{PINN}}) - s_{\text{PINN}} \right|^2. \quad (7)$$

Here, $N_{\text{collocation}}$ represents the number of collocation points in the domain.

2. Data loss minimizes the discrepancy between the predicted and observed pollutant concentrations at ground-based measurement points:

$$\mathcal{E}_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} |c_{\text{PINN}}(x_i, y_i, t_i; \theta) - c_{\text{obs}}(x_i, y_i, t_i)|^2, \quad (8)$$

where c_{obs} denotes the observed concentration values and N_{data} is the number of measurement points.

3. Loss of the boundary and initial condition ensures that the solution satisfies the boundary and initial conditions of the problem:

$$\mathcal{E}_{\text{BC}} = \frac{1}{N_{\text{boundary}}} \sum_{i=1}^{N_{\text{boundary}}} |c_{\text{PINN}}(x_i, y_i, t=0) - c_{\text{boundary}}(x_i, y_i)|^2. \quad (9)$$

In the PINN method, initial conditions are formally treated as a special case of boundary conditions since the temporal coordinate t is included among the spatial variables. This allows both types of conditions to be accounted for uniformly in the loss function. The total loss function minimized during the training is a weighted combination of these components:

$$\mathcal{E}_{\text{total}} = \lambda_{\text{PDE}} \mathcal{E}_{\text{PDE}} + \lambda_{\text{data}} \mathcal{E}_{\text{data}} + \lambda_{\text{BC}} \mathcal{E}_{\text{BC}}, \quad (10)$$

where λ_{PDE} , λ_{data} , and λ_{BC} are weights balancing the contributions of each term. Depending on the type and formulation of the problem, certain terms in the equation may not be used. For example, in the forward problem, the term corresponding to data losses $\mathcal{E}_{\text{data}}$ may be omitted, while in the inverse problem, the term related to boundary conditions $\mathcal{E}_{\text{BC}}$ may be excluded. The training process involves minimizing the total loss function $\mathcal{E}_{\text{total}}$ using a gradient-based optimizer such as Adam or L-BFGS. During training the neural network predicts c_{PINN} for collocation points and measurement points. The loss function is evaluated and backpropagated to update the network parameters θ . The schematic of the PINN architecture is illustrated in Figure 1.

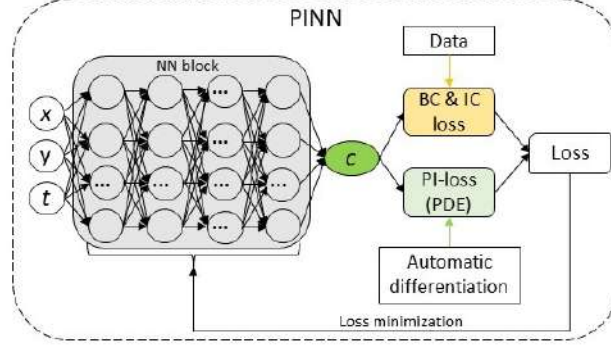


Fig. 1. Example of a PINN architecture. The network takes spatial coordinates (x, y) and time t as input and predicts the pollutant concentration $c(x, y, t)$. The loss function combines data consistency, PDE residuals, and boundary conditions

3 Implementation of PINN

We developed a versatile framework for PINN designed to solve a wide range of PDEs, including the advection-diffusion equation. This framework is built on PyTorch. By integrating GPU acceleration, our implementation enables efficient training and large-scale simulations. The framework is deployed on the high-performance computing (HPC) cluster at HSE University [12], which provides the necessary computational resources to handle complex simulations and large datasets. The network training was conducted on an NVIDIA V100 32 GB SXM GPU.

It is known that in certain cases, PINNs may fail to converge or produce sub-optimal results due to issues such as local minima in the loss function, inefficient parameter optimization, or architectural limitations [13,14]. To address these challenges, various strategies are implemented in the framework to enhance the convergence and accuracy of PINN. Next, we discuss some of the most promising methods.

One way to improve PINN performance is through advanced architectures like First-Order PINN (FO-PINN) [15]. FO-PINN modifies the standard PINN by using both the function values and their first derivatives to approximate solutions to PDEs. Unlike traditional PINN, which rely on second derivatives calculated using automatic differentiation, FO-PINN evaluates second derivatives indirectly using additional neural network outputs and corresponding loss terms. This approach reduces computational overhead and improves model accuracy.

Another approach to enhancing PINN performance is the Separable PINN (SPINN) [16]. SPINN represents the target function of multiple variables as a product or sum of functions of single variables. Instead of using collocation points across the entire domain, SPINN employs vectors of points along individual axes.

This separation of variables allows the use of simpler neural networks, reduces the number of trainable parameters, and improves computational efficiency.

One common reason for PINN convergence issues is the network getting trapped in local minima of the loss function. A promising strategy to mitigate this issue is to use sinusoidal mapping of input data [17]. This approach involves applying a sinusoidal activation function in the first layer:

$$\gamma(x) = \sin(2\pi(Wx + b)), \quad (11)$$

where W and b are trainable parameters.

The selection of weights in Eq. (10) is another critical factor in improving PINN convergence. An adaptive algorithm for selecting weights, such as the one proposed in [18], did not yield significant improvements in our experiments. During training, weights λ are adjusted to ensure that the terms in Eq. (10) are of the same order of magnitude. This adjustment not only reduces the residual error in initial conditions but also improves overall accuracy.

4 Results

4.1 Solution of the forward model

The forward problem involves modeling the distribution of pollutants in a computational domain, given a predefined velocity field. The goal is to use PINN to simulate the advection and diffusion processes and evaluate their accuracy in reproducing pollutant dispersion. For our applications, the forward problem is considered solved successfully if the mean squared error (MSE) between the PINN-predicted pollutant distribution and the finite element method (FEM) solution remains below 1×10^{-3} for all time steps. FEM is chosen as the reference method due to its well-established accuracy in solving partial differential equations, making it a reliable benchmark for evaluating PINN performance. Solving the forward problem first provides valuable insights into the model's behavior, simplifying the subsequent solution of the inverse problem.

We iteratively optimized the PINN architecture for the advection-diffusion equation. Initially, we tuned the architecture parameters manually; however, for improved results, one could employ stochastic search algorithms as outlined in [19]. The preliminary results for the PINN, which uses fully connected layers with 300 neurons per layer applied to the original equation 1, are presented in Figures 2 and 3. Specifically, Figure 2 shows the components of the loss function (left) and the total loss (right). The PDE loss curve levels off, suggesting inadequate training. Meanwhile, Figure 3 compares the PINN predictions (first row) to the FEM solutions (second row), along with the absolute error (third row) at times $t = 0, 1, 2$, and 3 seconds.

The optimal architecture incorporates First-Order PINN (FO-PINN) with $\lambda = 1000$ for initial and boundary conditions, together with ResNet blocks. ResNet blocks, which combine linear layers with residual connections, mitigate vanishing gradient issues and improve training stability. Each layer contains

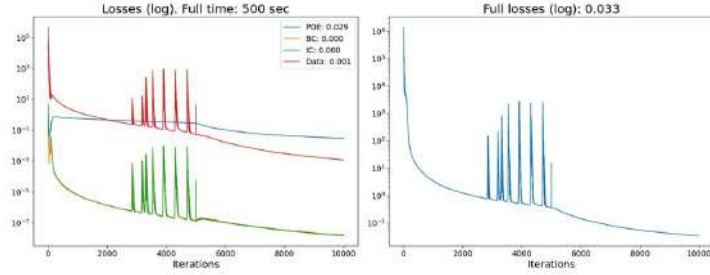


Fig. 2. Loss curves of PINN for the advection-diffusion equation

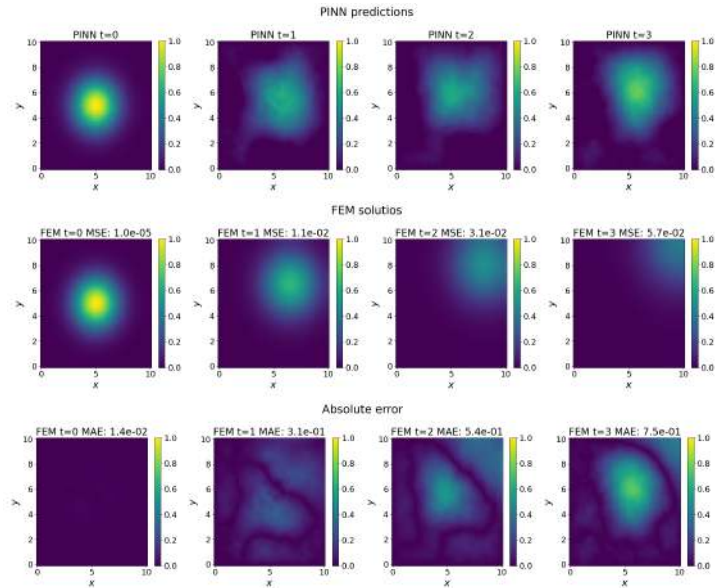


Fig. 3. Comparison of classical PINN results with FEM results for the advection-diffusion equation, $t = 0-3$

300 neurons. The weighting factor $\lambda = 1000$ was chosen to prioritize initial and boundary conditions.

The activation functions used are tanhshrink, with the $\sin 2\pi$ activation function applied to the first input layer. The optimizers used in this study are Adam and LBFGS, applied in a 50/50 ratio, with the total number of iterations set to 10000. In addition, the nondimensionalization procedure significantly improves the convergence behavior. From this point onward, all results are reported in dimensionless terms. The training process takes approximately 8 minutes when run on a GPU. The average training duration for each scenario examined in the

following sections was between 5 and 10 minutes. Figure 4 illustrates the tuned PINN architecture.

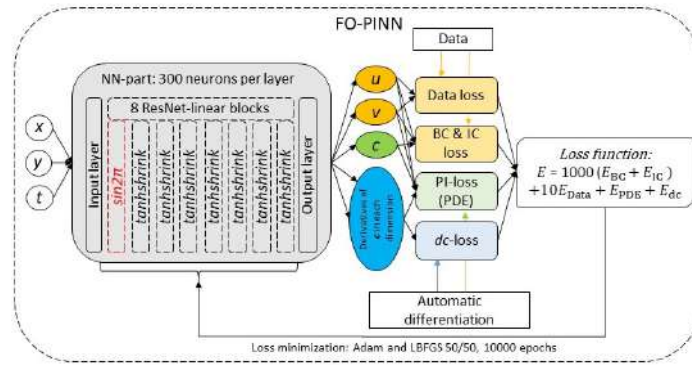


Fig. 4. FO-PINN architecture for solving the forward problem of the advection-diffusion equation

Figure 5 shows the loss curves, while Figure 6 illustrates the obtained solutions. The improved PINN achieves good agreement with the FEM results, with MSE values ranging from 1×10^{-5} to 1×10^{-6} and mean absolute error (MAE) values between 1×10^{-2} and 1×10^{-3} . These results demonstrate the effectiveness of the optimized PINN configuration in accurately modeling pollutant dispersion. With the optimal PINN setup in place, we now shift our focus to the inverse problem of identifying pollution sources using the same framework.

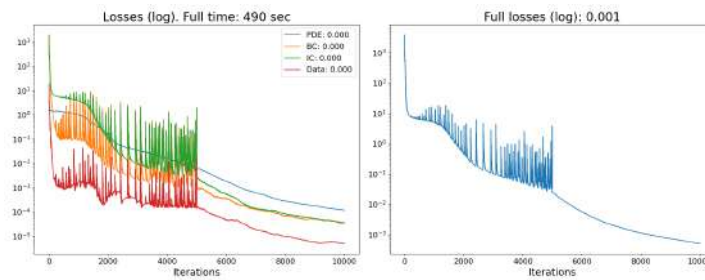


Fig. 5. Loss curves of the improved PINN for the normalized advection-diffusion equation

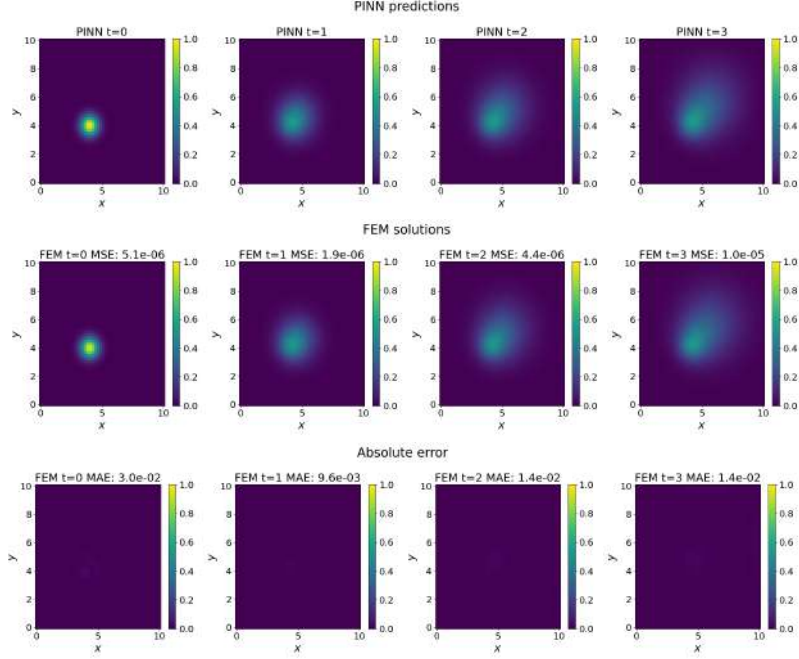


Fig. 6. Results for the forward problem of the normalized advection-diffusion equation: improved PINN

4.2 Solution of the inverse problem using synthetic data

The inverse problem involves determining the location and characteristics of an unknown pollution source based on limited observations of pollutant concentrations at specific points within the computational domain. This includes estimating both the coordinates and possibly the intensity of the source. Solving this problem has significant practical importance, as it enables the identification of pollution sources and the assessment of their environmental impact. In this section, we evaluate the effectiveness of PINN using test data generated by FEM to assess the accuracy and reliability of our model before applying it to real-world data. We define multiple synthetic scenarios, including cases with constant and variable winds, different diffusion coefficients, and varying numbers of sources.

To effectively address the inverse problem, we introduced several modifications to the PINN architecture. The FO-PINN architecture and the $\sin 2\pi$ activation function in the first layer were removed as they do not contribute to solving the inverse problem. The value of λ_{data} in Eq. (10) is set to 10000.

The final model configuration optimized for solving the inverse problem of the advection-diffusion equation is illustrated in Figure 7.

To identify the unknown source, we treat its coordinates as trainable parameters within the neural network. Similar to weights and biases, these parameters

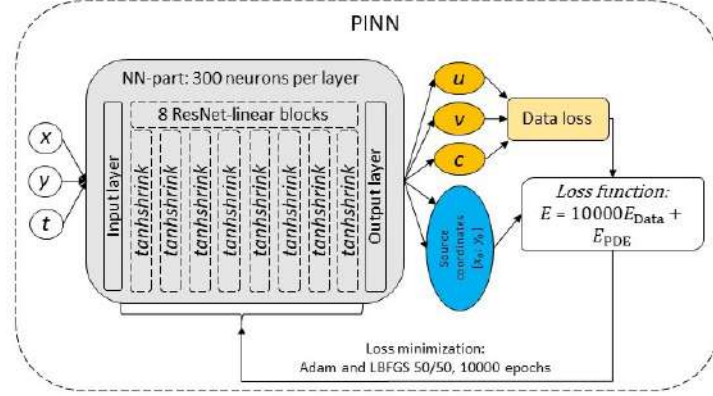


Fig. 7. Optimized PINN architecture for solving the inverse problem of the advection-diffusion equation

are iteratively updated using gradient-based optimization during the training process. An alternative approach would be to include additional outputs in the neural network and introduce corresponding loss function terms for these outputs. However, for the problem considered, this approach yielded unsatisfactory results and therefore not considered in this paper. In our simulations, the coordinates of the pollution source are initialized either as a randomly selected point within the computational domain or as its midpoint. During training, PINN substitutes these parameters into the governing equation and progressively refines the estimated source location.

In the first scenario, we place a point source at coordinates $(4, 4)$ in the domain $(10, 10)$ with a diffusion coefficient of $k = 0.5$, constant wind speeds $u = 0.7$ and $v = 0.7$, and consider a single time instance at $t = 3$. The Peclet number for this case is 11.8. Figure 8 presents the loss function plots and how the estimated coordinates converge towards the target source coordinates during training. In Figure 9, we present the reference solution obtained using FEM alongside the PINN solution with the identified pollution source coordinates, demonstrating a good agreement between the two.

Table 1 provides a comparison between the predicted source coordinates and the target values, along with the corresponding MSE. During the study, we observed that the number of training epochs had a more significant impact on the results than the number of collocation points. Our experiments were conducted for 10000–15000 epochs and 200–5000 collocation points. The results obtained for 15000 epochs with only 200 collocation points were comparable to those obtained with 10000 epochs and 2500 collocation points. The detailed comparison is presented in Table 2. This finding is particularly relevant for future studies, where real-world data availability may be limited to a small number of sensors.

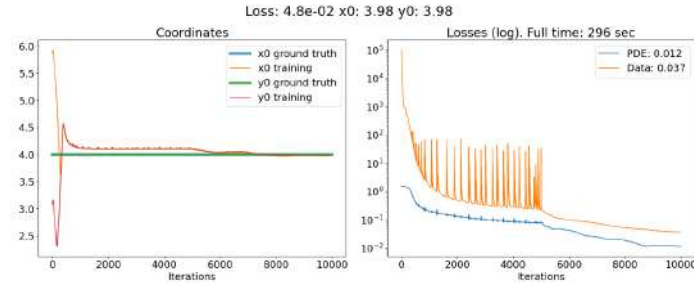


Fig. 8. The loss function plot and the process of locating the coordinates for $t = 3$ and constant wind speed

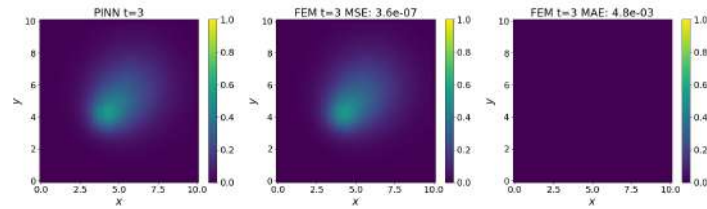


Fig. 9. Comparison of PINN results with FEM data for $t = 3$ and constant wind speed

Table 1. Comparison of true source coordinates with predicted coordinates using PINN for $t = 3$ and constant wind speed

	True coordinates	Predicted coordinates	MSE
x	4.0	3.979	4.4e-4
y	4.0	3.982	3.2e-4

Table 2. Predicted coordinates for three cases: 1) 10000 epochs, 2500 collocation points; 2) 15000 epochs, 2500 collocation points; 3) 15000 epochs, 200 collocation points

	1 case			2 case			3 case		
	True coordinates	Predicted coordinates	MSE	True coordinates	Predicted coordinates	MSE	True coordinates	Predicted coordinates	MSE
x	4.0	4.119	1.4e-2	4.0	4.060	3.5e-3	4.0	4.130	1.7e-2
y	4.0	4.091	8.2e-3	4.0	4.085	7.2e-3	4.0	4.099	9.7e-3

Next, we consider a more complex case involving variable wind speed field within the time range from 0 and 4. Figure 10 shows the loss function and evolution of source coordinates. This wind speed field is used throughout the paper unless stated otherwise. Figure 11 shows a comparison between the results of PINN and the FEM data used for training the network. As we can observe, compared to the previous case involving a single time instance, the current scenario presents some challenges for PINN. However PINN still accurately predicts the source coordinates, as shown in Table 3.

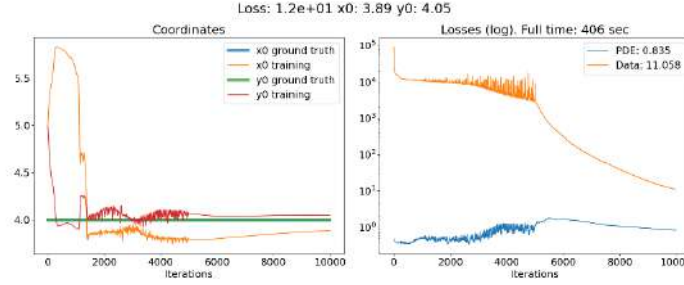


Fig. 10. The loss function plot and the process of locating the coordinates for variable wind speed, $t = 0-4$, and diffusion coefficient $k = 0.5$

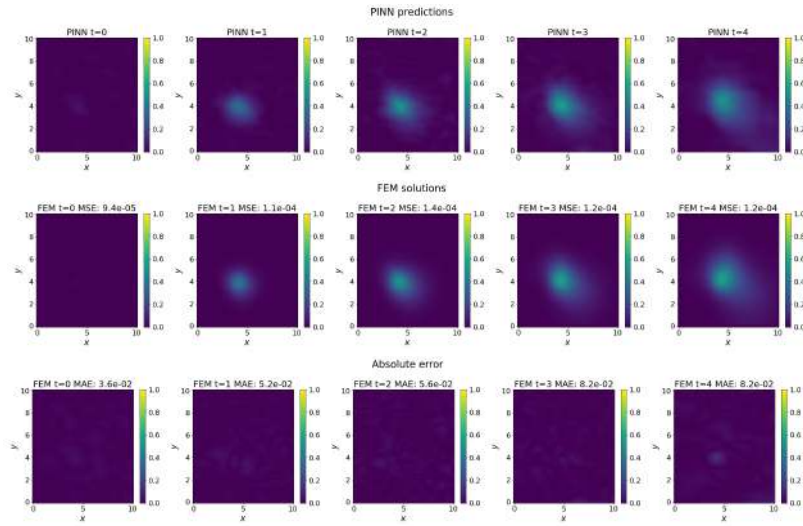


Fig. 11. Comparison of PINN results with FEM data for variable wind speed, $t = 0-4$ seconds, and diffusion coefficient $k = 0.5$

Table 3. Comparison of true source coordinates with predicted by PINN for variable wind speed, $t = 0-4$ seconds, and diffusion coefficient $k = 0.5$

	True coordinates	Predicted coordinates	MSE
x	4.0	3.885	1.3e-2
y	4.0	4.049	2.4e-3

Next, we consider a similar scenario but with a diffusion coefficient of 10^{-5} , with the corresponding Peclet number value of 590000. The process of locating the source coordinates and the training plot are presented in Figure 12. As the diffusion coefficient decreases, the transport of the substance by the wind

becomes more pronounced, slightly impacting the performance of PINN in terms of data learning (Figure 13) and coordinate prediction (Table 4). Despite these challenges, PINN continues to exhibit high efficiency in accurately determining the source coordinates.

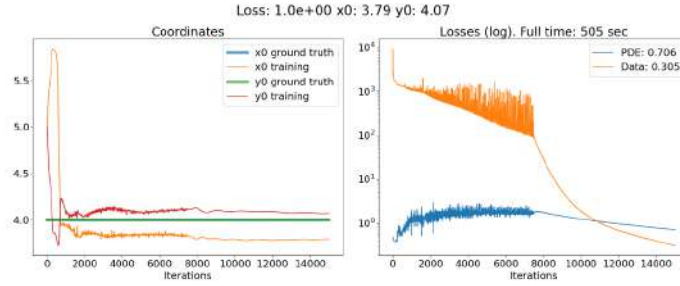


Fig. 12. The loss function plot and the process of locating the coordinates for variable wind speed, $t = 0-4$ seconds, and diffusion coefficient $k = 1e-5$

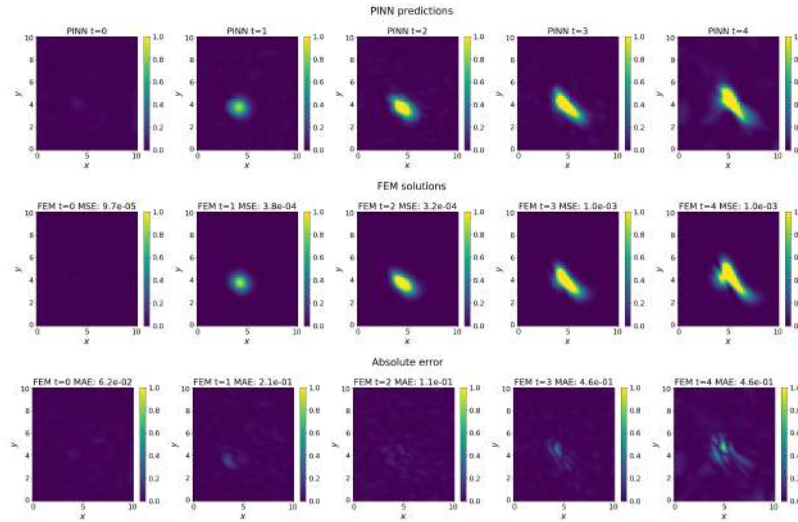


Fig. 13. Comparison of PINN results with FEM data for variable wind speed, $t = 0-4$ seconds, and diffusion coefficient $k = 1e-5$

Next, we consider a scenario similar to the previous one, but with two sources located at coordinates (4, 4) and (3, 6). Figure 14 illustrates the process of iden-

Table 4. Comparison of the true source coordinates with the ones predicted by PINN for variable wind speed, $t = 0-4$ sec, and diffusion coefficient $k = 1e-5$

	True coordinates	Predicted coordinates	MSE
x	4.0	3.793	4.3e-2
y	4.0	4.070	4.8e-3

tifying the source coordinates along with the corresponding loss function. Figure 15 presents the PINN predictions and their comparison with the reference data. Table 5 provides a quantitative comparison between the predicted and true source coordinates.

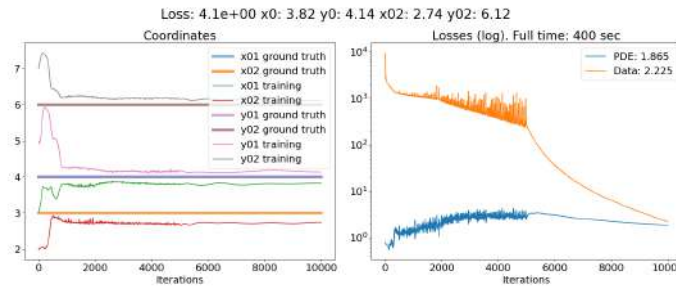


Fig. 14. The loss function plot and the process of locating the coordinates for two sources, variable wind speed, $t = 0-4$, and diffusion coefficient $k = 1e-5$

Table 5. Comparison of the target coordinates of two sources with the predicted coordinates by PINN for variable wind speed, $t = 0-4$ sec, and diffusion coefficient $k = 1e-5$

	True coordinates	Predicted coordinates	MSE
x1	4.0	3.823	3.1e-2
y1	4.0	4.137	1.9e-2
x2	3.0	2.740	6.7e-2
y2	6.0	6.121	1.5e-2

To validate the proposed approach, we apply PINN to a set of test pollution source coordinates within the domain $(10, 10)$: $(3, 3)$, $(4, 4)$, $(5, 5)$, $(6, 6)$, $(7, 7)$, $(4, 5)$, $(5, 2)$, $(6, 3)$, $(4, 6)$, and $(2, 4)$.

To assess the accuracy of the predicted source coordinates, we compare the coordinates obtained using the PINN-based algorithm with the test coordinates using the MSE metric.

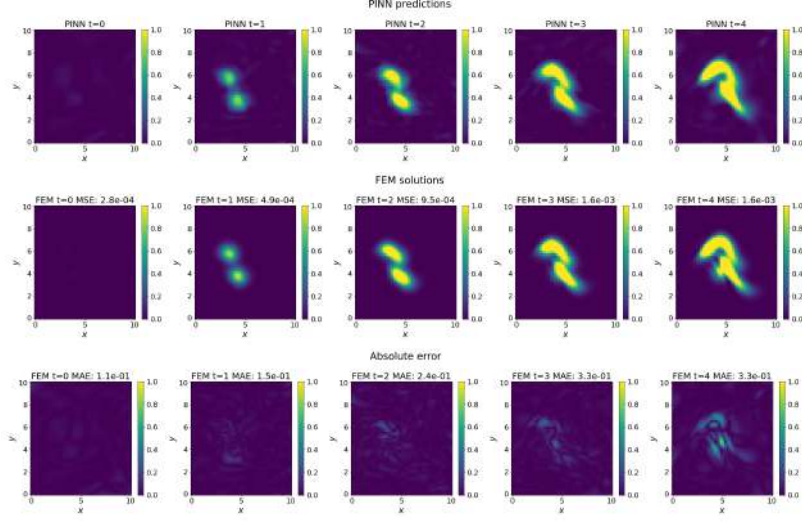


Fig. 15. Comparison of PINN results with FEM data for two sources, variable wind speed, $t = 0-4$, and diffusion coefficient $k = 1e-5$

Table 6 presents a comparison between the pollution source coordinates predicted by the PINN algorithm and the reference test coordinates. The results indicate that PINN successfully identifies the source locations in over 80% of the cases, with the error for any correctly predicted coordinate component not exceeding 10^{-1} .

We obtained robust results across various scenarios, including simple and complex wind fields, different diffusion coefficient values, and cases with both single and multiple sources. Additionally, we tested the PINN method on multiple pollution source coordinates, further confirming its reliability and adaptability. These findings demonstrate the flexibility and accuracy of the proposed approach.

4.3 Solution of the inverse problem using real data

In this section, we address the solution of the inverse advection-diffusion problem to determine the coordinates of the substance source, relying on real-world observation data provided by our industrial partner, which includes data on hydrogen sulfide concentration as well as wind speed and direction [20]. The real-world data represents a domain of 20 by 20 km, divided into a 100 by 100 grid with a 200-meter step. In this grid, the target source is located at the coordinate (37,44). To achieve satisfactory results, we increase the number of epochs and slightly adjust the ratio of the Adam and LBFGS optimizers: 10000 epochs for Adam and 3000 epochs for LBFGS. To ensure the comparability of MSE values calculated from different coordinate types (test and real), the nondimen-

Table 6. Verification results for different test coordinates

№	Test Coordinates	Predicted Coordinates	MSE	Result
1	(3,3)	(2.938, 3.193)	(3.9e-3, 3.7e-2)	✓
2	(4,4)	(3.793, 4.072)	(4.3e-2, 5.2e-3)	✓
3	(5,5)	(4.881, 4.845)	(1.4e-2, 2.4e-2)	✓
4	(6,6)	(5.892, 5.639)	(1.2e-2, 1.3e-1)	✓
5	(7,7)	(5.855, 3.929)	(1.3, 9.4)	✗
6	(4,5)	(3.813, 5.167)	(3.5e-2, 2.8e-2)	✓
7	(5,2)	(5.072, 1.865)	(5.2e-3, 2.1e-2)	✓
8	(6,3)	(5.909, 2.801)	(8.3e-3, 4.0e-2)	✓
9	(4,6)	(3.813, 6.103)	(3.5e-2, 1.1e-2)	✓
10	(2,4)	(5.849, 4.010)	(15, 9.3e-5)	✗ / ✓

sionalization procedure is applied. This approach helps to maintain the accuracy criterion of $1e-1$, which is important when comparing model performance across different scales.

Results of solving the inverse advection-diffusion problem using real data are presented. Figure 16 shows that the approximate coordinates are determined in 500 epochs, which takes less than 20 seconds. However, to achieve optimal accuracy, the increased training time may not have been sufficient. Increasing the number of LBFGS epochs to 5000 resulted in NaN values in the loss function, underscoring the importance of balancing training time and network stability to avoid overfitting or divergence during the optimization process. Figure 17 demonstrates how PINN adapts to the real data. While some uncertainty is present in the results obtained by PINN, it does not hinder its ability to locate the source coordinates, which are sufficiently close to the target values, as shown in Table 7. As a result of the conducted research, we observe that PINN method successfully solves inverse advection-diffusion problems, demonstrating high accuracy in determining the source coordinates based on real data. This confirms its effectiveness and potential for further application in this field.

Table 7. Comparison of true source coordinates with predicted coordinates using PINN

	True coordinates	Predicted coordinates	MSE
x	44.0	44.018	3.2e-6
y	37.0	38.379	1.9e-2

5 Conclusions

In this study, we have explored the application of PINNs for solving the inverse advection-diffusion problem to localize pollution sources. The PINN solutions

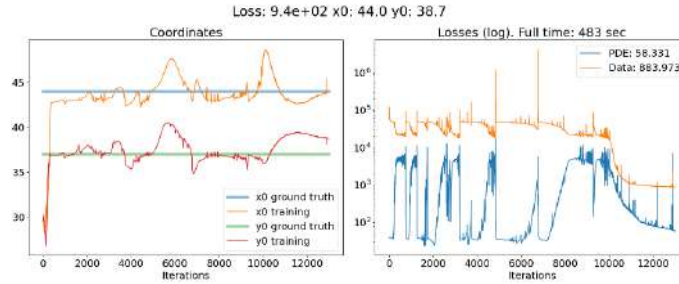


Fig. 16. Loss curves and coordinate search process for solving the inverse advection-diffusion problem with real-world data

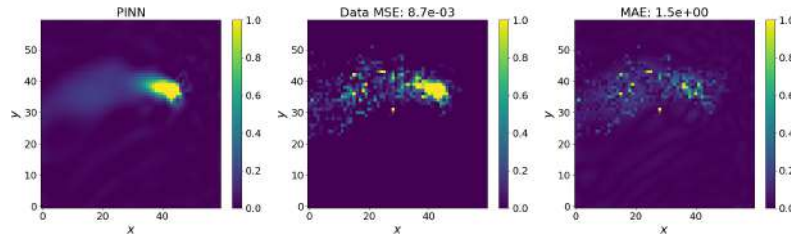


Fig. 17. Comparison of PINN results with FEM data for real-world data. The maximum substance concentration value was limited to 1, ensuring better clarity for the analysis

have been validated against reference solutions obtained using FEM. To extend the PINN framework to inverse problems, source coordinates were incorporated as trainable parameters within the network.

The proposed approach successfully identifies pollution source coordinates using real-world concentration and wind data, demonstrating its applicability for environmental monitoring. The accuracy of the method depends on the availability of pollutant concentration measurements and wind field data, which can be obtained from meteorological observations or by solving the Navier-Stokes equations. Although this study does not address solving the forward Navier-Stokes problem, future research may focus on integrating PINNs for simultaneous flow and pollutant dispersion modeling.

Despite their potential, PINNs require careful tuning of architecture and hyperparameters, making their application problem-specific and computationally intensive. While in this work, the optimization of the PINN architecture has been performed manually, automating this process through systematic guidelines and global optimization techniques could enhance efficiency and scalability. Future work will focus on expanding the validation of the approach across diverse atmospheric conditions and pollutant scenarios. Enhancing model accuracy while reducing computational costs remains a key objective. Additionally, transitioning

from two-dimensional to three-dimensional modeling will enable more realistic simulations, incorporating factors such as altitude and terrain effects, which are critical for accurate pollutant dispersion modeling.

Acknowledgments

This research was supported in part through computational resources of HPC facilities at HSE University.

References

1. C.R. Mechoso and A. Arakawa. *Numerical models / General Circulation Models*, page 153–160. Elsevier, 2015.
2. S. Cieslik. *Dispersion and Transport of Atmospheric Pollutants*, page 47–71. Springer Netherlands, 1991.
3. M. Waqas, U. W. Humphries, B. Chueasa, and A. Wangwongchai. Artificial intelligence and numerical weather prediction models: A technical survey. *Natural Hazards Research*, November 2024.
4. Yichen Yao, Xiaohui Zhong, Yongjun Zheng, and Zhibin Wang. A physics-incorporated deep learning framework for parameterization of atmospheric radiative transfer. *Journal of Advances in Modeling Earth Systems*, 15(5):e2022MS003445, 2023.
5. S. Goswami, A. Bora, Y. Yu, and G.E. Karniadakis. *Physics-Informed Deep Neural Operator Networks*, page 219–254. Springer International Publishing, 2023.
6. M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
7. S. Cuomo, M. De Rosa, F. Giampaolo, S. Izzo, and V. Schiano Di Cola. Solving groundwater flow equation using physics-informed neural networks. *Computers & Mathematics with Applications*, 145:106–123, 2023.
8. Roberto Mamud, Vinicius Albani, Helio Migon, and Antônio Silva Neto. Estimation of pollution sources with physics-informed neural network. 11 2022.
9. I. Chuprov, J. Gao, D. Efremenko, E. Kazakov, F. Buzaev, and V. Zemlyakov. Optimization of physics-informed neural networks for solving the nonlinear Schrödinger equation. *Doklady Mathematics*, 108, 02 2024.
10. X. Jin, S. Cai, H. Li, and G.E. Karniadakis. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, 2021.
11. I. A. Chuprov, J. Gao, D. S. Efremenko, F. A. Buzaev, and V. V. Zemlyakov. Solution of the multimode nonlinear Schrödinger equation using physics-informed neural networks. *Doklady Mathematics*, 110(Suppl. 1):S15–S24, 2024. Russian text published in *Doklady Rossiiskoi Akademii Nauk. Matematika, Informatika, Protsessy Upravleniya*, Vol. 520, No. 2, pp. 19–29.
12. P. S. Kostenetskiy, R. A. Chulkevich, and V. I. Kozyrev. HPC resources of the Higher School of Economics. *Journal of Physics: Conference Series*, 1740(1):012050, 2021.

13. S. Wang, Y. Teng, and P. Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.
14. S. Wang, H. Wang, and P. Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, October 2021.
15. R.J. Gladstone, M.A. Nabian, N. Sukumar, A. Srivastava, and H. Meidani. FO-PINNs: A first-order formulation for physics informed neural networks, 2023.
16. J. Cho, S. Nam, H. Yang, S.-B. Yun, Y. Hong, and E. Park. Separable PINN: Mitigating the curse of dimensionality in physics-informed neural networks, 2023.
17. J.C. Wong, C. C. Ooi, A. Gupta, and Y.-S. Ong. Learning in sinusoidal spaces with physics-informed neural networks. *IEEE Transactions on Artificial Intelligence*, 5(3):985–1000, March 2024.
18. S. Wang, X. Yu, and P. Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, January 2022.
19. F. Buzaev, J. Gao, I. Chuprov, and E. Kazakov. Hybrid acceleration techniques for the physics-informed neural networks: a comparative analysis. *Machine Learning*, 113(6):3675–3692, 2023.
20. CityAir. Cityair: Intelligent air quality monitoring. <https://cityair.ru/ru/>, 2024. Date of access: 01.12.2024.

Construction of self-healing PDMS materials models by supercomputer MD simulations*

T.M. Makarova, E.V. Bartashevich

South Ural State University, Chelyabinsk, Russia

To investigate self-healing and another properties of PDMS based materials we elaborated a computational procedure to obtain a three-dimensional atom-level structure of these multicomponent polymer materials. We obtained several condensed material structures from gas phase mixtures of the required components using the MD simulation performed on the “Tornado” supercomputer. In the obtained structures, we reproducibly observed a formation of ionic aggregates which are the prominent nature of ionomers, but in the case of PDMS materials, they were found to have a more complex structure with inner and outer surrounding layers.

Keywords: MD simulation, supercomputer, self-healing materials, PDMS.

1. Introduction

Self-healing materials with their capability to heal emergent mechanical defects are of increasing interest as a component of the sustainable development concept due to their increased durability and service life with reduced material waste [1]. They are expected to be widely used in different areas and devices [2]. Materials on the basis of polydimethylsiloxanes (PDMS) have demonstrated a high self-healing efficiency and a wide range of mechanical properties, thus being a very perspective class of such kind of materials [3]. The properties of known PDMS materials are described to be sensible to qualitative and quantitative composition of a pre-polymerization mixture, and this dependence is not always incremental and obvious. Thus, any further directed material development would require a reliable model of a structure-properties relationship for these polymers. The first stage of such a model would be an atomistic three-dimensional structure of this type of materials. However, irregular structure of polymer materials make them inexpedient objects for diffractional and other experimental methods.

In this research we aimed to adjust the MD simulations protocol in order to provide reliable structures for this class of self-healing materials and shed some light on the nature of self-healing mechanisms. Noteworthy, the starting structure for the simulation is a crucial point for the reliability of the molecular dynamic model. The available proprietary software for initial polymer cell construction rely on different probabilistic algorithms which, in fact, arbitrary mix the building blocks. This approach does not take into account chemical interactions that guide the self-organisation of a modelled material, which could play the crucial role in a mutual arrangement of polymer chains and other system components. So, it is critically important to start the MD simulations from the structure that was found as a minimum on a potential energy surface via virtual sliding on it guided by intermolecular interactions between the polymer system components. To carry out this, we elaborated an MD simulation protocol that in basic principles resembles some kind of the simulated annealing protocol [4], but heating and the temperature value here is a real kinetic energy that provides overcoming potential energy barriers and movement of the system towards the most optimal structure. Upon this protocol, the system was first heated to a high temperature and then slowly cooled down to the ambient temperature, during which molecules and ions aggregated according to intermolecular interactions modelled by a force field.

The first materials which were aprobed in our protocol were so-called examples of the

*The research was supported by the Ministry of Science and Higher Education (FENU 2024-0003). All the MD simulations were performed using Tornado supercomputer of the South Ural State University.

“siloxane equilibration” rubber. It is obtained by anionic polymerization of octamethylcyclotetrasiloxane molecules D4 with slight admixture of bis-D4, a product of a radical recombination of two D4 molecules with a C-C bond formation, that provides ethylene crosslinks in the resulting material (Fig. 1). The “siloxane equilibration” material also demonstrated a significant self-healing ability depending on the bis-D4 proportion [5, 6].

Our approach enabled to observe aggregation of positively charged counterions and self-organisation of charged and polar groups and molecules around these aggregates. These structures assembled reproducibly in different cells with varying composition and in different MD recalculation with randomly distributed components owing to the thermal interfusion.

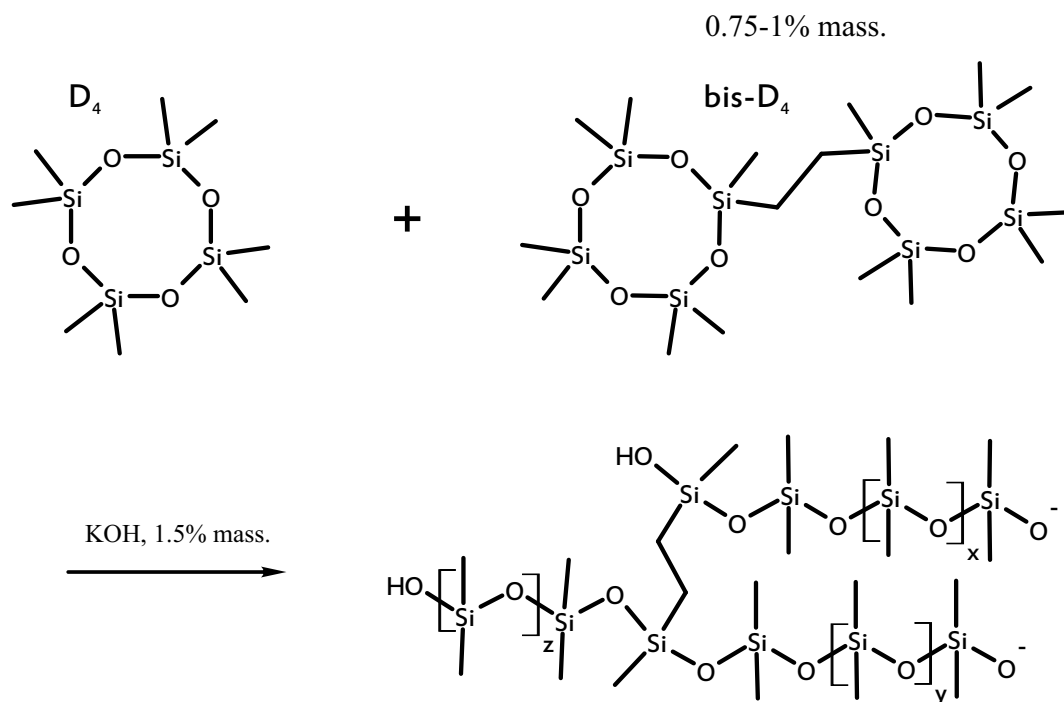


Figure 1. Polymerisation reaction for one of the “siloxane equilibration” material with bis-D4 admixture

2. Methods

2.1. System construction

The pre-condensation structures were constructed on the basis of mass ratio between D4, bis-D4 and initiator (KOH). Average chain lengths were calculated from the monomer/initiator ratio for every system. The chains were generated in the system with lengths distributed according to the Poisson distribution, and positioned in the cubic cell at a steady pace from each other in every three dimensions.

To obtain a reliable three-dimensional structure *in silico*, the cell had to contain enough polymer molecules to create a phase volume where all the chains had enough space to unfold and intertwist without artificial restrictions by borders of the cell (i.e. in a cell with insufficient amount of molecules, the polymer chains would be restricted and forced to adopt a globule-like conformation, which would be an artifact and distort the model). Thus, there could not be less than 2 molecules in every dimension in the initial cell before condensation, and space adopted by any molecule in the final condensed phase should be no more than half of the edge of the final

condensed cell. The molecules should be evenly distributed in the cubic cell for further isotropic condensation. Therefore, after several preliminary attempts of system construction, every cell contained 216 polymer chains, among which only 11 had an ethylene crosslink originated from the bis-D4 reagent. They were distributed in the cell dimensions as $2 \times 6 \times 18$. The number of chains in every dimension of the cell and in total were selected in order to reproduce the required D4/bis-D4 ratio. The typical init cell size was around 38 nm.

The configuration around the ethylene crosslink from bis-D4 molecule may be di-substituted (as it is usually depicted on typical chemical formulas of the “siloxane equilibration”, when every Si atom neighbouring to the crosslink is connected to one PDMS chain and one -OH group; so, the crosslink is substituted by 2 PDMS chains), tri-substituted (with only one -OH group, as it is depicted on Fig. 1) and tetra-substituted (without -OH groups in the crosslink surroundings). Though in our program di:tri:tetra ratio is tunable, we made it 1:1:1 for all the systems to observe the behavior of every variant of the crosslink configuration.

All these procedures were carried out by the Python script applying the *cmd* module.

After that, randomly distributed ions and water molecules were added to the cell using *gmx genion* and *gmx solvate* instruments of GROMACS software, respectively. In total, the systems contained from 112.9 to 115.6 thousands of atoms.

2.2. MD simulations

The molecular mechanical models were prepared using the General Amber force field [7]. The missing covalent force field parameters for the “siloxane equilibration” monomers were extracted from DFT calculations with M06-2X/6-311++G** basis, along with organosilicon compounds parameters from [8]. Partial charges for these residues were assessed using the RESP model [9].

The MD simulations were carried out by GROMACS software [10,11], version 2019.4. The integration time step was 2 fs with writing coordinates to the XTC trajectory file every 10 thousand steps. The LINCS algorithm controlled all the bonds lengths during the simulations [12]. The velocity rescaling thermostat with additional stochastic correction [13] was applied for the temperature control, while Berendsen barostat [14] maintained pressure in NPT simulations. Long-range electrostatic interactions were treated by Ewald particle mesh method with the sixth order of interpolation and a 0.1 nm grid step [15].

Every MD simulation was preceded by a geometry optimisation using the Broyden-Fletcher-Goldfarb-Shanno method [16]. The general protocol of molecular modeling is described in (Table 1).

2.3. Details of supercomputer calculations

The size of the system cells (113-116 thousands of atoms) and the required lengths of MD trajectories (from 20 to 500 ns) required significant computational sources with deep parallelisation. So, all the MD simulations of this work were performed using the “Tornado” supercomputer [17] on 24 CPU nodes (Intel Xeon 5680X, 6 cores, 3.33 GHz) for every single simulations. The performance in every trajectory calculation is given in Table 2. The NVT dynamics proceeded a bit slower due to a high load imbalance (up to 60%), but these were the shortest simulations in all the protocols. This could be explained by an intensive particle exchange between the domains, into which the cell is divided for the parallel calculation, so changing of particle assignment to a node occurred in larger quantities. The domain decomposition of the cell volume between the nodes was $2 \times 3 \times 4$ in all the cases.

Table 1. MD simulation protocol applied for the “siloxane equilibration” materials structure obtaining. All the simulations and energy optimizations are described sequentially

№	Time	Temperature	Pressure
1	L-BFGS optimization		
2	20 ns	1073 K	NVT
3	L-BFGS optimization		
4	3 ns	1073 K	30 bar
5	50 ns	1073 K to 298 K	30 bar
6	50 ns	298 K	30 bar
7	L-BFGS optimization		
8	500 ns	298 K	1 bar

Table 2. Performance of the MD simulations of the PDMS-bases systems on the “Tornado” supercomputer, ns per day. “WC” denotes water content in mass. %. *NVT* denotes the first simulation at a constant volume and 1073 K, *NPT cond.* – the second one with the temperature decrease, *NPT final* – the final 500 ns one under ambient conditions

Trajectory	WC, 0%	WC, 0.47%	WC, 1.4%
NVT	30.8	31.3	31.1
NPT cond.	40.4	42.1	42.2
NPT final	42.2	41.9	42.2

3. Results and discussion

3.1. Development of a system preparation protocol

The composition of the “siloxane equilibration” mixture was reconstructed from mass ratios of the initial components: D4, bis-D4 and the polymerization initiator. As a starting point, ratios of the sample, obtained by R.M. Islamova and co. at the Institute of Chemistry of the Petersburg State University, exactly 0.75% mass. bis-D4 and 1.5% mass. KOH, was applied. Every effective initiator molecule was considered to start one uniq polymer chain, taking into account that every crosslink region of the bis-D4 fragment consumes two anions of the initiator, thus every ethylene crosslink connects two polymer chains. After calculation of a number of polymer molecules and their average length, the chains were generated with lengths distributed according to the Poisson distrubution and uniformly spread in the cubic cell (Fig. 2A). The counterions, unconsumed initiator ions were added to the cell further.

Also, polysiloxanes can absorb some amount of water. Therefore, we created 3 equivalent systems and added 0.47% or 1.4% mass. of water molecules to the second and the third systems

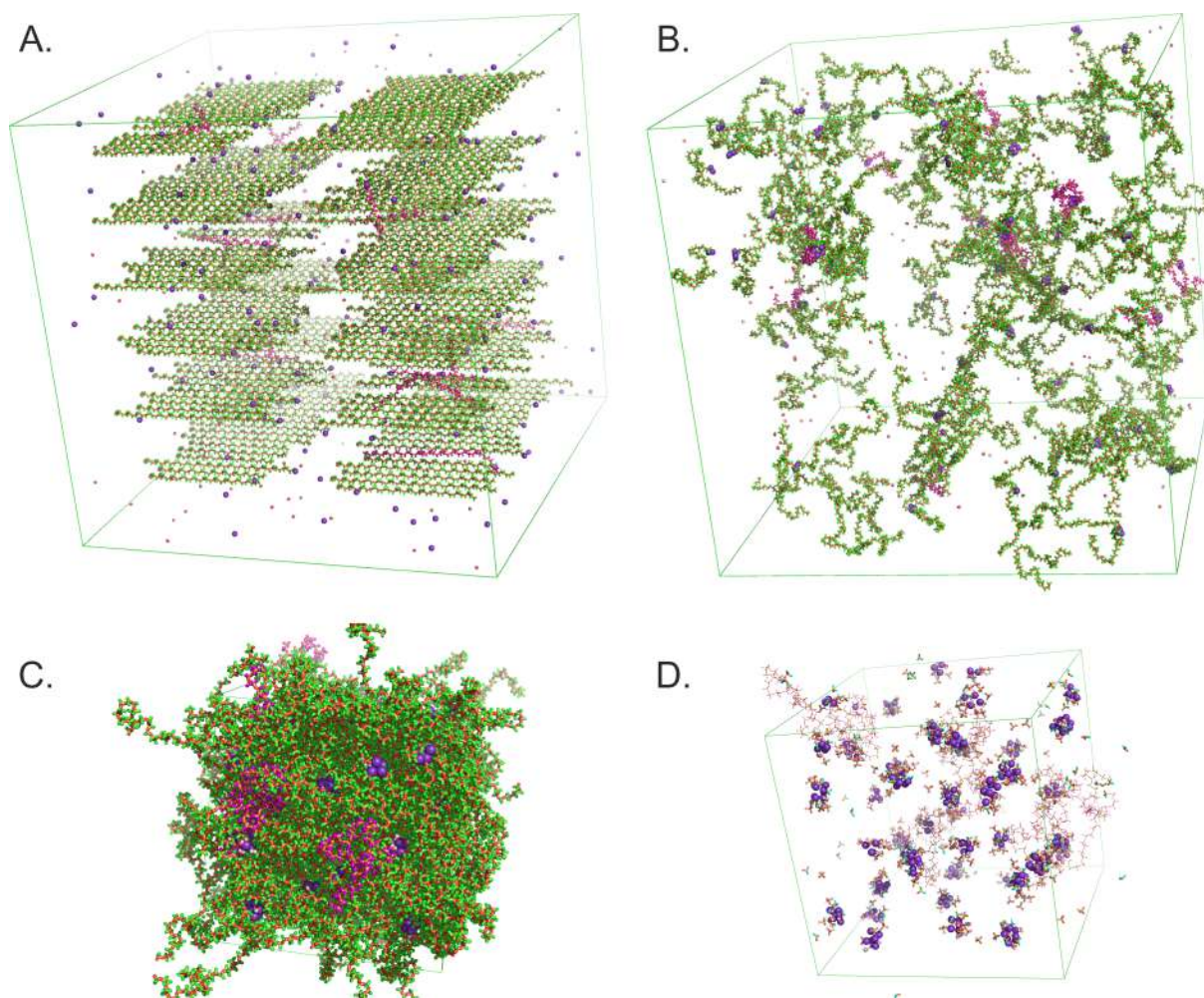


Figure 2. Condensation of the PDMS material upon the MD simulations protocol. The linear PDMS chains are shown in green, while those ones with ethylene crosslinks – in magenta. The K^+ ions are depicted as purple spheres, while water molecules – as red-and-white spheres. **A.** The initially constructed system. **B.** The system after 20 ns of the NVT simulation at 1073 K. **C.** The system after the NPT condensation. **C.** The system after the NPT condensation in details. The most of PDMS chains are hidden except carrying crosslinks (magenta), their negatively charged terminal residues (cyan) and polar initial chain residues (pink)

according to the thermogravimetric experiment (with 0.47% or 1.4% mass loss upon heating up to 100 and 200°C, respectively) carried out with a material sample in our laboratory.

To reconstruct the three-dimensional organisation of the material, a stepwise interfusion and condensation molecular dynamics protocol was applied (Table 1). At the first step, the 20 ns NVT dynamics at 1073K was calculated. Upon this simulation, the polymer chains and other components interfused intensively remaining spread all over the cell without significant condensation. Upon this dynamics, K^+ counterions retained close to the anionic groups. Some of them formed aggregates of several molecules, which, however, were not complete stable upon this simulation and underwent ion exchanges with each other. The example of resulting structures of these simulations is shown on Fig. 2B.

After that, NPT simulation with gradual temperature decrease was performed. The temperature first was retained at 1073 K for 3 ns, then linearly declined down to the ambient 298 K temperature during next 50 ns, and this value was maintained during the last 50 ns. At the normal pressure (1 bar) and standart cell compressibility the cell sizes almost didn't react on

the temperature decrease (i.e. system condensation was too slow). So, we had to increase the pressure up to 20–30 bar and the cell compressibility up to 10^{-4} bar $^{-1}$, and this was just enough high to provide steady but smooth compression of the system upon condensation and without urging the LINCS algorithm to interrupt the simulation. With these parameters, the system smoothly compressed into the condensed phase with final density around 0.954 (Fig. 3), which fell within the 0.91–1.00 diapason for such kind of PDMS materials [18].

3.2. Ion aggregation in PDMS materials during the MD simulations

Upon the condensation in the MD simulations, the cationic aggregates enlarged up to 5–6 cations in average (Fig. 2C). All the water molecules and the unconsumed initiator ions diffused to the cationic aggregates, therefore, the space between the aggregates was filled with low-polarity PDMS chains only. The molecular basis of the ionic aggregates formation was provided by both electrostatic and van-der-Vaals interactions reproduced in the model by the forcefield. Electrostatic attraction between the anionic PDMS tails and the cations in combination with hydrophobic expulsion of all charged and polar groups (including water) gathered all them together into compact aggregates, where the terminal anionic groups connected the K^+ ions at a 4–5 Å distance, which is significantly lesser then in solid crystals, but still is a rather close distance. The polymer chains remained rather extended upon the phase condensation due to connections of all its terminal and initial residues to the ionic aggregates and absence of any solvent in the non-polar medium of the PDMS chains, so the system did not undergo any kind of a coil-globule phase transition during the *in silico* condensation.

The final trajectories of 500 ns length were obtained at 298 K and 1 bar from resulting structures of the previously described condensation simulations. Upon the pressure change, a very slight decrease of density (around $3/10^{-3}$) was detected. The cationic aggregates retained stable upon all the simulations, no ion exchange between them were detected. Such the aggregates are typical for polymers with charged groups neutralized by corresponding counterions, called ionomers [19]. In the case of PDMS materials these aggregates appeared to have a more complex structure: anionic terminal groups intercalated inside the cationic aggregates between the very K^+ ions, while water molecules and relatively polar initial chain groups were found to form the outer layers of the aggregates. Another feature of the siloxane polymer was that every chain has one or, in the case of chains carrying an ethylene crosslink, two negatively charged terminal groups, every of which is connected to the cationic aggregate. Thus, every polymer molecule is involved in the so-called “restricted mobility region” (RMR) in the ionomer, and several polymer molecules may form a connection between different aggregates (see Fig. 2D). Though some ionomer materials also demonstrated self-healing abilities [20–22], PDMS materials are known to demonstrate a high effectiveness in uninduced self-healing under ambient conditions, while most of the ionomers require heating or another kind of activation for significant self-healing effectiveness. The role of the structural features in the PDMS self-healing ability is a subject of a further deeper investigation.

Thus, the condensation of the polymer mixture upon the MD simulations provided formation of extremely stable ionic aggregates within the PDMS materials which are described for another polymers with a moderate content of ionic groups and their counterions. The structure, distribution and organisation of polysiloxane chains around the detected aggregates is expected to be the key to the connection between the structure and the properties of the PDMS materials. A further investigation is expected to establish the role of the size and the structure of the aggregates and their surroundings in the self-healing capability. The elaborated method can be easily applied to another materials with different ratios of components, ion and initiator specificity, or with additional copolymer blocks.

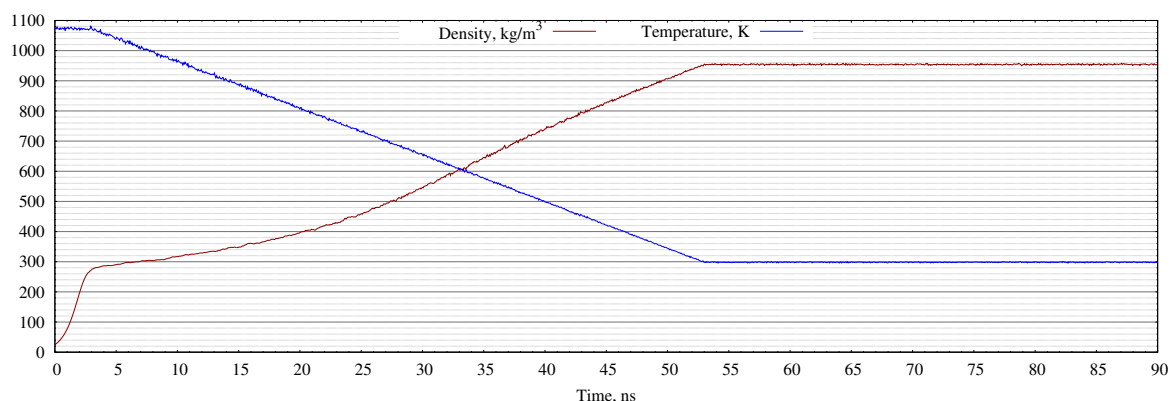


Figure 3. Temperature decrease and density increase upon the PDMS material condensation in the NPT MD simulation

References

1. Li B., Cao P.-F., Saito T., Sokolov A.P. Intrinsically self-healing polymers: From mechanistic insight to current challenges // *Chemical Reviews*. 2022. Vol. 123, no. 2 P. 701–735. DOI: 10.1021/acs.chemrev.2c00575.
2. Bergman S.D., Wudl F. *Re-Mendable Polymers* // *Self Healing Materials*. Springer Netherlands, 2007. P. 45–68. DOI: 10.1007/978-1-4020-6250-6_3.
3. Deriabin K.V., Filippova S.S., Islamova R.M. Self-healing silicone materials: Looking back and moving forward // *Biomimetics*. 2023. Vol. 8, no. 3. P. 286. DOI: 10.3390/biomimetics8030286.
4. Rutenbar R. Simulated annealing algorithms: an overview // *IEEE Circuits and Devices Magazine*. 1989. Vol. 5, no. 1. P. 19–26. DOI: 10.1109/101.17235.
5. Zheng P., McCarthy T.J. A surprise from 1954: Siloxane equilibration is a simple, robust, and obvious polymer self-healing mechanism // *Journal of the American Chemical Society*. 2012. Vol. 134, no. 4. P. 2024–2027. DOI: 10.1021/ja2113257.
6. Schmolke W., Perner N., Seiffert S. Dynamically cross-linked polydimethylsiloxane networks with ambient-temperature self-healing // *Macromolecules*. 2015. Vol. 48, no. 24. P. 8781–8788. DOI: 10.1021/acs.macromol.5b01666.
7. Wang J., Wolf R.M., Caldwell J.W. et al. Development and testing of a general Amber force field // *J. Comput. Chem*. 2004. Vol. 25. P. 1157–1174. DOI: 10.1002/jcc.20035.
8. Dong X., Yuan X., Song Z., Wang Q. The development of an Amber-compatible organosilane force field for drug-like small molecules // *Physical Chemistry Chemical Physics*. 2021. Vol. 23, no. 22. P. 12582–12591. DOI: 10.1039/D1CP01169C.
9. Bayly C.I., Cieplak P., Cornell W., Kollman P.A. A well-behaved electrostatic potential based method using charge restraints for deriving atomic charges: the RESP model // *J. Phys. Chem*. 1993. Vol. 97, no. 40. P. 10269–10280. DOI: 10.1021/j100142a004.
10. van der Spoel D., Lindahl E., Hess B. et al. GROMACS: fast, flexible, free // *J. Comput. Chem*. 2005. Vol. 26. P. 1701–1718. DOI: 10.1002/jcc.20291.

11. van der Spoel D., Lindah E., Hess B., Kutzner C. GROMACS 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation // *J. Chem. Theory Comput.* 2008. Vol. 4. P. 435–447. DOI: 10.1021/ct700301q.
12. Hess B., Bekker H., Berendsen H.J.C., Fraaije J.G.E.M. LINCS: A linear constraint solver for molecular simulations // *J. Comput. Chem.* 1997. Vol. 18, no. 12. P. 1463–1472. DOI: 10.1002/(SICI)1096-987X(199709)18:12<1463::AID-JCC4>3.0.CO;2-H.
13. Bussi G., Donadio D., Parrinello M. Canonical sampling through velocity rescaling // *J. Chem. Phys.* 2007. Vol. 126. P. 014107–014106. DOI: 10.1063/1.2408420.
14. Berendsen H., Postma J., van Gunsteren W. et al. Molecular dynamics with coupling to an external bath // *J. Chem. Phys.* 1984. Vol. 81. P. 3684–3690. DOI: 10.1063/1.448118.
15. Darden T., York D., Pedersen L. Particle mesh Ewald: An $N\log(N)$ method for Ewald sums in large systems // *J. Chem. Phys.* 1993. Vol. 98. P. 10089–10092. DOI: 10.1063/1.464397.
16. Byrd R., Lu P., Nocedal J. A limited memory algorithm for bound constrained optimization // *SIAM J. Scientific. Statistic. Comput.* 1995. Vol. 16. P. 1190–1208. DOI: 10.1137/0916069.
17. Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC Resources of South Ural State University // *Parallel Computational Technologies. PCT 2022*. Vol. 1618. Springer, Cham, 2022. P. 43–55. *Communications in Computer and Information Science*. DOI: 10.1007/978-3-031-11623-0_4.
18. Seethapathy S., Gorecki T. Applications of polydimethylsiloxane in analytical chemistry: A review // *Analytica Chimica Acta*. 2012. Vol. 750. P. 48–62. DOI: 10.1016/j.aca.2012.05.004.
19. Eisenberg A., Hird B., Moore R.B. A new multiplet-cluster model for the morphology of random ionomers // *Macromolecules*. 1990. Vol. 23, no. 18. P. 4098–4107. DOI: 10.1021/ma00220a012.
20. Kalista S.J., Ward T. C. Thermal characteristics of the self-healing response in poly(ethylene-co-methacrylic acid) copolymers // *Journal of The Royal Society Interface*. 2006. Vol. 4, no. 13. P. 405–411. DOI: 10.1098/rsif.2006.0169.
21. Varley R.J., van der Zwaag S. Development of a quasi-static test method to investigate the origin of self-healing in ionomers under ballistic conditions // *Polymer Testing*. 2008. Vol. 27, no. 1. P. 11–19. DOI: 10.1016/j.polymertesting.2007.07.013.
22. Varley R. Ionomers as Self Healing Polymers // *Self Healing Materials*. Springer Netherlands, 2007. P. 95–114. DOI: 10.1007/978-1-4020-6250-6_5.

Research and optimization pruning method in train process: DPIREC

D.A. Novikov, D.Y. Buryak

The faculty of Computational Mathematics and Cybernetics
of Lomonosov Moscow State University

This work focuses on the implementation and investigation of a novel pruning method for deep convolutional neural networks, DPIREC (Dynamic Pruning by Importance of Random Excluded Channels), which enables pruning directly during training. The core idea of this method is to apply random masks to the channels of convolutional layers during each training iteration, temporarily excluding certain channels from the training process. Subsequently, the importance values of the channels used in the current step are updated. Repeating these actions over multiple training epochs allows the identification of channels with the least impact on the loss function, which can then be permanently excluded from the pruned neural network. We conducted a comprehensive analysis of approaches to assessing parameter relevance. Based on this, a new approach was proposed that takes into account the dynamics of changes in model accuracy. Comparative experiments demonstrated the superiority of the DPIREC method over several existing techniques. When pruning the ResNet18 convolutional neural network during training on the CIFAR-100 dataset by 40%, the accuracy loss was 0.89%.

Keywords: convolutional neural networks, dynamic pruning, structured pruning.

1. Introduction

Modern advancements in deep learning are driven not only by the development of numerous novel neural network architectures and the engagement of a large number of specialists but also by continuously increasing computational power. Deep neural networks can encompass dozens or even hundreds of hidden layers, with each layer containing up to hundreds of neurons, resulting in an immense number of parameters [1]. Training such networks has become feasible due to the widespread availability of powerful parallel computing devices, such as GPUs (Graphics Processing Units) and TPUs (Tensor Processing Unit) [2]. However, many users lack access to high-performance computational resources equipped with thousands of CUDA cores.

Nevertheless, it is well-established that many parameters within a neural network are redundant and exert minimal influence on the final outcome. This insight has inspired the concept of reducing model weights and operations with minimal quality degradation. An alternative approach involves leveraging only those weights and operations that contribute most significantly to the network's performance. Furthermore, a body of research has demonstrated that removing a small fraction of non-essential parameters can lead to improvements in test-set performance. The process of removing redundant parameters aids in discarding noisy weights and disrupts patterns that the network may have learned from training data but which do not generalize across the entire data distribution [3, 4].

2. Related works

Pruning of neural networks is an optimization technique aimed at identifying and removing parameters with minimal impact on the network's accuracy [5]. This process involves employing various criteria to assess parameter significance, enabling precise determination of which parameters can be excluded without noticeable degradation in the network's performance. The primary body of research on pruning is predominantly focused on convolutional neural networks

(CNNs), widely used for image classification tasks, which form the foundation for a broad range of computer vision applications.

Within this domain, several key strategies have been developed for pruning algorithms, alongside criteria that assist in identifying parameters whose removal exerts minimal impact on the overall performance of the model.

2.1. Iterative pruning

Iterative pruning methods are typically applied to pre-trained neural networks. In this approach, after selecting an appropriate criterion, a series of parameters is progressively removed from the model. The updated network is then fine-tuned over several epochs, ensuring its further refinement. This cycle is repeated until the performance of the pruned neural network declines below a predefined threshold relative to the original model, or until the desired level of parameter reduction is achieved.

2.1.1. Magnitude-based pruning

One of the most effective and widely adopted approaches to pruning is magnitude-based pruning [6]. The central idea of this criterion is the exclusion of weights from the neural network whose absolute values fall below a predefined positive threshold. For instance, convolution operations can be represented as:

$$y_i = \sum_{j=0}^n w_{ij}x_j, \quad i = \overline{0, m},$$

indicating that weights with the smallest absolute values contribute the least to the resulting sum. Consequently, their removal is expected to have a minimal impact on the network's overall performance. Due to its simplicity, magnitude-based pruning is frequently employed in practice; however, its effectiveness diminishes when the neural network is heavily pruned.

2.1.2. Pruning based on the L_1 norm

This pruning method extends to the removal of entire filters in the convolutional layers of a neural network, based on their L_1 norm [7, 8] being below a specified threshold. The L_1 norm of a tensor $F \in \mathbb{R}^n$ is computed during each iteration of the pruning process as follows:

$$\|F\| = \sum_{k=1}^n |w_k|.$$

2.1.3. Pruning based on Euclidean distance

Another variant of this method involves removing parameters or filter tensors in convolutional layers based on their pairwise Euclidean distances [8–10] being less than a defined threshold. The Euclidean distance between any two parameter tensors $P, Q \in \mathbb{R}^n$ is calculated at each iteration of the pruning process as:

$$d(P, Q) = \sqrt{\sum_{k=1}^n (w_k^{(p)} - w_k^{(q)})^2}.$$

This computation yields a symmetric distance matrix $D \in \mathbb{R}^{I \times I}$:

$$D = \begin{cases} d(F_i, F_j), & i \neq j \\ 0, & i = j \end{cases},$$

where I – number of parameter tensors in the model, with F_i denoting the tensor at index i .

Subsequently, based on the computed distance matrix, one tensor from each pair with a distance below the threshold is removed from the model. This iterative process ensures reduction of redundant parameters while preserving the network's core functionality.

2.2. Pruning models during training

There exist algorithms that enable pruning of a neural network immediately after parameter initialization, allowing for the exclusion of redundant parameters directly during the training process. This approach relies on predefined criteria to assess parameter significance, enabling the network structure to be dynamically adapted throughout training [11, 12].

One of the primary advantages of this method is the significant time savings in creating a lightweight version of the neural network. Since the stages of training and removal of insignificant parameters occur simultaneously, the model can achieve an optimal state more rapidly, which is particularly critical in scenarios with limited time or resources.

Additionally, this approach enhances the model's generalization capability, as the removal of non-essential parameters during training can reduce the risk of overfitting. In this way, the model more effectively identifies the most important characteristics in the data, which potentially improves the quality of the final results.

2.2.1. Sparse evolutionary training

The sparse evolutionary training (SET [13]) method is based on evolutionary algorithms. The application of pruning during the network design phase, even before the start of training, significantly reduces the number of parameters in the model. This, in turn, lowers memory requirements and improves computational efficiency. Research has shown that neural networks constructed using sparse (or sparsely connected) layers exhibit excellent performance and quality comparable to traditional fully connected layers. Such layers, trained via the SET method, can replace fully connected ones without a loss in accuracy and can even feature a quadratic reduction in parameter count at the design stage. This approach involves a regular process of removing the smallest positive and largest negative weights from sparse connections. These weights are closest to zero, indicating they have minimal impact on the network's performance. After removing these weights, an equivalent number of new random connections are added to the current layer, maintaining a balanced number of connections during training. The cycle of weight removal and addition is repeated over a set number of training epochs. After training concludes, the layers are fixed at the final step of weight removal without adding new connections.

2.2.2. Trainability preserving pruning

Numerous studies emphasize the importance of preserving the trainability of neural networks, which can be disrupted by the pruning process. The trainability preserving pruning (TPP [14]) method focuses on minimizing correlations between convolutional layer filters to maintain their trainability post-removal. To achieve this, a regularization term is introduced into the loss function, penalizing filter non-orthogonality.

The authors of this method propose a more flexible constraint that does not demand perfect preservation of trainability but ensures that gradients can effectively propagate through the network without obstructions. Unlike stringent requirements for orthogonality, which mandate that singular values of the Jacobian matrix equal exactly one, TPP only requires that these values avoid extreme deviations, enabling stable training.

A critical aspect of this method is the separation of retained filters from removed ones in the target Gram matrix associated with the convolutional layer filters. All elements corresponding to removed filters are zeroed out, preserving the remaining values. Filters are pruned based on their L_1 norms, with the lowest values considered non-essential. Furthermore, it is crucial to

account for batch normalization layers, as their states are also subject to training. Removing filters can alter the internal distribution of features, necessitating appropriate modifications to the statistics collected during training. To mitigate this effect, the authors suggest ordering the lists of pruned filters and introducing a penalty term for batch normalization layers, thus providing necessary regularization for these layers.

2.2.3. Adding before pruning

The adding before pruning (ABP [15]) method is an innovative approach that allows the model to focus on significant filters based on training rather than artificial criteria such as norms, ranks, or other metrics.

In this method, an attention level is introduced to improve the evaluation of filter significance. At this level, binary significance scores are assigned, helping the system more clearly determine the importance of each filter during training. Furthermore, to optimize gradient propagation at the attention level, a specialized gradient estimator was developed. This tool has proven effective in ensuring convergence of computational flows through rigorous mathematical validation, enabling more efficient training management.

One of the key features of the ABP method is simultaneous pruning and training of filters, which facilitates the automatic elimination of redundancy in the neural network. This is achieved by reducing reliance on complex prior knowledge typically required to establish threshold criteria. Simplifying this process increases the model's flexibility and adaptability to various tasks.

Experimental results obtained during testing on standard image classification datasets show that the proposed ABP method significantly outperforms previous pruning algorithms in efficiency and accuracy. These achievements make the method particularly promising for modern machine learning and computer vision systems, where both resource efficiency and high performance are critical.

2.2.4. Graph convolutional network pruning

Graph convolutional network-based pruning (GCNP [16]) is designed to improve model compression, adapting networks for resource-constrained environments without compromising classification accuracy. The primary idea involves leveraging graph structures to make more effective decisions about which channels to retain or remove. This process consists of several key stages, from feature extraction at each layer to training a reinforcement learning-based agent.

The first step involves extracting channel features for each layer of the neural network. This information is then used to construct a graph, where the graph nodes represent channels, and edges between nodes are established based on feature similarity. The Kullback–Leibler divergence is used to evaluate this similarity, and edges are formed if the distance between channels falls below a predefined threshold.

Next, a GCN-based agent analyzes the constructed graph and synthesizes information on the importance of various channels. The agent decides which channels to prune, generating a probabilistic distribution of actions. To enhance adaptability and convergence to optimal pruning schemes, the agent is trained using the Policy Gradient method. This method enables the agent to optimize its actions based on the model's post-pruning performance.

The agent's training process is iterative. For each layer, the network generates multiple pruning schemes, selecting those that minimize the target objective as training labels. Parameter updates for the GCN are performed using backpropagation, gradually improving decision quality.

3. Dynamic pruning by importance of random excluded channels

Previously we proposed a novel neural networks pruning method, named DPIREC (dynamic pruning by importance of random excluded channels) [21]. The proposed pruning approach

builds on the concept of random subset feature selection (RSFS [20]), where the classification task is constructed as an ensemble of classifiers, each utilizing random subsets of features from the dataset. Each feature is assigned a relevance score determined by the accuracy of the classifiers that use it for prediction. Pruning in this algorithm is performed dynamically during the neural network's training process.

Let F denote the set of neural network parameters (e.g., weights, filters, or channels). Each parameter $f_j \in F$ is associated with a relevance value $r_j \in \mathbb{R}$. Similarly, the set of dummy parameters Z is defined, where each dummy parameter z_j is assigned a relevance value $q_j \in \mathbb{R}$.

During each i -th iteration of the neural network training, the following steps are executed:

1. Randomly select n parameters from F using a uniform distribution to form the subset S_i .
2. Similarly, randomly select m dummy parameters from Z using a uniform distribution to form the subset Y_i .
3. Compute the loss function value L_i for the neural network, where all parameters outside S_i are excluded. Define $l_i = -L_i$.
4. The relevance values r_j for parameters $f_j \in S_i \subset F$ are updated as $r_j = r_j + l_i - E(l)$, where $E(l)$ is the expected value of the loss function (average value across previous iterations).
5. Similarly, the relevance values q_i of the dummy parameters $z_j \in Y_i \subset Z$ are updated: $q_j = q_j + l_i - E(l)$.

The weights of the neural network are updated according to standard training algorithms, involving all trainable parameters. Dummy parameters do not participate in the training process or loss computation. However, their relevance values q_j are accumulated in the same manner as for true parameters. Consequently, the accumulation of relevance values q_j for a dummy parameter z_j becomes a random process unrelated to the actual performance of the neural network.

To determine which parameters f_j significantly influence the quality of neural network training, we establish a baseline level $q_{baseline}$ using the relevance of dummy parameters q_j . True parameters are deemed significant if their relevance exceeds this baseline level, as measured by their contribution to minimizing the neural network's loss function.

Finally, to identify the subset $S \subset F$ of parameters that really surpass the baseline relevance of dummy parameters, a statistical test is performed. Specifically, the relevance r_j of a parameter f_j must satisfy:

$$p(r_j > q_{baseline}) \geq \delta, \quad \forall f_j \in S \subset F, \quad (1)$$

where δ is a fixed probability threshold. The random baseline level $q_{baseline}$ is modeled as a normal distribution of dummy relevance values $\bar{q}_j$. The probability that a parameter is more relevant than a dummy parameter is then derived from the normal distribution:

$$p(r_j > q_{baseline}) = \frac{1}{\sigma\sqrt{2\pi}} \int_0^{r_j} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) dx, \quad (2)$$

where μ and σ are the mean and standard deviation of q_j across all dummy parameters.

The method for evaluating relevance represents one of the key elements of the DPIREC approach. Therefore, in this work, we conducted an analysis of the existing method and introduced a new algorithm that accounts for changes in the loss function during the neural network training process. The advantage of the new approach was demonstrated through extensive comparative testing of DPIREC against several existing pruning methods on image recognition tasks.

4. Reflected losses approximation

One of the primary challenges associated with the DPIREC method is its inability to effectively respond to reductions in the values of the loss function during neural network training. To overcome this limitation, we propose an innovative approach based on reflected losses approximation (RLA), in which each parameter f_j from the set F is linked not to a single relevance value but to a dynamic collection of reflected loss function values $r_j = \{r_{jk} \mid -\infty < r_{jk} \leq 0\}$. Similarly, the dummy z_j parameter of the set Z will also correspond to the set of reflected loss function values $q_j = \{q_{jk} \mid -\infty < q_{jk} \leq 0\}$.

During each iteration i of the neural network training process, the following sequence of operations is executed:

1. A subset S_i of n parameters is randomly sampled from the set F using a uniform distribution.
2. Similarly, a subset Y_i of m dummy parameters is randomly selected from the set Z under the same uniform distribution.
3. The value of the loss function L_i is evaluated for the neural network, retaining only the parameters in S_i while discarding all others. Define $l_i = -L_i$.
4. The computed value l_i is added into the set of relevance values r_j corresponding to each parameter $f_j \in S \subset F$.
5. The value l_i is added to the set of relevance values q_j associated with each dummy parameter $z_j \in Y_i \subset Z$.
6. To derive a numerical measure of parameter importance, polynomial fitting is applied to the set of loss values r_j , in this approach first degree polynomials are used. The coefficient corresponding to the highest-degree term of the polynomial is used to determine the relevance $\bar{r}_j$. A similar procedure is applied to compute the relevance $\bar{q}_j$ for dummy parameters.

Following the computation of the relevance metrics $\bar{r}_j$ for true parameters and $\bar{q}_j$ for dummy parameters, a statistical evaluation analogous to the baseline approach is conducted. The decision rule defined by equations (1) and (2) is applied, with $\bar{r}_j$ and $\bar{q}_j$ substituted as the respective relevance measures. This adaptation ensures that the statistical framework remains consistent with the baseline methodology while incorporating the refined relevance metrics $\bar{r}_j$ and $\bar{q}_j$.

4.1. RLA approach advantages demonstration

Consider an example that demonstrates the advantages of the DPIREC approach based on reflected losses approximation (RLA) compared to DPIREC approach based on an iterative change in relevance:

- We define the size of the set $|F| = 3$ and `iterations_count` = 30.
- Then, we execute steps 1, 3 and 4 of the algorithm iteratively for `iterations_count` times.
- For the obtained sets r_1, r_2, r_3 , polynomial fitting is performed.
- The parameter relevance values f_1, f_2, f_3 will be represented by the coefficients corresponding to the highest-degree term of the polynomial – $\bar{r}_1, \bar{r}_2, \bar{r}_3$.

As can be seen from the table and graph, parameter r_1 contributes the least to the changes in the loss function and thus has the lowest relevance value, increasing the likelihood of its exclusion from the neural network model. However, the approach based on an iterative change in relevance determines parameter r_3 as the least significant for the model, which does not align with its impact on the loss function. Excluding r_3 would result in greater accuracy losses in the model compared to excluding r_1 .

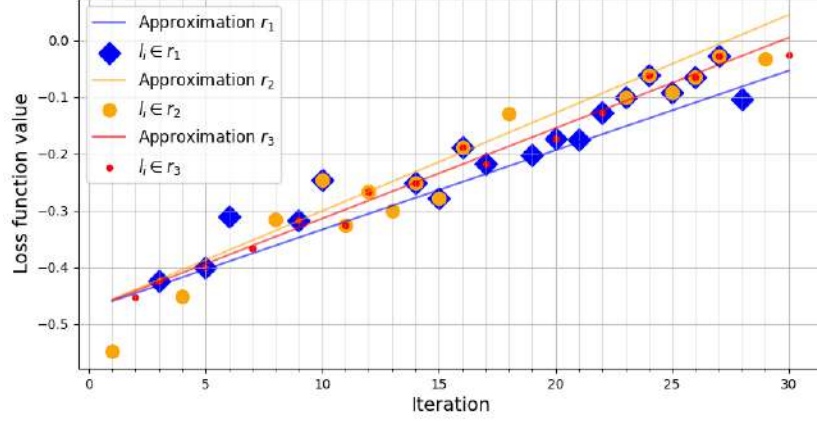


Figure 1. The points on the graph below correspond to the loss function values at each iteration. If a point was included in the set r_j of parameter f_j , it will be highlighted with the respective color associated with the parameter. The lines represent the approximation polynomials. The lines are offset to one point to emphasize the differences in the values $\bar{r}_1, \bar{r}_2, \bar{r}_3$

Table 1. Parameter – the parameter for which the relevance values were calculated, Base – the relevance value obtained using the iterative DPIREC approach, RLA – the relevance value obtained using the DPIREC RLA approach

Parameter	Base	RLA
f_1	0.786	0.014
f_2	0.844	0.017
f_3	0.628	0.016

As illustrated, the DPIREC RLA method more accurately reflects the true contribution of each parameter to the loss function, thereby facilitating more effective pruning decisions. This contrasts with the iterative DPIREC approach, which might misidentify less impactful parameters, leading to suboptimal pruning outcomes.

5. Experiments and results

5.1. Datasets

The CIFAR [17] datasets are widely used benchmarks in computer vision tasks. They are divided into two parts: CIFAR-10 and CIFAR-100, containing 10 and 100 classes, respectively. Each dataset consists of 50,000 training images and 10,000 test images with a resolution of 32×32 pixels. All images are uniformly distributed among the classes, ensuring balanced data.

During model training, data augmentation methods were applied, including random horizontal flipping and padding with cropping. For both CIFAR-10 and CIFAR-100, the padding size was set to 4 pixels. These strategies help reduce the risk of overfitting. This study does not require additional sophisticated data augmentation methods, as the primary focus is on model compression.

5.2. Network architectures

Two widely recognized convolutional neural network architectures, VGG [18] and ResNet [19], were selected for the experiments, as they serve as reliable benchmarks for evaluating model compression techniques, including pruning and other optimization methods.

The following architecture variants were used in the experiments:

- VGG-16: A classical architecture with 14.73 million parameters;
- ResNet-18: A residual block-based model containing 11.17 million parameters;
- ResNet-20: A compact model with 0.27 million parameters;
- ResNet-56: A mid-sized model with 0.86 million parameters.

These architectures were chosen due to their differences in size and complexity, allowing for an in-depth investigation of how parameters pruning affects performance.

To account for the differences in dataset scale, the architectures were adapted to minimize parameter redundancy. This ensures a balance between model performance and computational efficiency. Such adaptations facilitate a more meaningful comparison of models with varying parameter counts and structural complexities, providing deeper insights into the effects of pruning methods, such as sparsity, on classification accuracy.

5.3. Performance assessment

To investigate the effectiveness of the DPIREC RLA approach for pruning convolutional neural networks, it is necessary to conduct an experimental comparison with existing convolutional neural network pruning methods. The final accuracy of the neural network model at an equivalent level of sparsity will be used as the criterion for comparing the pruning methods. Below, we present the results of an experimental comparison of the DPIREC approach with pruning methods based on the L1 norm and Euclidean distance, which were also implemented.

Table 2. Comparison of pruning quality for various neural networks on the CIFAR-10 dataset. Network – convolutional neural network model; Method: (1) – pruning based on the L_1 norm, (2) – pruning based on Euclidean distance, (3) – iterative DPIREC based on an iterative change in relevance, (4) – DPIREC RLA; Baseline Acc. – accuracy of the model on a test sample without pruning; Pruned parameters – number of parameters excluded from the model 20%, 40%, 60% correspondingly; Acc.(%) and $\downarrow$ (%) – accuracy and decrease of accuracy pruned neural network respectively

Network	Method	Baseline Acc.(%)	Pruned params					
			20%		40%		60%	
			Acc.(%)	$\downarrow$ (%)	Acc.(%)	$\downarrow$ (%)	Acc.(%)	$\downarrow$ (%)
VGG16	(1)	92.3	91.82	−0.48	91.18	−1.12	89.64	−2.66
	(2)		91.87	−0.43	91.26	−1.04	89.84	−2.46
	(3)		92.03	−0.27	91.31	−0.99	90.27	−2.03
	(4)		92.16	−0.14	91.38	−0.92	90.38	−1.92
ResNet18	(1)	93.7	93.53	−0.17	93.28	−0.42	92.43	−1.27
	(2)		93.56	−0.14	93.29	−0.41	92.55	−1.15
	(3)		93.58	−0.12	93.40	−0.31	92.92	−0.78
	(4)		93.61	−0.09	93.43	−0.27	92.96	−0.74
ResNet20	(1)	92.1	91.76	−0.34	91.3	−0.8	89.74	−2.36
	(2)		91.84	−0.26	91.32	−0.78	89.95	−2.15
	(3)		91.92	−0.18	91.44	−0.66	90.24	−1.86
	(4)		91.98	−0.12	91.51	−0.59	90.29	−1.81
ResNet56	(1)	93.2	92.91	−0.29	92.62	−0.58	91.85	−1.35
	(2)		92.99	−0.21	92.65	−0.55	91.95	−1.25
	(3)		93.05	−0.15	92.89	−0.31	92.36	−0.84
	(4)		93.10	−0.10	92.93	−0.27	92.39	−0.81

Table 3. Comparison of pruning quality for various neural networks on the CIFAR-100 dataset

Network	Method	Baseline Acc.(%)	Pruned params					
			20%		40%		60%	
			Acc.(%)	↓(%)	Acc.(%)	↓(%)	Acc.(%)	↓(%)
VGG16	(1)	69.5	68.71	−0.79	66.93	−2.57	64.39	−5.11
	(2)		68.75	−0.75	66.98	−2.52	64.62	−4.88
	(3)		68.90	−0.61	67.63	−1.87	65.66	−3.84
	(4)		68.96	−0.54	67.76	−1.74	65.86	−3.64
ResNet18	(1)	74.32	73.94	−0.38	73.04	−1.28	71.86	−2.46
	(2)		73.99	−0.33	73.11	−1.21	71.94	−2.38
	(3)		74.08	−0.24	73.34	−0.98	72.33	−1.99
	(4)		74.16	−0.16	73.43	−0.89	72.65	−1.67
ResNet20	(1)	68.23	67.75	−0.48	66.64	−1.59	65.12	−3.11
	(2)		67.81	−0.42	66.73	−1.50	65.24	−2.99
	(3)		67.91	−0.32	66.89	−1.34	65.75	−2.48
	(4)		68.01	−0.22	67.29	−0.94	66.12	−2.11
ResNet56	(1)	72.46	72.06	−0.40	71.12	−1.34	69.81	−2.65
	(2)		72.12	−0.34	71.19	−1.27	69.91	−2.55
	(3)		72.19	−0.27	71.57	−0.89	70.34	−2.12
	(4)		72.27	−0.19	71.83	−0.63	70.66	−1.80

The DPIREC method based on reflected losses approximation (RLA) demonstrates superior performance among all reviewed methods.

- Across all architectures (VGG16, ResNet18, ResNet20, ResNet56) and datasets (CIFAR-10, CIFAR-100), DPIREC RLA ((4) in the table) achieves the lowest accuracy decrease at the same parameter pruning levels (20%, 40%, 60%).
- The advantage of DPIREC RLA becomes particularly evident at higher levels of parameter pruning. For instance, in the case of ResNet56 on CIFAR-10, with 60% pruning, the accuracy decrease is only 0.81% for DPIREC RLA, compared to 1.35% for the L1-norm-based pruning method ((1) in the table).

Accuracy decrease reduces with larger ResNet architectures

- Larger models, such as ResNet56, demonstrate less quality degradation under DPIREC RLA compared to more compact architectures like ResNet20.
- For example, with 60% pruning on CIFAR-10, the accuracy decrease for ResNet20 is 1.81%, whereas for ResNet56 it is only 0.81%. This highlights the capability of larger models to retain performance even under significant parameter pruning.

DPIREC RLA effectively considers parameters relevance

- The use of reflected losses approximation (RLA) approach enables a more precise evaluation of parameter contributions to loss reduction, resulting in more balanced and effective pruning.
- This is particularly evident when comparing DPIREC based on an iterative change in relevance and DPIREC RLA – the latter consistently outperforms the former.

5.4. Comparison with other state-of-the-art solutions

This section provides a comparative analysis of the DPIREC RLA method with modern state-of-the-art approaches to neural network pruning during training, such as adding before pruning (ABP [15]) and graph convolutional network-based pruning (GCNP [16]). This comparison is essential for an objective evaluation of the effectiveness of the developed method within the context of the current state of pruning technology.

Table 4. Network – convolutional neural network model; Method – pruning method; Baseline Acc. – accuracy of the model on a test sample; Pruned Acc. – accuracy of the pruned model on a test sample; Acc. $\downarrow$ – accuracy decrease after pruning; Params $\downarrow$ – number of excluded parameters

Dataset	Network	Method	Baseline Acc.(%)	Pruned Acc.(%)	Acc. $\downarrow$ (%)	Params $\downarrow$ (%)
CIFAR-10	ResNet20	ABP	92.15	91.03	−1.12	45.1
		DPIREC RLA	92.1	91.36	− 0.64	45
		GCNP	92.25	91.58	−0.67	38.51
		DPIREC RLA	92.1	91.57	− 0.59	40
	ResNet56	GCNP	92.25	92.22	−0.33	27.42
		DPIREC RLA	92.1	91.89	− 0.21	30
		ABP	93.41	93.1	−0.31	45.7
		DPIREC RLA	93.2	92.93	− 0.29	45
CIFAR-100	ResNet20	GCNP	68.38	68.95	0.57	8.16
		DPIREC RLA	68.23	68.16	−0.07	10
		GCNP	68.38	66.14	−2.24	49.42
		DPIREC RLA	68.23	66.57	− 1.66	50
	ResNet56	GCNP	72.86	72.22	−0.64	39.83
		DPIREC RLA	72.46	71.83	− 0.63	40

The DPIREC RLA method demonstrates competitive performance relative to other state-of-the-art techniques. In most cases, DPIREC RLA achieves lower accuracy degradation (Acc. $\downarrow$) at comparable or better parameter pruning levels compared to the ABP and GCNP methods. For instance, with ResNet20 on CIFAR-10 and approximately 45% sparsity, DPIREC RLA reduces accuracy by just 0.64%, outperforming ABP (−1.12%) and GCNP (−0.67%).

The method performs equally well on both smaller models (e.g., ResNet20) and larger architectures (e.g., ResNet56), consistently achieving either minimal accuracy losses or competitive results at similar levels of sparsity.

6. Conclusion

We conducted a comprehensive investigation and enhancement of the previously proposed neural network pruning method DPIREC (Dynamic Pruning by Importance of Random Excluded Channels). The primary focus was on optimizing the new parameter exclusion approach and improving the overall efficiency of the method.

The following key results were achieved during the course of this research:

- A detailed analysis of the existing parameter significance evaluation criterion in the DPIREC method was carried out.

- A novel approach for evaluating parameter relevance based on polynomial approximation was proposed, enabling a more accurate consideration of the dynamics of model accuracy changes.
- Extensive experimental studies were performed on various convolutional neural network architectures (VGG, ResNet) using the CIFAR-10 and CIFAR-100 datasets.
- Results demonstrated the superiority of the improved DPIREC method over several existing pruning techniques. Specifically, when pruning 40% of the ResNet18 architecture on the CIFAR-100 dataset, the drop in accuracy was only 0.89%.

The obtained results confirm the effectiveness of the proposed improvements and suggest that this method can be recommended for optimizing neural network architectures. In future work, we plan to adapt the method for larger-scale architectures and explore its applicability in other areas of deep learning.

References

1. Gusak J., Cherniuk D., Shilova A. et al. Survey on Efficient Training of Large Neural Networks // Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, July 23-29, 2022. DOI: 10.24963/ijcai.2022/765.
2. Wang Y., Wei G., Brooks D. Benchmarking TPU, GPU, and CPU Platforms for Deep Learning // John A. Paulson School of Engineering and Applied Sciences, Harvard University, July 24, 2019. DOI: 0.48550/arXiv.1907.10701.
3. Li Z., Chen T., Wang Z. et al. Can pruning improve certified robustness of neural networks? // Transactions on Machine Learning Research. 2022. DOI: 10.48550/arXiv.2206.07311.
4. Harutyunyan H., Reing K., Steeg G.V. et al. Improving Generalization by Controlling Label-Noise Information in Neural Network Weights // Proceedings of the 37th International Conference on Machine Learning, ICML'20, July 13, 2020. Article 381. P. 4071–4081. DOI: 10.5555/3524938.3525319.
5. Cheng H., Zhang M., Shi J.Q. A Survey on Deep Neural Network Pruning-Taxonomy, Comparison, Analysis, and Recommendations // IEEE Trans. Pattern Anal. Mach. Intell. 2024. Vol. 46, no. 12. P. 10558–10578. DOI: 10.1109/TPAMI.2024.3447085.
6. Li H., Kadav A., Durdanovic I. et al. Pruning Filters for Efficient ConvNets // 5th International Conference on Learning Representations, ICLR 2017, April 24-26, 2017, Toulon, France. DOI: 10.48550/arXiv.1608.08710.
7. He Y., Lin J., Liu Z. et al. AMC: AutoML for Model Compression and Acceleration on Mobile Devices // Computer Vision – ECCV 2018. Vol. 11211. Springer, Cham, 2018. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-01234-2_48.
8. Lee J., Park S., Mo S. et al. Layer-adaptive Sparsity for the Magnitude-based Pruning // International Conference on Learning Representations, Vienna, Austria, May 4, 2021. DOI: 10.48550/arXiv.2010.07611.
9. He Y., Zhang X., Sun J. Channel Pruning for Accelerating Very Deep Neural Networks // 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, October 22-29, 2017. DOI: 10.1109/ICCV.2017.155.

10. Louizos C., Welling M., Kingma D.P. Learning Sparse Neural Networks through L_0 Regularization // International Conference on Learning Representations (ICLR), 2018. DOI: 10.48550/arXiv.1712.01312.
11. Zhu M., Gupta S. To prune, or not to prune: exploring the efficacy of pruning for model compression // International Conference on Learning Representations (ICLR), October 5, 2017. DOI: 10.48550/arXiv.1710.01878.
12. Frankle J., Carbin M. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks // International Conference on Learning Representations (ICLR), March 4, 2019. DOI: 10.48550/arXiv.1803.03635.
13. Mocanu D.C., Mocanu E., Stone P. et al. Scalable Training of Artificial Neural Networks with Adaptive Sparse Connectivity inspired by Network Science // *Nature Communications*. 2018. Vol. 9. Article 2383. DOI: 10.1038/s41467-018-04316-3.
14. Wang H., Fu Y. Trainability Preserving Neural Pruning // International Conference on Learning Representations (ICLR), July 25, 2022. DOI: 10.48550/arXiv.2207.12534.
15. Tian G., Sun Y., Liu Y. et al. Adding Before Pruning: Sparse Filter Fusion for Deep Convolutional Neural Networks via Auxiliary Attention // *IEEE Transactions on Neural Networks and Learning Systems*. 2021. Vol. 36, no. 3. P. 3930–3942. DOI: 10.1109/TNNLS.2021.3106917.
16. Yang C., Liu H. Channel pruning based on convolutional neural network sensitivity // Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, Nanjing, China, October 1, 2022. Vol. 507. P. 97–106. DOI: 10.24963/ijcai.2022/431.
17. Krizhevsky A., Hinton G. Learning Multiple Layers of Features from Tiny Images // Technical Report, Toronto, Ontario, April 8, 2009. URL: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>
18. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition // International Conference on Learning Representations, September 4, 2014. DOI: 10.48550/arXiv.1409.1556.
19. He K., Zhang X., Ren S. et al. Deep Residual Learning for Image Recognition // IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 27–30, 2016, P. 770–778. DOI: 10.1109/CVPR.2016.90.
20. Pohjalainen J., Räsänen O., Kadioglu S. Feature selection methods and their combinations in high-dimensional classification of speaker likability, intelligibility and personality traits // *Computer Speech & Language*. 2015. Vol. 29. P. 145–171. DOI: 10.1016/2013.11.004.
21. Novikov D.A., Buryak D.Y. Neural network pruning method during training process // Russian Supercomputing Days, September 25–26, 2023, Moscow, Russia. P. 53–63. DOI: 10.29003/m3478.978-5-317-07070-0.

Efficiency of Machine Learning Tasks on HPC Devices

G. Promyslov¹, A. Efremov¹, Y. Ilyasov¹,
V. Pisarev^{1,2}[0000-0002-9004-1839], and A. Timofeev^{2,1}[0000-0003-1156-893X]

¹ HSE University, Moscow, Russia

² Joint Institute for High Temperatures of RAS, Moscow, Russia

Abstract. Accurate benchmarking is critical for selecting computing architectures optimized for machine learning (ML) tasks. Conventional benchmarks such as High-Performance Linpack (HPL) and High Performance Conjugate Gradients (HPCG) often fail to capture the diversity and complexity of modern ML workloads. This study investigates the correlation between hardware parameters (e.g., processor architecture, cache size, frequency) and ML performance across various devices, including CPUs and accelerators. By studying a wide range of ML tasks, we identify key performance bottlenecks and explore whether a correlation-based approach can guide the selection of optimal hardware for an entire class of ML tasks. Our findings offer practical recommendations for future computing architectures and advance the efficient use of high-performance computing for diverse ML applications.

Keywords: HPC · Machine learning · Benchmarking · Performance Profiling · Hardware Efficiency

1 Introduction

Selecting an optimal computing architecture plays a pivotal role in attaining highest performance computationally-intensive tasks. Traditional benchmarks, such as High-Performance Linpack (HPL) and High Performance Conjugate Gradients (HPCG), have long been standards for evaluating supercomputer performance [7,6]. However, because they primarily focus on solving systems of linear equations, they do not fully representative for other types of workloads. In high-performance computing (HPC), this shortfall becomes particularly critical, as resource utilization in supercomputing environments must be maximized [8,22]. A particular domain that will be explored in this study is the ML workloads, whose unique characteristics include large-scale matrix operations, neural network optimization, and intensive data exchange.

Modern ML applications have become increasingly prevalent across diverse domains, ranging from big-data analytics to computer vision and natural language processing [11]. In the last decade, ML and deep learning methods are gaining popularity in scientific-oriented domains, such as computation chemistry [12,10], molecular simulations [3,16,23], materials design and discovery [18].

The growing popularity of ML has been accompanied by a proliferation of hardware platforms designed to accelerate computations, including graphics processing units (GPUs), tensor processing units (TPUs), and specialized accelerators [15]. This variety of hardware solutions heightens the need for more precise, ML-oriented evaluation criteria and benchmark suites that can determine which devices offer the most efficient execution of specific tasks.

Despite the abundance of benchmarks and testing frameworks, an unresolved question remains: can the correlations between hardware characteristics and ML performance be leveraged to select optimal equipment for an entire class of ML tasks? In this work, we hypothesize that such correlation analysis will not only provide a more objective assessment of performance but also inform the design of future computing systems optimized for ML workloads.

The main objective of this study is to develop and validate a correlation-based benchmarking method for analyzing hardware characteristics when running various ML tasks and obtain recommendations for computer elements and architecture which are optimal for ML class tasks.

Potential applications of these findings include data center procurement recommendations, the architectural planning of supercomputers for specific ML workloads, and the optimization of cloud-based platform configurations.

The remainder of this paper is organized as follows. The next section describes the methodology and the set of ML tasks employed in the study. We then present our performance analysis results and discuss the identified correlations. The final section summarizes our conclusions and provides recommendations for designing future computing systems aimed at ML workloads.

2 Related work

High-Performance Computing (HPC) has traditionally focused on large-scale simulations and data-intensive workloads using multi-core Central Processing Units (CPUs) and Graphics Processing Units (GPUs). In recent years, a substantial body of research has examined how the characteristics of these hardware components correlate with the performance of Machine Learning (ML) tasks, especially in the context of deep neural networks and gradient-boosting algorithms [6]. Early investigations in this area concentrated on establishing how memory bandwidth, core count, and heterogeneous computing architectures influence throughput and model convergence times. For example, studies have demonstrated that increasing core counts can reduce training time for convolutional neural networks (CNNs), but diminishing returns often arise beyond a certain threshold due to memory contention [24]. These findings underscore the importance of balancing compute and memory resources—a concept formalized in Roofline models that capture performance limitations imposed by arithmetic intensity and memory bandwidth [24].

A variety of tools and benchmarks have been employed to assess the computational efficiency of ML tasks. Traditional HPC performance metrics such as floating-point operations per second (FLOPS) are now coupled with additional

ML-specific measures like accuracy, throughput, and timed epochs. Profilers (e.g., NVIDIA Nsight, Intel VTune) and platform-monitoring frameworks (e.g., PAPI, LIKWID) facilitate detailed performance tracing, allowing researchers to identify pipeline inefficiencies and bottlenecks [4]. Moreover, alternative evaluation methodologies such as Roofline analysis [24] and domain-specific benchmark suites (e.g., HPCG for high-performance conjugate gradient [6]) provide further insight into how architectural features and algorithmic structures intersect.

Benchmarking efforts specific to ML frameworks have grown substantially in recent years. TensorFlow [2], PyTorch [21], and XGBoost [5] are among the most widely used packages in both academic and industrial settings. Comparative studies have investigated scaling behaviors, GPU utilization, memory usage, and precision trade-offs (e.g., FP32 vs. FP16) to determine optimal configurations [2,21]. Much of this work relies on representative workloads such as image classification or language modeling for measuring throughput and latency. In parallel, community-driven benchmark suites like MLPerf [9,19] and AI Benchmark [14] emerged to provide standardized methodologies for evaluating end-to-end performance of ML pipelines. These suites incorporate tasks that stress different aspects of hardware and software stacks, such as dense and sparse computation, or recurrent and convolutional layers.

Despite these advances, existing studies often adopt narrow task definitions or focus on a limited subset of performance metrics. For instance, they may primarily target training throughput while overlooking other critical factors like performance variability across different hardware vendors or the interplay between HPC-style modeling tasks and intensive ML workloads. Furthermore, the emphasis on standard neural network architectures (ResNet, Transformer, etc.) in many benchmarking efforts can make it challenging to generalize findings to more specialized or emerging ML applications. Another frequently cited shortcoming is the lack of integrated analysis methods that combine hardware monitoring, energy consumption data, and application-level performance metrics—a combination essential for holistic evaluations in HPC environments.

In this paper, we address these deficiencies by proposing an elaborated benchmarking approach that jointly considers ML tasks alongside a set of physics-based kernels within a single HPC context. Unlike prior work, our methodology tracks both application-level metrics (e.g., iteration time, solution accuracy) and low-level hardware characteristics (e.g., memory throughput, instruction mix) to capture a more comprehensive view of resource utilization. We also extend existing performance analysis techniques by incorporating real-time profiling and cross-hardware comparisons (CPU vs. GPU vs. multi-GPU setups), providing insights into scaling efficiency for diverse workload categories. This dual focus on ML and physics-based tasks enables us to explore synergies and trade-offs not visible in ML-only or HPC-only studies. Consequently, our work contributes a novel perspective on how to optimize HPC infrastructures for both data-driven and simulation-driven applications simultaneously.

3 Methodology

As there is a large body of benchmarking data available for certain ML tasks, this study uses the hybrid methodology: original performance results are obtained for platforms available to the authors, and meta-analysis of the publicly available data is done for other platforms.

For evaluation, we explore several benchmark suites tailored for ML tasks, ai-benchmark [13], MLPerf [9,19], other test suites [1]. The methodology for individual benchmarks is presented in the next subsections.

3.1 Methodology for ai-benchmark

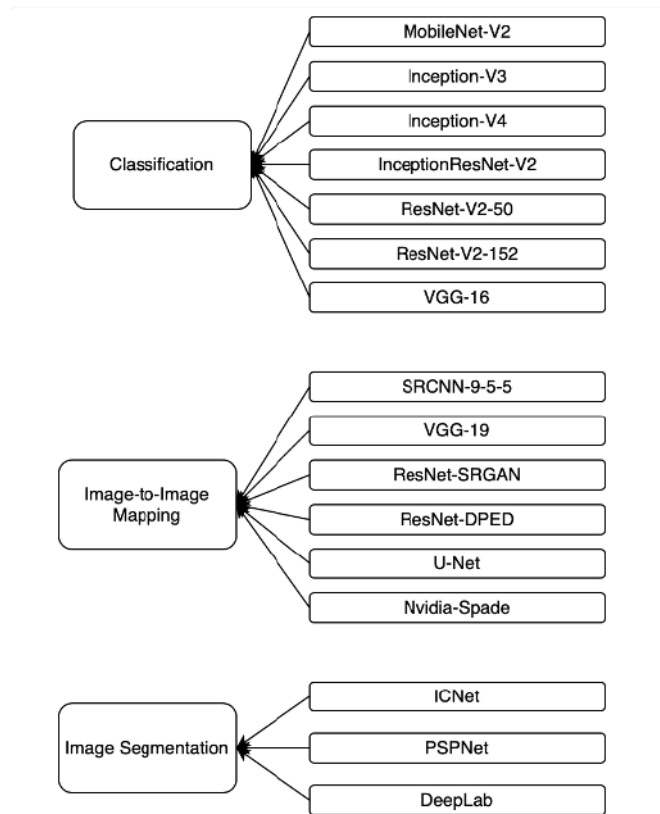


Fig.1: Deep learning models used in ai-benchmark, divided into three main classes of problems

ai-benchmark is a set of test problems chosen to characterize the average performance of a platform for classification and computer vision tasks. That

benchmark consists of 19 different neural network models that cover most deep learning architectures used in popular tasks and allows to test both their training and inference processes. The models used in *ai-benchmark* are shown in Fig. 1 and are divided into three main classes and several subclasses. The main classes of problem-solving models are: models for object classification, models for image-to-image mapping and models for image segmentation.

For massively-parallel systems, this benchmark can be used to evaluate single-node performance. Testing with an *ai-benchmark* performance evaluation tools was performed on nodes of the supercomputer cHARISMa. That system was chosen due to the need to design and establish simple and common methods for launching, testing, and profiling in random complex environment. In Fig. 2 characteristics of node of the supercomputer cHARISMa are shown that were used for testing. The detailed characteristics and performance of those components will be shown later in the article alongside other models.

10 Type B Computing nodes	
Processor Model	2 x Intel Xeon Gold 6152 2.1-3.7 HHZ (22 cores)
GPU Model	4 x NVIDIA Tesla V100 32 GB
RAM	1536 GB
Solid-state drive	2 x SSD 240 GB (RAID 1)
Network Adapter InfiniBand	2 x Mellanox 100Gb/s Infiniband Dual Port
Network Adapter Ethernet	Intel Ethernet 10G 4P X710/I350

Fig. 2: Supercomputer cHARISMa node Type B characteristics, on which the calculations for testing were performed

We design the method for launching the benchmark together with the procedure to test the performance of random system during benchmark launch and by using external software on the node configuration presented in Fig. 2.

The values of various metrics were also compared with their theoretical values over a given class of scientific problems to verify the assertion that belonging to a certain class of scientific problems will lead to similar performance results on different systems.

In this article we analyze only the metrics that directly affect the performance of the supercomputer components for the tasks of deep learning. They will be discussed later in details.

3.2 Methodology for MLPerf HPC Training

MLPerf HPC Training [9,19] is a specialized performance evaluation test designed to assess the computational capabilities of supercomputers in training

complex deep learning models. This standardized benchmarking suite is focused on high-performance computing and includes various scenarios that reflect key workloads in scientific and engineering research.

Scenarios of MLPerf HPC Training:

- CosmoFlow: Cosmological process modeling using convolutional neural networks. The primary computational workload involves processing large volumes of multidimensional data, requiring high memory bandwidth and significant computational resources for both CPUs and GPUs.
- DeepCAM: Satellite image analysis for climate research. This scenario employs convolutional neural networks for anomaly detection, imposing high demands on computational power and efficient utilization of parallel computing.
- OpenCatalyst: Chemical reaction modeling using density functional theory in combination with machine learning. The computations involve both traditional HPC algorithms and deep learning training, making this scenario computationally intensive.
- OpenFold: Protein structure prediction based on deep learning and transformer architectures. Introduced in MLPerf HPC Training v3.0, this scenario requires substantial computational resources, particularly from GPUs, due to intensive matrix multiplication operations.

The selection of these performance evaluation scenarios is justified by their scientific significance and high computational complexity. As MLPerf HPC Training has evolved, these scenarios have been adapted to the changing requirements of computing systems: CosmoFlow, DeepCAM, and OpenCatalyst have maintained continuity across versions, enabling performance trend analysis, while OpenFold has expanded the scope of tested workloads by incorporating more modern machine learning models.

MLPerf HPC Training was chosen as the primary evaluation tool because it is tailored for scientific computing and provides a comprehensive assessment of supercomputer performance in deep learning tasks. It is developed and maintained by MLCommons, ensuring its relevance and alignment with current HPC standards.

The performance analysis is based on data published on the official MLCommons website [20] and specifications of individual CPU and GPU models provided by manufacturers such as Intel, AMD, NVIDIA, and FUJITSU. This selection ensures data reliability, as the sources are authoritative and enable reproducibility of the analysis.

3.3 Methodology for other test suites

The data for this study was collected from <https://openbenchmarking.org>, a repository of Phoronix test suite benchmark results on various platforms. Specifically, we focused on machine learning (ML) test suites that provide performance metrics for different CPUs and GPUs. The benchmark results include metrics

such as inference or training time, which are critical for evaluating the performance of hardware in ML tasks.

Since benchmark results are expressed in different units of measurement, results were converted to either inverse seconds or units per second for the sake of consistency. Benchmark results data was also grouped or filtered based on the number of compute devices in the dataset or the benchmark scenarios. For CPUs following characteristics were considered for analysis — number of cores, base clock frequency, L3 cache size, memory bandwidth, peak FP32 performance. For GPUs — number of cores, clock frequency, memory bandwidth, peak FP32 performance.

3.4 Common part of methodologies

The following metrics were used to evaluate computational performance:

- System performance, expressed as the inverse execution time of a given task, independent of the contribution of CPUs or GPUs.
- Relative performance, representing the fraction of the inverse execution time that corresponds to the share of either CPU or GPU contribution to the overall system performance, proportional to the peak floating-point performance (FLOPS) of the respective device.
- Balance, calculated as the ratio of the peak performance sum (either for all CPUs or all GPUs) to the total memory bandwidth (main memory for CPUs or video memory for GPUs), ensuring a consistent comparison within the same type of computational resources.
- Average execution time, calculated as the arithmetic mean of execution times for a specific computational device characteristic.
- Potential peak Performance, computed as the product of the average execution time and the peak FLOPS performance of the given computing device.

Key hardware characteristics considered in the analysis:

- CPU: Number of cores, clock frequency, L3 cache size, memory bandwidth, peak FP32 performance.
- GPU: Number of cores, clock frequency, memory bandwidth, peak FP32 performance.

For data processing and analysis, aggregation methods were applied based on the logical grouping of datasets by benchmark version, scenario, computing device type and model. The software implementation of these principles accelerated the processing and facilitated a more detailed analysis and visualization of results. Data validation and cleaning were performed manually, with results corresponding to computing device models whose specifications could not be confirmed from reliable sources being excluded from the analysis.

4 Results

4.1 Results for ai-benchmark

The *ai-benchmark* score is calculated as geometric mean of the inverse runtime values of different models in the benchmark, so the higher the score is, the better model performed. For the convenience of the experiment, the averages among all presented models for which information was available are being calculated. Those averages are presented in Figures 3 and 5 by solid lines. We decided to divide results of CPU testing shown on the Figure 3 into two different classes depending of manufacturers: Intel or AMD. Tables with the data on the basis of which those graphs were constructed are presented later in the article.

Let us now directly show and discuss the results of CPU testing. Figure 3 represents the dependencies of score from chosen metrics: number of CPU cores, amount of threads, L3 Cache size (MB), theoretical performance (Flops) and recommended retail price (USD).

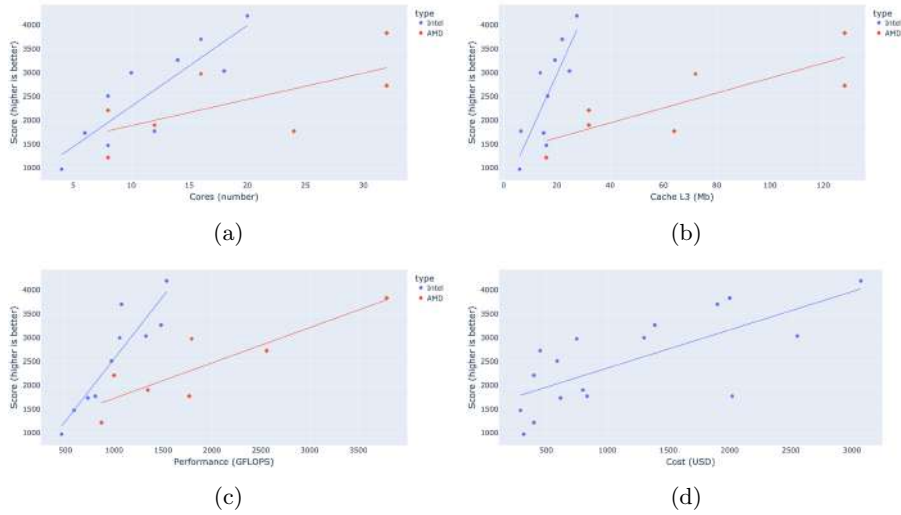


Fig. 3: Dependency of *ai-benchmark* score from the number of cores in CPU (a), the CPU L3 cache size (b), rated CPU FLOPS (c), CPU cost (d) (launch date prices were taken when possible). Red lines and symbols correspond to the AMD CPU models and blue lines and symbols to Intel CPU models

Figure 4 represents the existing dependency between L3Cache size and performance of different CPU models. That proves visible dependencies between figures shown at 3

Figure 5 represents the dependencies of score on GPU platforms from selected metrics: number of GPU cores, clock speed, memory bandwidth, peak perfor-

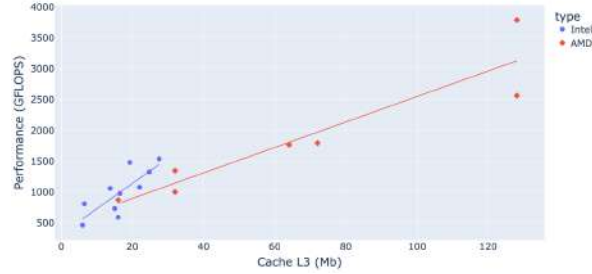


Fig. 4: Dependence of evaluated CPU performance from L3Cache size. Red line and symbols correspond to the AMD CPU models and blue line and symbols to Intel CPU models

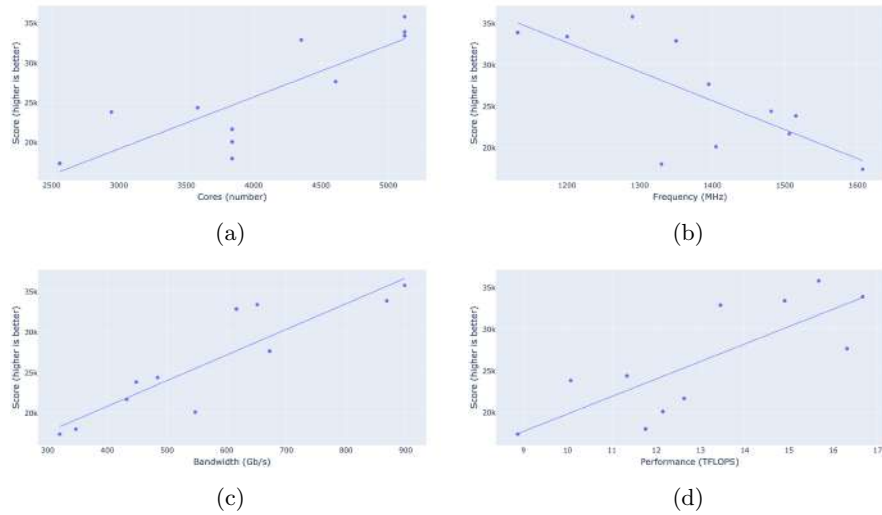


Fig. 5: Dependency of *ai-benchmark* score from the number of GPU cores (a), the GPU clock speed (b), GPU memory bandwidth (c), GPU peak performance in FP32 (d)

mance in FP32 FLOPS in graphics processing unit (GPU). From the figures we see that the primary characteristic determining the benchmark performance on CPU is the peak floating-point performance, suggesting that the problems chosen for the benchmark are compute-bound on the tested platforms. For GPU-accelerated platforms, the performance increases with the number of GPU cores, peak flops and memory bandwidth. However, those characteristics typically increase simultaneously with each new GPU generation, so that it's hard to decide what plays the most significant role.

To identify the bottleneck characteristic, we have conducted a roofline analysis for the benchmark running on a cHARISMa Type B node. Figure 6 shows the obtained result. We can see that the benchmark tasks (red-circled marker) lie within the compute-bound region, meaning that *ai-benchmark* tasks are compute-bound both on CPU-only and on CPU+GPU platforms. We should also note that the performance is significantly below the peak GPU performance, meaning that such tasks have a significant potential for improved performance by algorithmic changes.

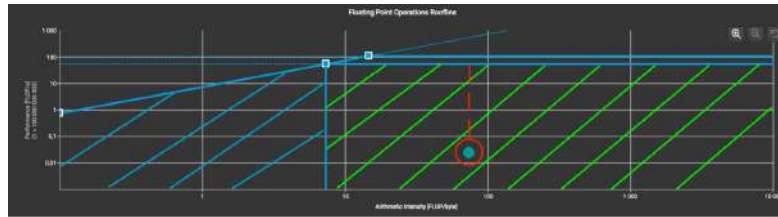
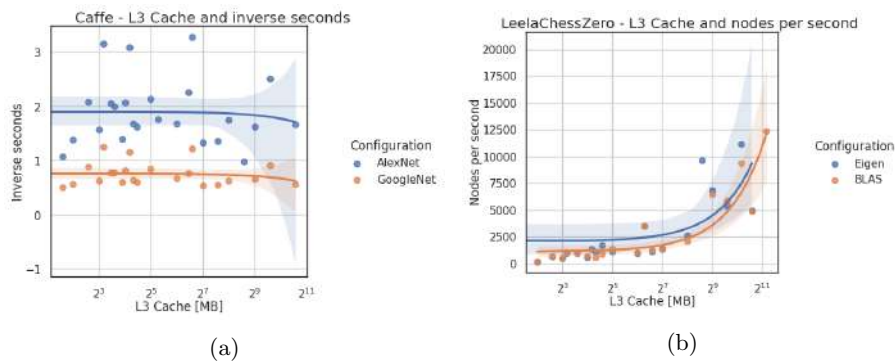


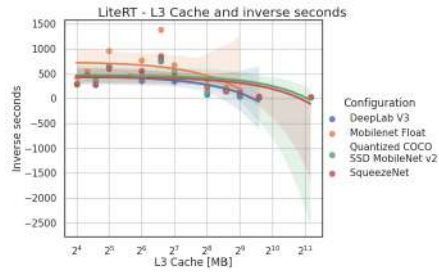
Fig. 6: Roofline analysis of *ai-benchmark* on cHARISMa Type B computing node. Blue shading denotes the bandwidth-bound zone, green shading denotes the compute-bound zone for NVidia V100 GPU. The red-circled marker is the position of *ai-benchmark*

4.2 Results for other CPU test suites

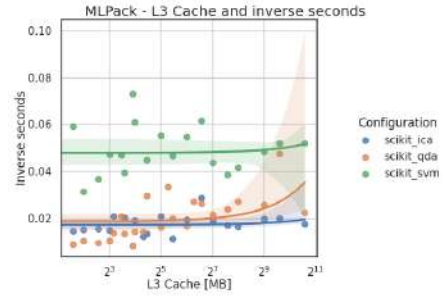
This subsection provides graphs showing the dependencies between CPU specifications and benchmark results within each test suite.

L3 Cache On the Figure 7 there are plots with results in benchmarks and cache sizes. In some packages, processors with larger L3-cache have better performance (e.g., LeelaChessZero, MLPack, Numenta and Whisper.cpp). However, it is difficult to distinguish the dependence between them.

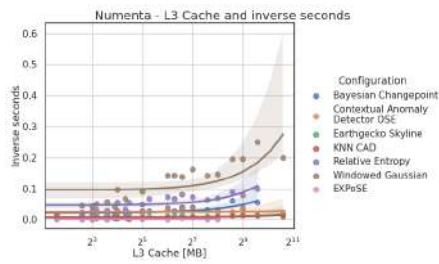




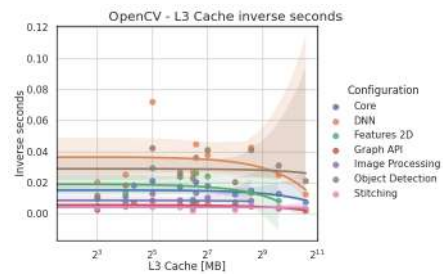
(c)



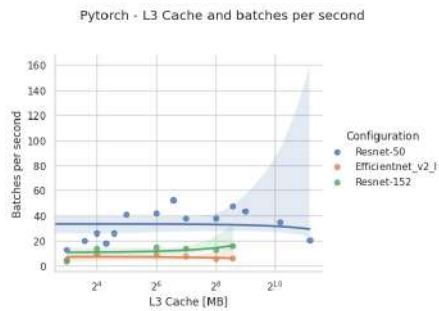
(d)



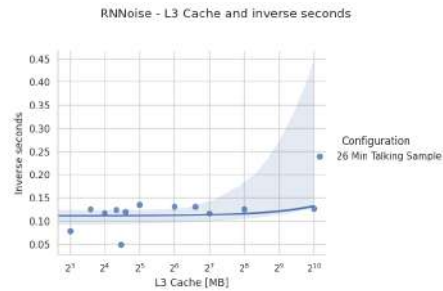
(e)



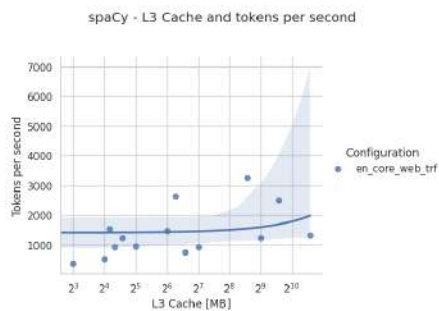
(f)



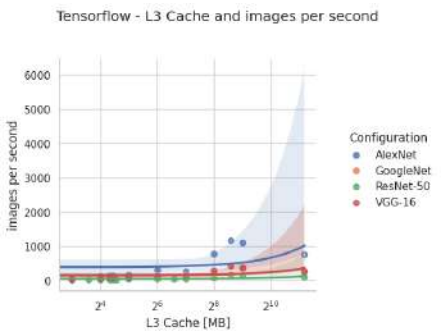
(g)



(h)



(i)



(j)

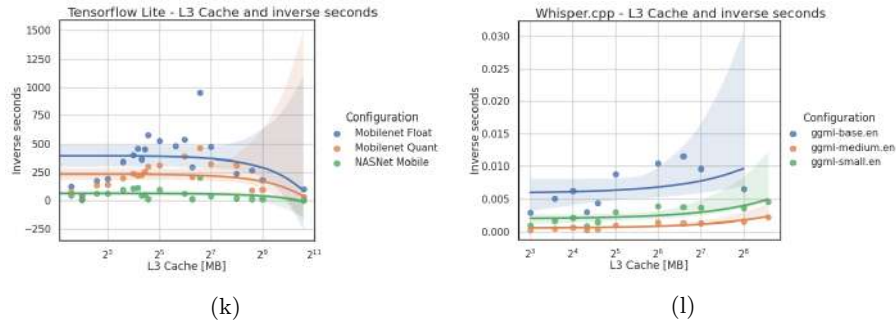
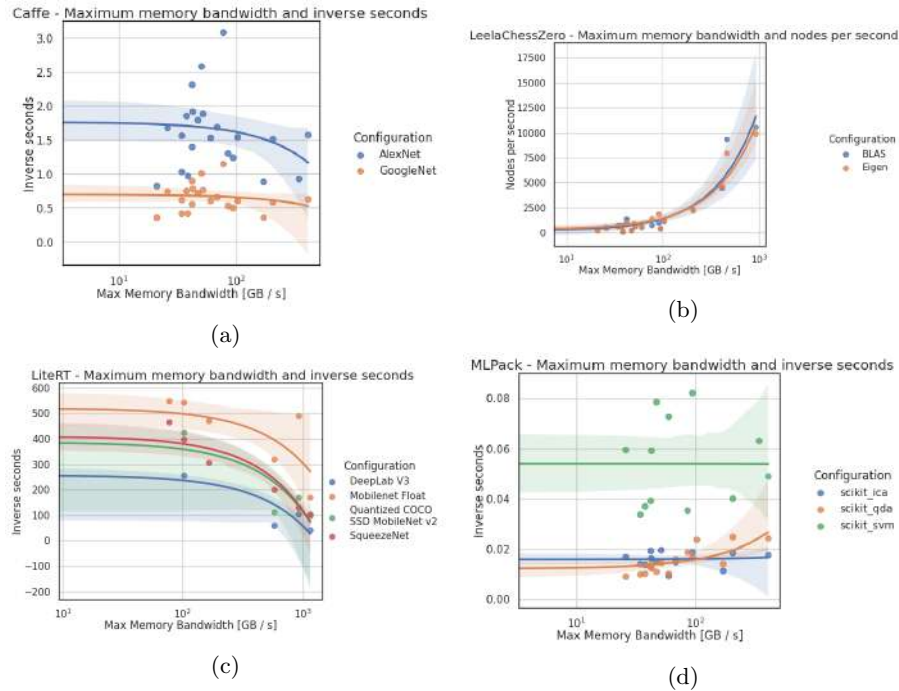


Fig. 7: Dependency of CPU performance from L3-cache in ML tasks within each test suite

Memory Bandwidth On the Figure 8 there are plots with results in benchmarks and maximum memory bandwidth. In most test suites, larger memory bandwidth gives better performance in machine learning tasks.



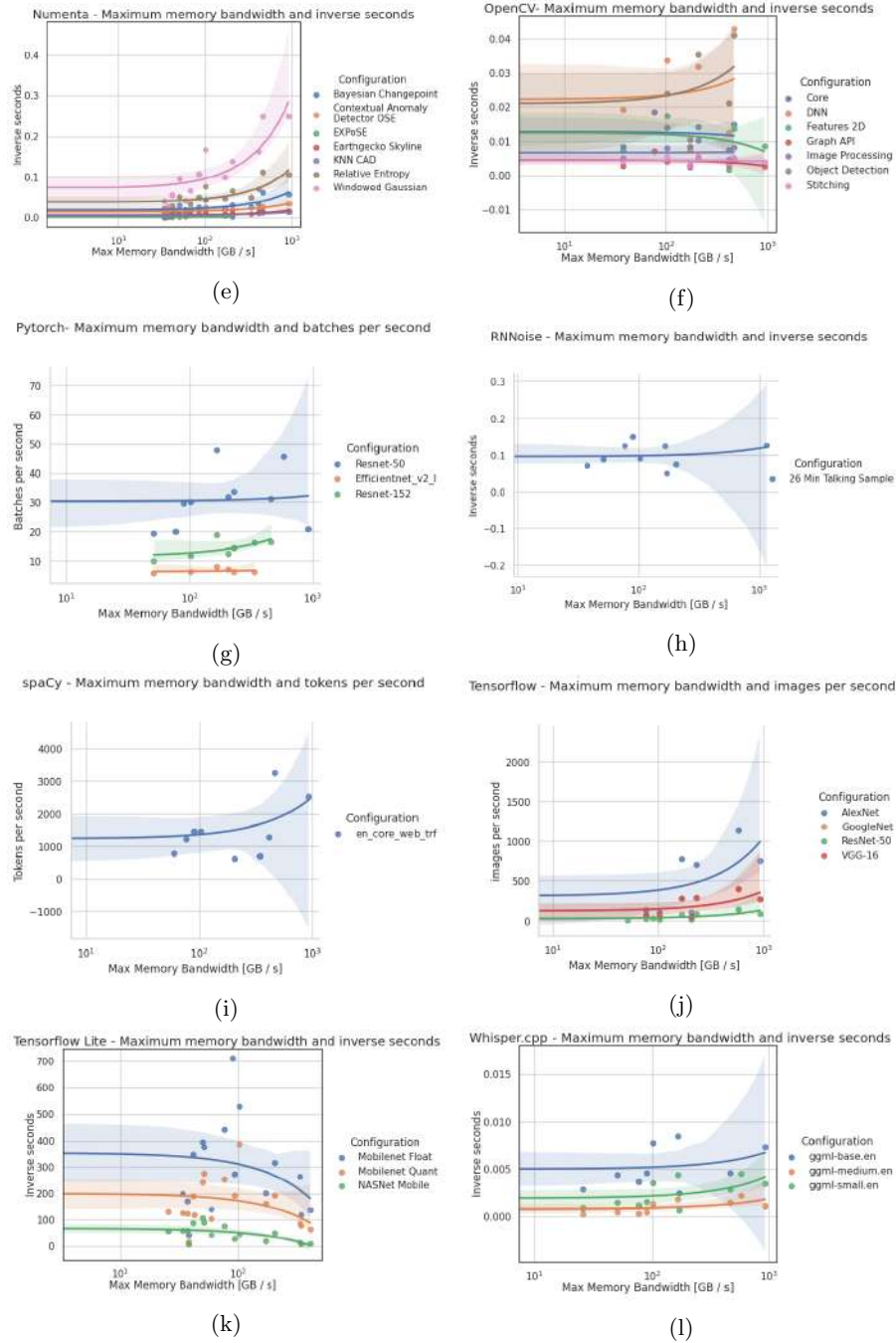
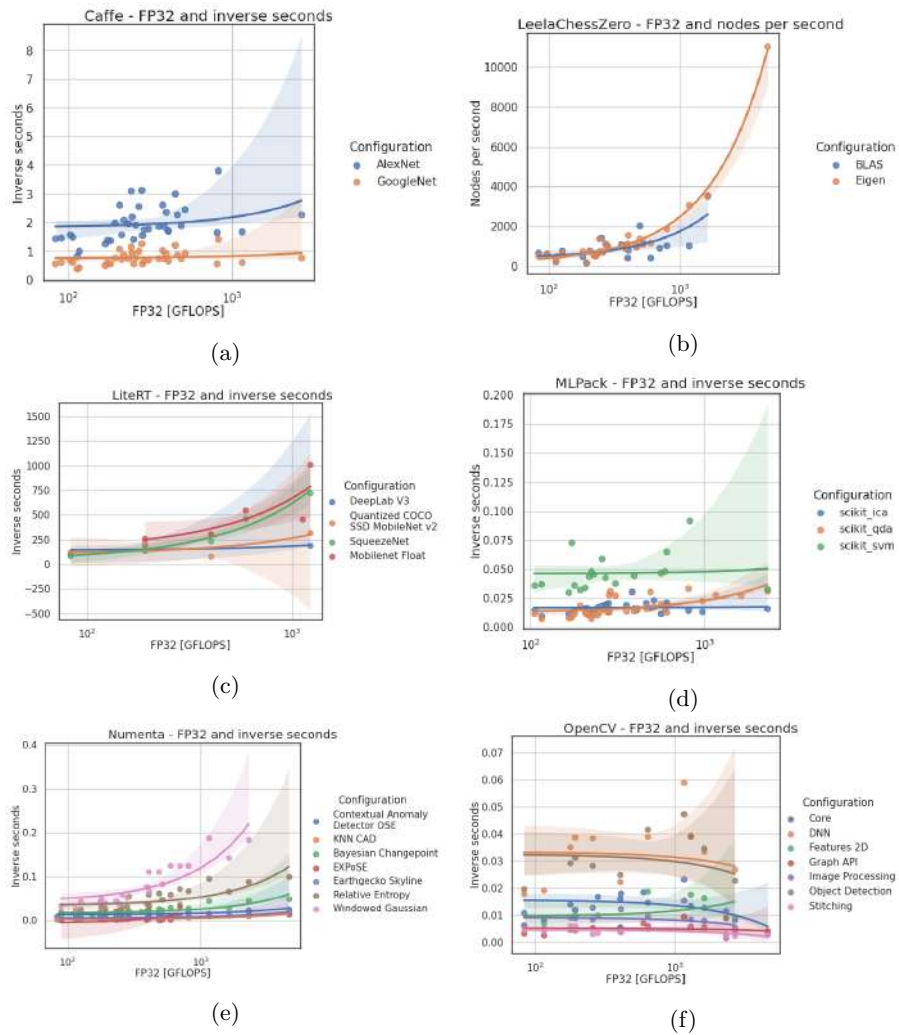


Fig. 8: Dependency of CPU performance from memory bandwidth in ML tasks within each test suite

FP32 Performance On the Figure 9 there are plots with results in benchmarks and FP32 performance. In most cases, processors with higher FP32 performance show better results in ML tasks.



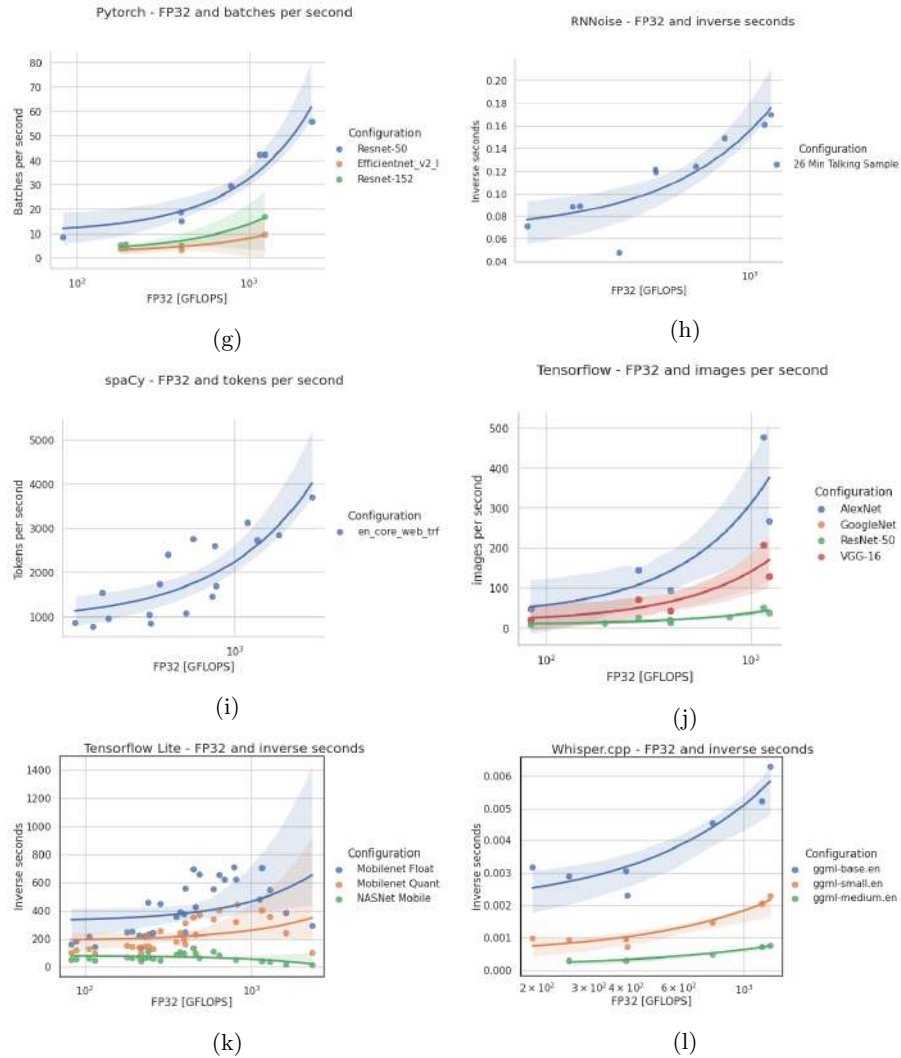
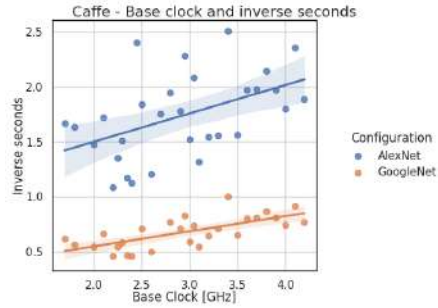
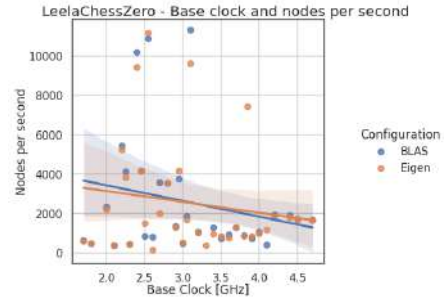


Fig. 9: Dependency of CPU performance from FP32 performance in ML tasks within each test suite

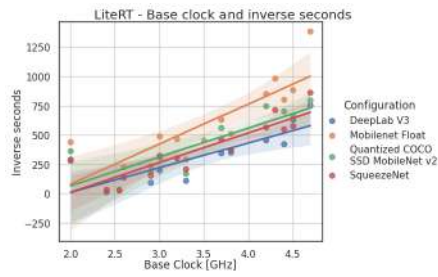
Base clock On the Figure 10 there are plots with results in benchmarks and base clock. In most cases higher clock rate gives better performance, but there are exceptions (e.g. Tensorflow and LeelaChessZero).



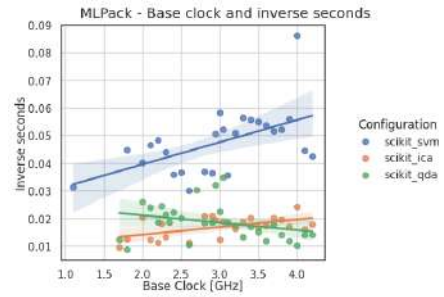
(a)



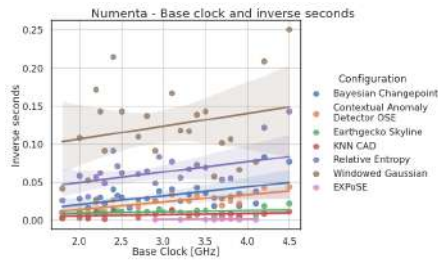
(b)



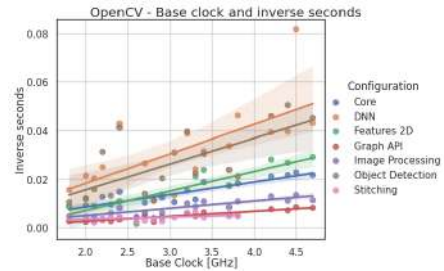
(c)



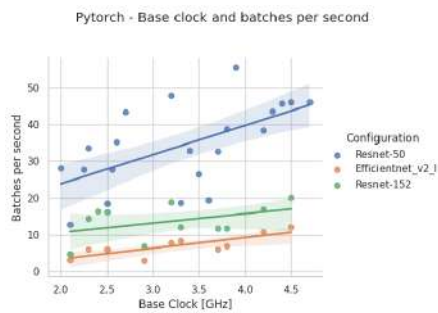
(d)



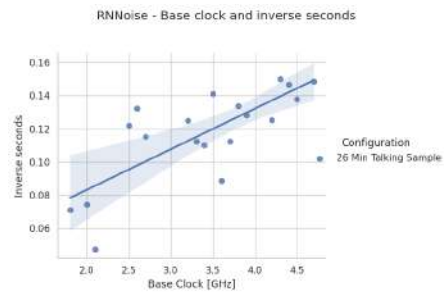
(e)



(f)



(g)



(h)

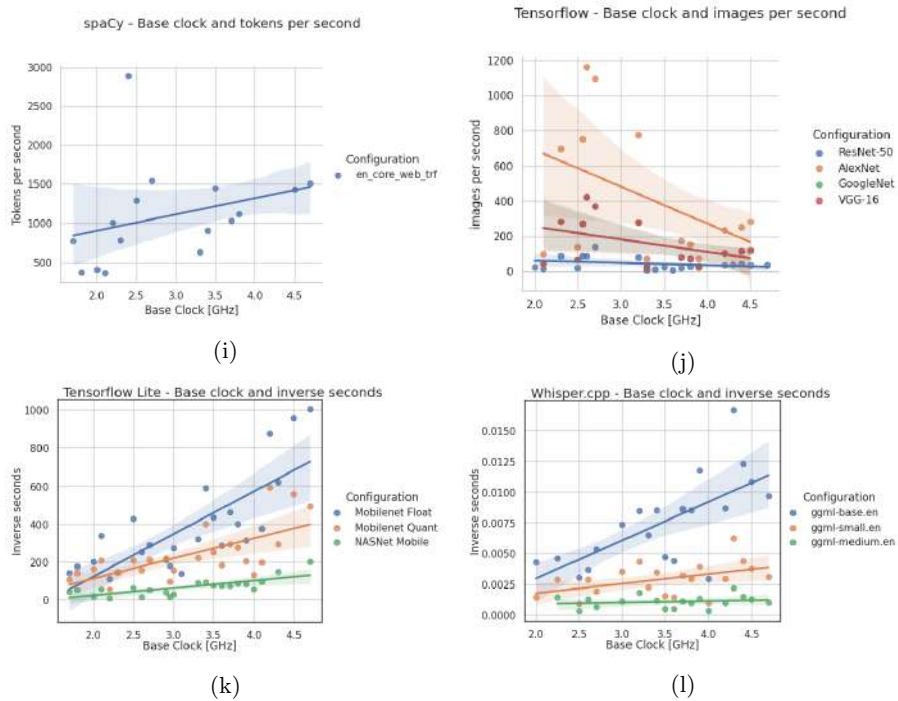
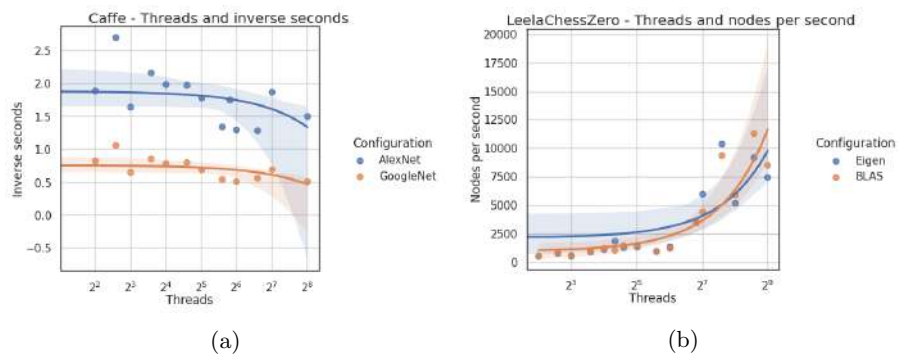
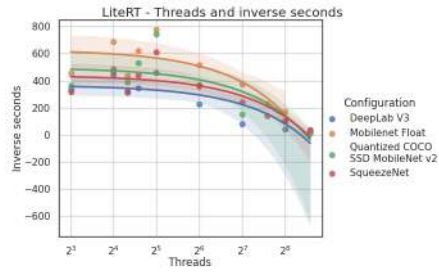


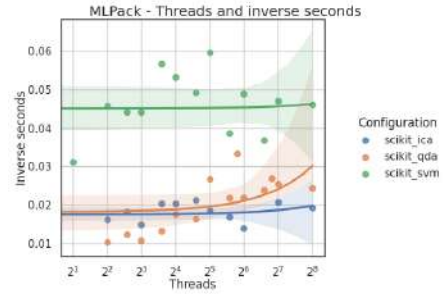
Fig. 10: Dependency of CPU performance from base clock in ML tasks within each test suite

Threads number On the Figure 11 there are plots with results in benchmarks and threads number. In some suites, more threads give better performance, but there are opposite cases. This may be due to the fact that the tests may have been run in single-threaded or multithreaded mode. However, not all suites specify testing mode.

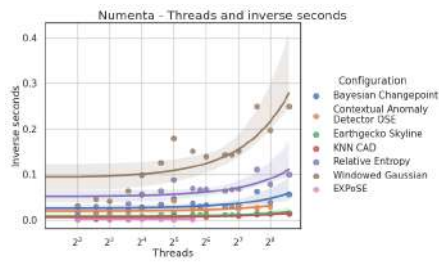




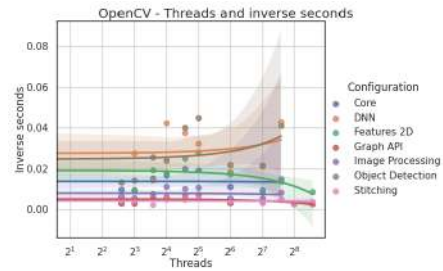
(c)



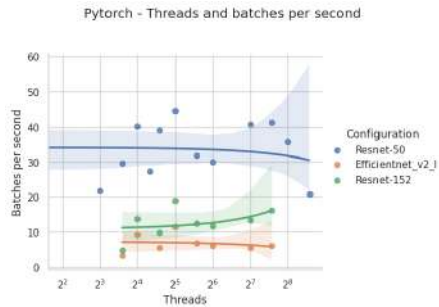
(d)



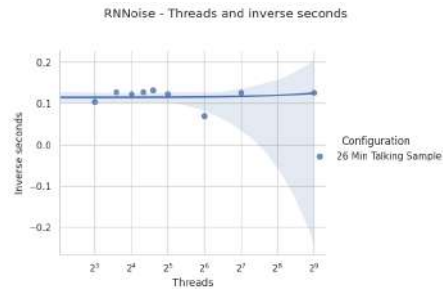
(e)



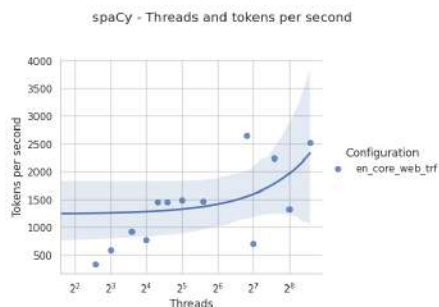
(f)



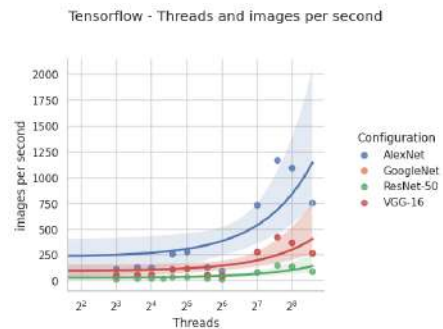
(g)



(h)



(i)



(j)

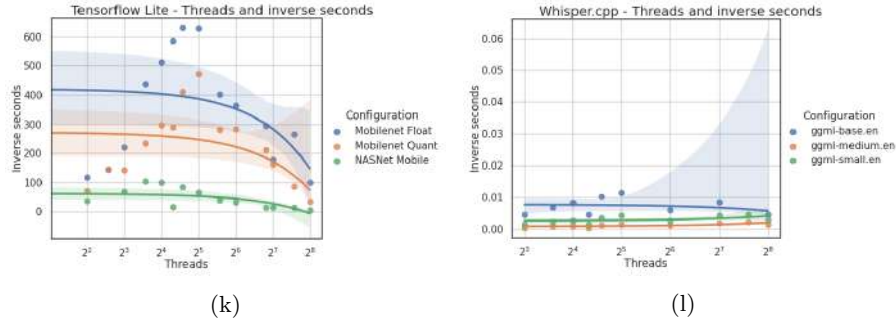


Fig.11: Dependency of CPU performance from threads number in ML tasks within each test suite

4.3 Results for other GPU test suites

This subsection provides graphs showing the dependencies between graphics card specifications and benchmark results within each test suite. When comparing GPU benchmark results in Caffe and in Llama.cpp, some dependencies are observed. For example, the higher the number of cores in GPU, the higher the performance in ML tasks. The same is true for clock rate, FP32 performance and memory bandwidth of GPU.

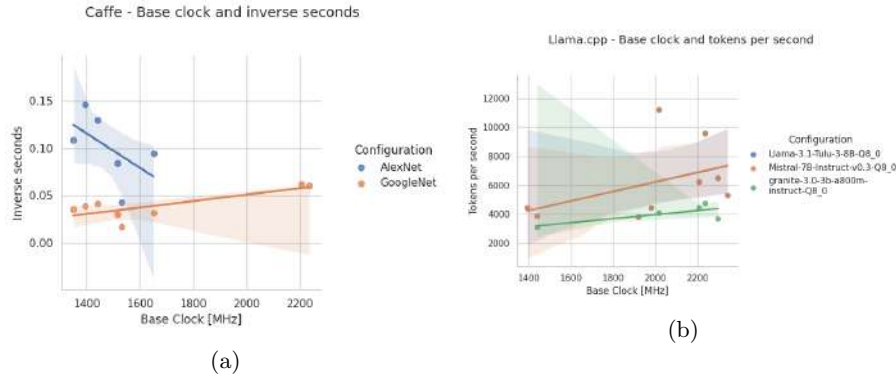


Fig.12: Dependency of GPU performance from cores number in ML tasks in Caffe and Llama.cpp suites

4.4 Results for MLPerf HPC Training

This section presents the key dependencies of computational device characteristics on the performance metrics defined in the previous section. The analysis

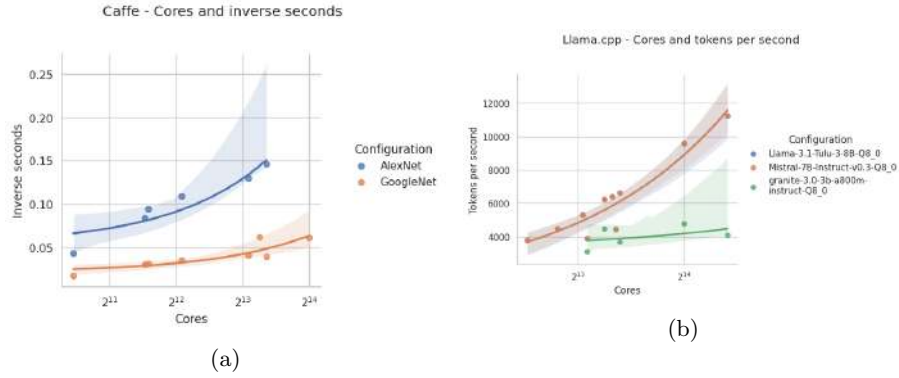


Fig. 13: Dependency of GPU performance from clock rate in ML tasks in Caffe and Llama.cpp suites

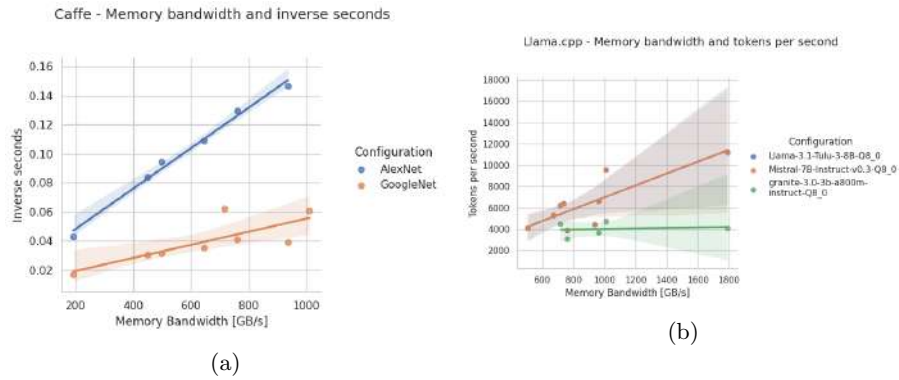


Fig. 14: Dependency of GPU performance from memory bandwidth in ML tasks in Caffe and Llama.cpp suites

includes graphical representations illustrating the relationships between CPU and GPU parameters. The presented graphs provide a comparative evaluation of computing platforms used in MLPerf HPC Training benchmarks, demonstrating performance scaling trends across different supercomputing configurations.

The open dataset provided by MLCommons [20] contains performance measurements obtained from supercomputers, which may either rely solely on CPU resources or employ a hybrid configuration combining both CPU and GPU components. Due to the limited number of runs conducted exclusively on CPUs, a comprehensive analysis based solely on such data would be insufficient. Therefore, in the presented diagrams, these data points are considered alongside results from hybrid configurations. For hybrid systems, an approximate share of CPU performance is estimated based on the ratio of FLOPS contributions within the system. Similarly, in the case of GPUs, all recorded runs inherently involve a hy-

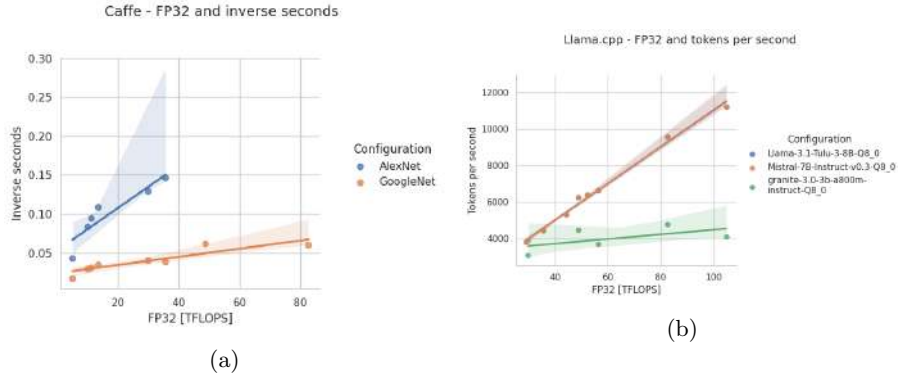


Fig. 15: Dependency of GPU performance from FP32 performance in ML tasks in Caffe and Llama.cpp suites

brid setup utilizing both CPU and GPU resources. Thus, an analogous approach is applied to estimate the GPUs contribution to the overall system performance.

Fig. 16 presents performance diagrams for CPU and GPU, showing their dependency on the balance metric. This metric is defined as the ratio of computational performance (for CPU or GPU) to the peak memory bandwidth of the respective device. It is important to note that, due to the lack of exact memory bandwidth data for the devices used in the performance test, an assumption was made that the maximum possible bandwidth, based on the specifications provided by the manufacturers for computing devices, can be used as a proxy. This assumption may affect the precision of the balance calculations but provides a reasonable baseline for comparative analysis.

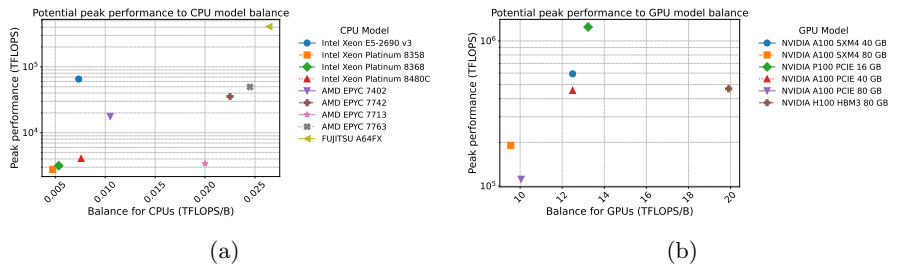


Fig. 16: Dependencies of computing device model potential peak performance to balance metric based on results gained for all MLPerf HPC Training scenarios for CPU (a), for GPU (b)

The diagrams on the Fig. 17 showing the dependencies of CPU potential performance on the balance, now divided by the MLPerf HPC Training scenarios

help to highlight more clearly the relationships between CPU performance and memory bandwidth. The data is aggregated across all MLPerf HPC Training versions, from v1.0 to v3.0. The scenario-specific separation may be crucial, as different scenarios impose varying levels of computational demand on the system. This division allows for a more precise understanding of how each scenario affects CPU performance.

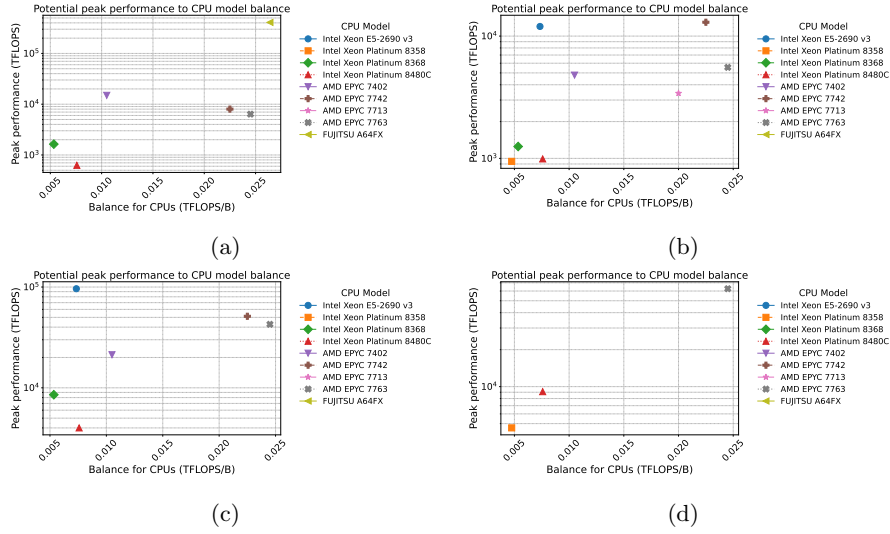


Fig. 17: Dependencies of CPU model potential peak performance to balance metric based on results gained for separate MLPerf HPC Training scenarios CosmoFlow (a), DeepCAM (b), OpenCatalyst (c), OpenFold (d)

When analyzing systems where all CPU-based configurations exhibit Balance 0.025 TFLOPS/B, it can be inferred that the evaluated workloads in benchmarks like MLPerf HPC are likely compute-bound rather than memory-bound. A low Balance value implies that the systems peak computational throughput is small relative to its memory bandwidth. The Y-axis in the discussed plots ranks CPUs by theoretical compute power, with higher values indicating superior performance for compute-bound tasks.

Similarly, Fig. 18 shows diagrams depicting the dependencies of GPU potential performance on the balance, with data separated by the MLPerf HPC Training scenarios. This separation offers a clearer view of how GPU performance relates to memory bandwidth. The results are aggregated across all MLPerf HPC Training versions, from v1.0 to v3.0. By distinguishing between the scenarios, the diagrams provide a more detailed picture of how GPU performance scales with memory bandwidth under different workloads.

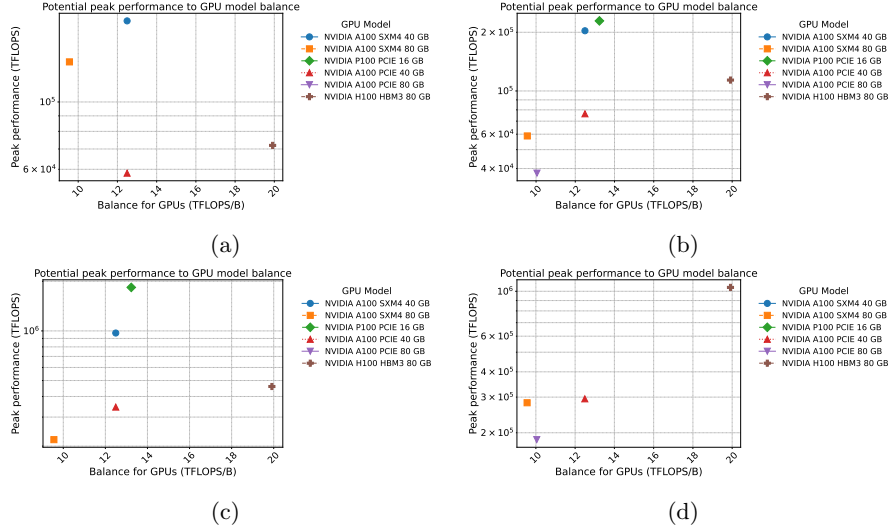


Fig.18: Dependencies of GPU model potential peak performance to balance metric based on results gained for separate MLPerf HPC Training scenarios Cosmoflow (a), DeepCAM (b), OpenCatalyst (c), OpenFold (d)

A Balance value of 20 TFLOPS/B and lower can be treated as exceptionally low for GPUs, indicating severe memory-bound limitations. When Balance falls to 20 or lower, the memory subsystem becomes a critical bottleneck, leaving compute resources underutilized due to insufficient data supply. On a Peak Performance vs. Balance graph, the most capable GPUs for all scenarios whether compute-heavy training or memory-sensitive inference cluster in the upper-right quadrant. This region represents optimal hardware balance: high computational power (Y) paired with efficient memory bandwidth utilization (X).

The diagrams presented in Fig. 19 illustrate the relationships between CPU characteristics and the relative performance of a single CPU, as calculated according to the methodology outlined in the previous section. The results are aggregated by MLPerf HPC Training scenarios and versions, with a global trend displayed for each individual scenario. Notably, these diagrams highlight a significant increase in performance as the number of CPU cores or the size of the L3 cache grows.

5 Conclusion

In this work, we investigated the efficiency of a highly demanded class of machine learning (ML) tasks on different types of processors and graphics accelerators. As benchmarks, we employed the AI-Benchmark suite, the MLPerf HPC benchmarks (including the training scenarios Cosmoflow, DeepCAM, OpenCatalyst,

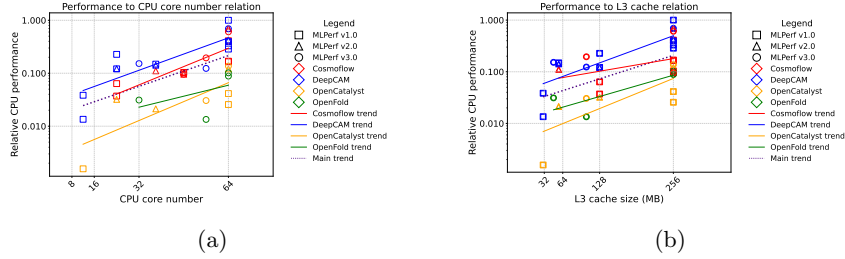


Fig. 19: Dependencies of relative performance impact of a single CPU based on its characteristics, such as core count (a) and L3 cache size (b)

and OpenFold), as well as a variety of tasks from LeelaChessZero, MLPack, Numenta, Whisper, Caffe, and Llama. Separate analyses of processor-based computation and GPU-based computation were conducted to provide a comprehensive performance overview.

From our extensive analysis of a wide range of ML tasks and software packages, we demonstrated that these workloads typically fall into the compute-bound category across both central processing units (CPUs) and graphics processing units (GPUs). In particular, the AI-Benchmark results revealed stark differences in the efficiency of Intel versus AMD processors, also highlighting that processor characteristics have a substantial impact on performance. A Roofline analysis of the AI-Benchmark further confirmed that these ML tasks are compute-bound when run on GPUs; however, it also showed that GPU efficiency still requires significant improvement to fully realize their potential for ML workloads.

In contrast, hybrid benchmarks from MLPerf, which handle large datasets from diverse domains, tend to shift more toward memory-bound behavior. Our analysis also demonstrated very decent scalability with respect to both the number of CPUs and the number of GPUs across all considered MLPerf HPC benchmarks, emphasizing the importance of resource scalability for achieving optimal performance in HPC-based machine learning.

Understanding the interplay between different compute devices and the nature of ML tasks is crucial for selecting the optimal hardware platform to achieve rapid and efficient high-performance computing for ML. The findings on task type, device-specific performance, and overall efficiency provide valuable insights for guiding hardware choices in future HPC-based ML deployments.

6 Acknowledgments

The research was financially supported by the Russian Science Foundation (project No. 20-71-10127), <https://rscf.ru/project/20-71-10127/>. The part of work performed on the computational resources of HPC facilities at HSE University [17] were carried out by G.Promyslov within the framework of the Basic Research Program at HSE University.

References

1. Machine Learning Test Suite Collection - OpenBenchmarking.org. <https://openbenchmarking.org/suite/pts/machine-learning>
2. Abadi, M., Barham, P., Chen, J., et al.: TensorFlow: a system for Large-Scale machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp. 265–283 (2016)
3. Behler, J.: Representing potential energy surfaces by high-dimensional neural network potentials. *J. Phys.: Condens. Matter* **26**(18), 183,001 (2014)
4. Benedict, S.: Energy-aware performance analysis methodologies for HPC architectures - An exploratory study. *Journal of Network and Computer Applications* **35**(6), 1709–1719 (2012)
5. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785–794 (2016)
6. Dongarra, J., Heroux, M.A., Luszczek, P.: High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems. *The International Journal of High Performance Computing Applications* **30**(1), 3–10 (2016)
7. Dongarra, J.J., Luszczek, P., Petitet, A.: The LINPACK benchmark: past, present and future. *Concurrency and Computation: practice and experience* **15**(9), 803–820 (2003)
8. Dunn, B.: Optimizing high performance computing systems, resource utilization and throughput by leveraging machine learning. Ph.D. thesis (2021)
9. Farrell, S., Emani, M., Balma, J., et al.: MLPerf HPC: A Holistic Benchmark Suite for Scientific Machine Learning on HPC Systems. In: 2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC), pp. 33–45 (2021). doi:10.1109/MLHPC54614.2021.00009
10. Gigli, L., Goscinski, A., Ceriotti, M., Tribello, G.A.: Modeling the ferroelectric phase transition in barium titanate with DFT accuracy and converged sampling. *Phys. Rev. B* **110**(2), 024,101 (2024)
11. Goodfellow, I.: Deep learning, vol. 196. MIT press (2016)
12. How, W.B., Chong, S., Grasselli, F., et al.: Adaptive energy reference for machine-learning models of the electronic density of states. *Phys. Rev. Mater.* **9**(1), 013,802 (2025)
13. Ignatov, A.: Ai-benchmark website. <https://ai-benchmark.com>
14. Ignatov, A., Timofte, R., Chou, W., et al.: Ai benchmark: Running deep neural networks on android smartphones. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops, pp. 0–0 (2018)
15. Jouppi, N.P., Young, C., Patil, N., et al.: In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th annual international symposium on computer architecture, pp. 1–12 (2017)
16. Kocer, E., Ko, T.W., Behler, J.: Neural network potentials: A concise overview of methods. *Ann. Rev. Phys. Chem.* **73**(1), 163–186 (2022)
17. Kostenetskiy, P., Chulkevich, R., Kozyrev, V.: HPC resources of the higher school of economics. In: *Journal of Physics: Conference Series*, vol. 1740, p. 012050. IOP Publishing (2021)
18. Liu, Y., Niu, C., Wang, Z., Gan, Y., Zhu, Y., Sun, S., Shen, T.: Machine learning in materials genome initiative: A review. *J. Mater. Sci. Tech.* **57**, 113–122 (2020)

19. Mattson, P., Reddi, V.J., Cheng, C., et al.: MLPerf: An industry standard benchmark suite for machine learning performance. *IEEE Micro* **40**(2), 8–16 (2020)
20. MLCommons: MLPerf HPC Training Benchmarks (2025). URL <https://mlcommons.org/benchmarks/training-hpc/>. Accessed: 2025-02-04
21. Paszke, A., Gross, S., Massa, F., et al.: Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* **32** (2019)
22. Raju, M., Gulhane, K., Gulshan, B.A., et al.: Architectures of high-performance computing systems for machine learning workloads. In: *Integrating Machine Learning Into HPC-Based Simulations and Analytics*, pp. 435–460. IGI Global Scientific Publishing (2025)
23. Wang, H., Zhang, L., Han, J., Weinan, E.: DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Comp. Phys. Comm.* **228**, 178–184 (2018)
24. Williams, S., Waterman, A., Patterson, D.: Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM* **52**(4), 65–76 (2009)

Extraction of properties of anisotropic spin model by deep transfer learning methods* [†]

D.D. Sukhoverkhova^{1,2}, L.N. Shchur^{1,2}

¹National Research University Higher School of Economics, 101000, Moscow, Russia

²Landau Institute for Theoretical Physics, 142432, Chernogolovka, Moscow region,
Russia

We apply supervised deep machine learning techniques to extract properties of the anisotropic Ising model. We consider two cases of anisotropy: orthogonal and diagonal. From the predictions of the neural network, we obtained phase probability functions, from which we measured two quantities: the critical temperature and the critical exponent of the correlation length. We estimated the values of the anisotropy parameter in both cases at which the neural network predictions correctly reproduce the critical behaviour. When the anisotropy is significant, the neural network predicts phases incorrectly. We attribute this to a change in the behaviour of the correlation function. For example, in the case of diagonal anisotropy, these are oscillations of the correlation function that lead to significant deviations in the predictions.

Keywords: phase transitions, Ising model, supervised machine learning, transfer learning.

1. Introduction

Machine learning methods have been widely applied in recent decades in tasks requiring the processing of large amounts of data, including the study of statistical physics systems characterised by an extensive phase space. For example, the research in [1] focused on extracting information about the second-order phase transition in the Ising model. The authors formulated a classification problem for a neural network where the phase transition occurs between ferro- and paramagnetic phases representing two classes. By analysing the probability distribution of these phases, it is possible to estimate the critical transition temperature and some universal properties of the Ising model.

Models with the same dimensionality, symmetry, and degeneracy of the ground state form universality classes, and in the critical region models from the same class have identical properties. Using machine learning, it is possible to identify the properties of a particular universality class to which the model in question belongs [2]. It is known that the correlation length growth rate that diverges at the point of phase transition of the second kind [3], can be estimated using a neural network method. This leads to the question of the feasibility and accuracy of using such an approach to estimate the critical correlation length exponent for other models from similar universality classes. Realising this requires using the knowledge gained during training of the neural network on new data.

Traditional approaches to extract properties of spin models are Monte Carlo class methods: cluster algorithms (Wang-Landau [4], Wolff [5]) and multicanonical algorithms [6]. Spin models have also been investigated by machine learning methods [1,7,8], and there are papers on transfer learning between different universality classes [9]. But to the best of authors' knowledge, no one has investigated the impact of transfer of learning in a single universality class.

We focus on transfer learning in one universality class. For this purpose, we train a convolutional neural network on samples of isotropic Ising model, and then we test the already trained network on anisotropic samples. Anisotropy in the system arises due to changes in the coupling

*This paper is the result of a research project implemented as part of the fundamental research programme of the National Research University Higher School of Economics (HSE).

[†]This research was supported in part through computational resources of HPC facilities at HSE University [17].

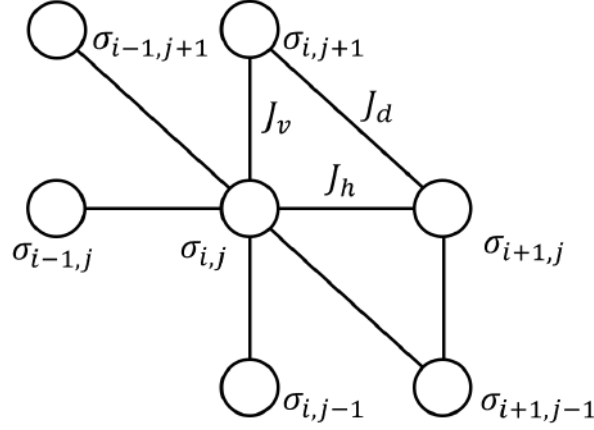


Figure 1. Spin $\sigma_{i,j}$ and its nearest neighbours of the Ising model on a triangular lattice

constants between spins. We consider two cases of anisotropic interactions: in the first case, the anisotropy parameter is the ratio of the magnitude of the vertical bond to the horizontal bond on a square lattice. In the second case, the vertical and diagonal links are equal, and the parameter is the ratio of the magnitude of the diagonal link to the horizontal link on a triangular lattice. Thus, we verify the accuracy with which universal properties of the system can be extracted using neural network analysis.

2. Methods

2.1. Models under study

For studying transfer learning applicability, we consider the Ising model on triangular lattice (Fig. 1):

$$\mathcal{H} = - \sum_{i,j=1}^L \sigma_{i,j} [J_h \sigma_{i+1,j} + J_v \sigma_{i,j+1} + J_d \sigma_{i+1,j+1}]. \quad (1)$$

Here the parameters J_v, J_h, J_d are the coupling constants on the vertical, horizontal and diagonal respectively, L is the system size.

When $J_v = J_h, J_d = 0$, it is the isotropic Ising model in a square lattice with the Z_4 symmetry of a lattice. Respectively, if we vary coupling constants, symmetry is broken and the system becomes anisotropic.

We consider two cases of anisotropy, in each of which the spatial Z_2 symmetry is preserved:

- $J_v = J_h = J = 1, \kappa_1 = J_d/J \neq 0$ – diagonal anisotropy
- $J_d = 0, \kappa_2 = J_v/J_h \neq 1$ – orthogonal anisotropy

Each instanton of the Ising model exists at its own temperature T , depending on which the system can be in an ordered (ferromagnetic) or disordered (paramagnetic) state. Orderedness is characterised by spontaneous magnetisation in the absence of an external magnetic field.

2.2. Neural network output

We feed the instantaneous samples (snapshots) of the model 1 to a convolutional neural network (see ‘Neural network pipeline’ section) and train the network to solve the classification problem.

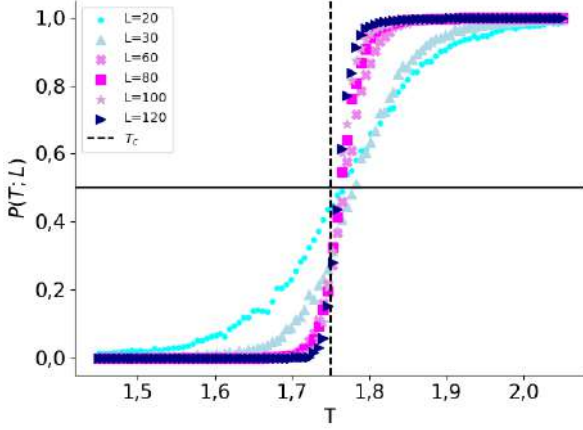


Figure 2. Probability function of samples belonging to paramagnetic phase $P(T; L)$; $\kappa_1 = -0.3$

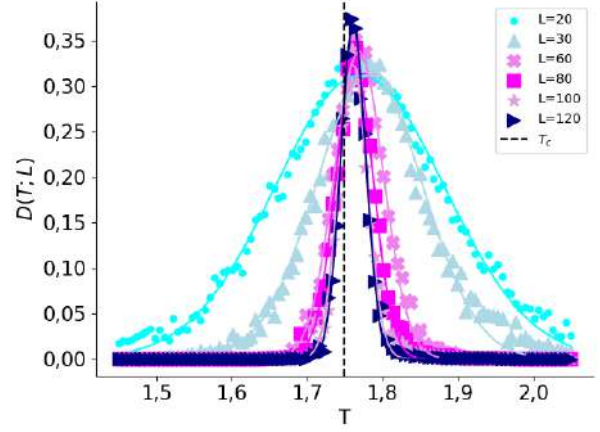


Figure 3. Standard deviations of the probability of samples belonging to paramagnetic phase $D(T; L)$; $\kappa_1 = -0.3$

In the Ising model, a phase transition of the 2nd kind between the ferromagnetic and paramagnetic phases is observed at the critical point T_c , also known as the phase transition temperature. The phase transition temperature of this model is known from the analytical solution [10]:

$$\sinh \frac{2J_v}{k_B T_c} \sinh \frac{2J_h}{k_B T_c} + \sinh \frac{2J_h}{k_B T_c} \sinh \frac{2J_d}{k_B T_c} + \sinh \frac{2J_d}{k_B T_c} \sinh \frac{2J_v}{k_B T_c} = 1, \quad (2)$$

$$J_v + J_h > 0, \quad J_h + J_d > 0, \quad J_d + J_v > 0.$$

Knowing $T_c(J_v, J_h, J_d)$, we divide all data into two classes: 0 – ferromagnetic ($T < T_c$), 1 – paramagnetic ($T > T_c$). After that, we train the neural network to separate the incoming data into two classes.

The output of the neural network is $p_i(T; L)$ – the probability of a sample i with temperature T and size $L \times L$ to be in the paramagnetic phase. Having obtained the predictions for N samples, we construct the paramagnetic phase probability functions $P(T; L)$ 3 and the second moments of this function $D(T; L)$ 4:

$$P(T; L) = \frac{1}{N} \sum_{i=1}^N p_i(T; L), \quad (3)$$

$$D(T; L) = \sqrt{\frac{1}{N} \sum_{i=1}^N (p_i(T; L))^2 - \left(\frac{1}{N} \sum_{i=1}^N p_i(T; L) \right)^2}. \quad (4)$$

2.3. Temperature estimation

We build functions $P(T; L)$ and $D(T; L)$ for several lattice sizes and over a given temperature range estimate the phase transition temperature T_c° found by the neural network: the distribution $D(T; L)$ at each L is approximated by a Gaussian function with mean μ and standard deviation σ . We consider $T_c^- = \mu(L)$ as an estimate of the critical temperature.

All estimates are generated by testing the neural network on finite grid dimensions L . We then make an estimate of the corresponding temperature in the thermodynamic limit based on the shift of the critical temperature in finite dimensions $T_c^-(\infty) = T_c^-(L) + a/L$.

2.4. Critical exponent estimation

It is known [3] that in the universality class of the two-dimensional Ising model, the correlation length ξ diverges at the phase transition point: $\xi \propto \tau^{-1/\nu}$ with critical exponent $\nu = 1$, and $\tau = (T - T_c)/T_c$ is the reduced temperature.

We find an estimate of the exponent ν using the hypothesis [2] that the width of $\sigma(L)$ at finite lattice sizes behaves in the same way as the width of thermodynamic functions [11, 12]:

$$\sigma(L) \propto bL^{-1/\nu}.$$

3. Neural network pipeline

The data are generated using the Metropolis algorithm, with each time point being a black and white image or snapshot. The thermalization time is $20 \times L^{2.15}$ [13]. Once equilibrium is reached, each snapshot is saved in every $2 \times L^{2.15}$ Monte Carlo step, corresponding to $L \times L$ local spin flips.

Each data set is created in the range $T_c \pm 0.3$, comprising 100 temperature points uniformly distributed in increments of 6×10^{-3} . The value of T_c varies with J_v according to the formula 2. For each temperature T in the specified range, we store $N = 2048$ snapshots under isotropic sampling, where $J_v = J_h = 1$. In the case of generating anisotropic datasets when $J_v \neq J_h$, we only store $N = 512$ snapshots. As a result, each dataset contains $N \times 100$ images.

We use a convolutional neural network (CNN) architecture, which includes one convolutional layer, two fully connected layers, and ReLU activation between them [2]. We train neural networks within one epoch to avoid overtraining [14].

The neural networks are trained on isotropic samples with parameters $J_d = 0$, $J_v = J_h = 1$. The pre-trained networks are then tested on each test sample: in the case of diagonal anisotropy, we vary the parameter κ_1 from -0.7 to 1.0 inclusive in steps of 0.1 ; $J_v = J_h = 1$; in the case of orthogonal anisotropy we change the parameter κ_2 : $1, 3/4, 1/2, 1/8, 1/16$; $J_d = 0$.

4. Results

As described above, we estimated the phase transition point T_c^- and the critical exponent $1/\nu$ for different anisotropy parameters κ_1 and κ_2 . The results of the critical temperature estimation are shown in the graphs 4, the $1/\nu$ estimates are collected in the Tables 1–3 and depicted in the graphs 5.

In the results of estimation of both quantities there is a systematic deviation from theoretical values: growth of the relative error of critical temperature and deviation of critical exponent $1/\nu$ from the theoretical value of 1. Such an effect is observed when: $\kappa_1 \leq -0.5$; $\kappa_2 < 1/4$.

Table 1. Estimates of the critical exponent of correlation length ν obtained from analysing the width $\sigma(L)$ of the function $D(T; L)$: the second line is full width, the right half-width $\sigma_r(L)$ of this function is the third line, the left half-width $\sigma_l(L)$ is the fourth line; $\kappa_1 \in [-0.7, 0.0]$

κ_1	−0.7	−0.6	−0.5	−0.4	−0.3	−0.2	−0.1	0.0
$\frac{1}{\nu}$	0.52(15)	0.68(9)	0.88(10)	1.02(5)	1.06(2)	1.07(4)	1.11(2)	1.09(2)

On a triangular lattice with diagonal anisotropy, we observe a deviation of the critical temperature and correlation length exponent when $\kappa_1 < -0.4$. For example, at $-0.4 < \kappa_1 \leq 1.0$ the temperature estimates $\mu(L)$ lie on a curve corresponding to the normalised critical temperature known analytically [15] (Fig. 6a). As the parameter κ_1 decreases, the critical temperature decreases accordingly and the phase diagram shifts towards the region containing the disorder

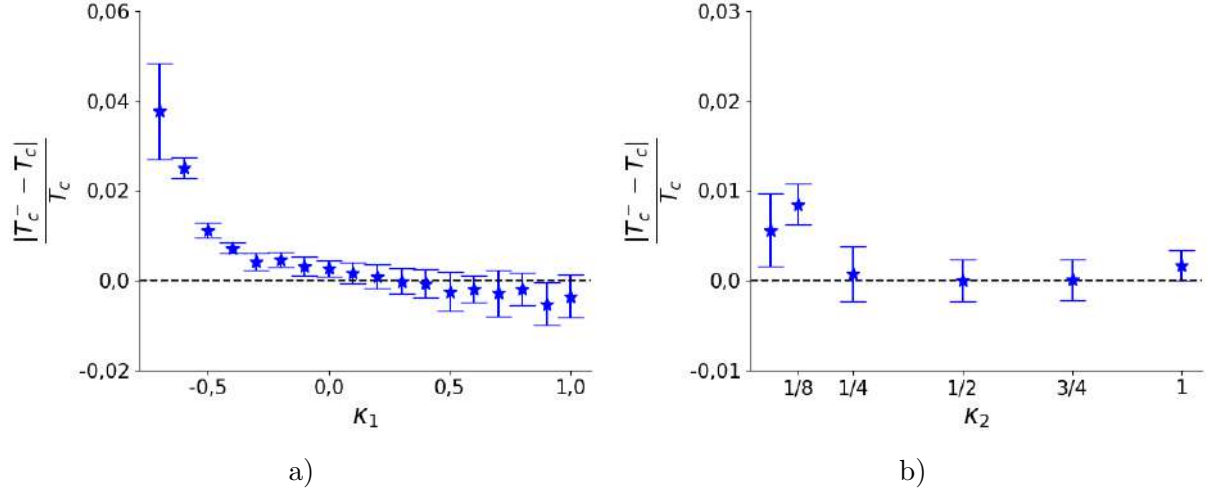


Figure 4. Relative error of the critical temperature estimate T_c^- : a) $\kappa_1 \in [-0.7, 1.0]$, b) $\kappa_2 = 1/16, \dots, 1$

Table 2. Estimates of the critical exponent of correlation length ν obtained from analysing the width $\sigma(L)$ of the function $D(T; L)$: the second line is full width, the right half-width $\sigma_r(L)$ of this function is the third line, the left half-width $\sigma_l(L)$ is the fourth line; $\kappa_1 \in [0.1, 1.0]$

κ_1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$\frac{1}{\nu}$	1.11(5)	1.11(5)	1.10(5)	1.08(3)	1.11(5)	1.07(5)	1.06(5)	1.09(3)	1.09(4)	1.08(7)

Table 3. Estimates of the critical exponent of correlation length ν obtained from analysing the width $\sigma(L)$ of the function $D(T; L)$: the second line is full width, the right half-width $\sigma_r(L)$ of this function is the third line, the left half-width $\sigma_l(L)$ is the fourth line; $\kappa_2 = 1/16, \dots, 1.0$

κ_2	1	3/4	1/2	1/4	1/8	1/16
$\frac{1}{\nu}$	1.10(3)	1.07(4)	1.03(3)	1.06(4)	0.87(7)	0.66(11)

temperature. In this region, pairwise correlated spin oscillations are observed in the system, which arise due to the diagonal antiferromagnetic interaction. During training, the neural network was fed with data that did not contain such oscillations, so the results of the estimation of the quantities of interest deviate from the theoretical values in the same regions.

As for the orthogonal anisotropy, we assume that the deviations at small parameters κ_2 are due to the peculiarities of the correlation function in this region. In [16], a decomposition of the correlation function $\langle \sigma_{00}\sigma_{ij} \rangle$ over radially measured distance R was obtained (sign + or - stands for temperature region: greater or less than T_c):

$$\langle \sigma_{00}\sigma_{ij} \rangle = F_{\pm}(t)/R^{1/4} + F_{1\pm}(t)/R^{5/4} + o(R^{-5/4}). \quad (5)$$

Here R is the radial radius, in the isotropic case $R = L\sqrt{2}$; t is the reduced temperature in variables $z_1 = \tanh \beta J_h$, $z_2 = \tanh \beta J_v$:

$$t = |z_1 z_2 + z_1 + z_2 - 1| \left[z_1 z_2 (1 - z_1^2)(1 - z_2^2) \right]^{1/4} R. \quad (6)$$

For example, we construct the ratio R_1 :

$$R_1 = \frac{F_{1-}(t)}{tF_{-}(t)}. \quad (7)$$

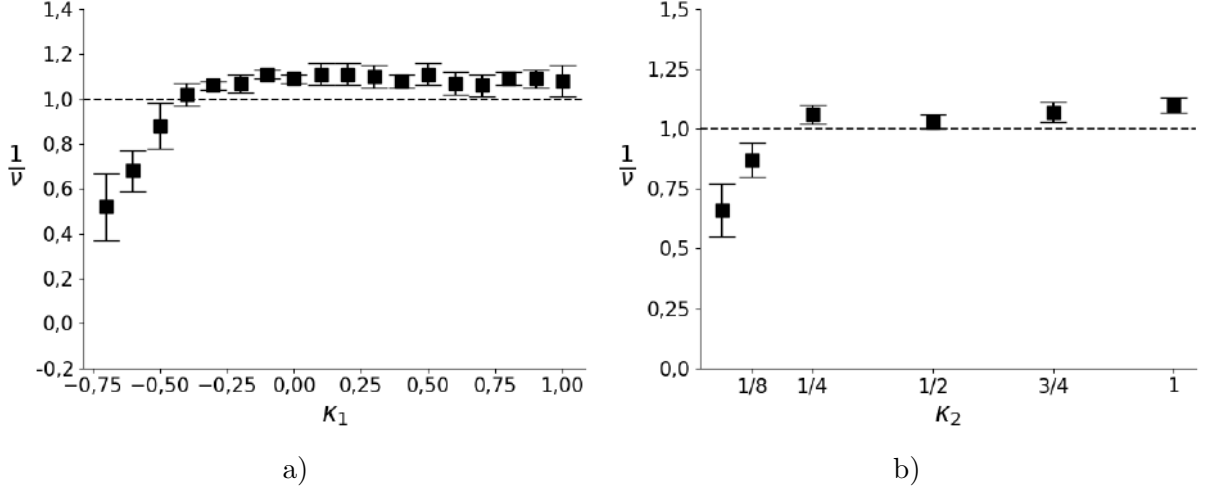


Figure 5. Estimation of the critical correlation length exponent $(1/\nu)_-$: black squares, $1/\nu$, full width; red triangles, $(1/\nu)_r$, right half-width; green stars, $(1/\nu)_l$, left half-width; a) $\kappa_1 \in [-0.7, 1.0]$, b) $\kappa_2 = 1/16, \dots, 1$

Fig. 6b repeats Fig. 2 from the paper [16], and at the points $\kappa_2 = 1/8, 1/16$, an increase in the R_1 ratio is noticeable. This means that in samples at lower temperatures than T_c possessing orthogonal anisotropy, the influence of the correction term $F_{1-}(t)R^{-5/4}$ increases, which affects the spatial behaviour of the correlation function, which is also not taken into account when training the neural network.

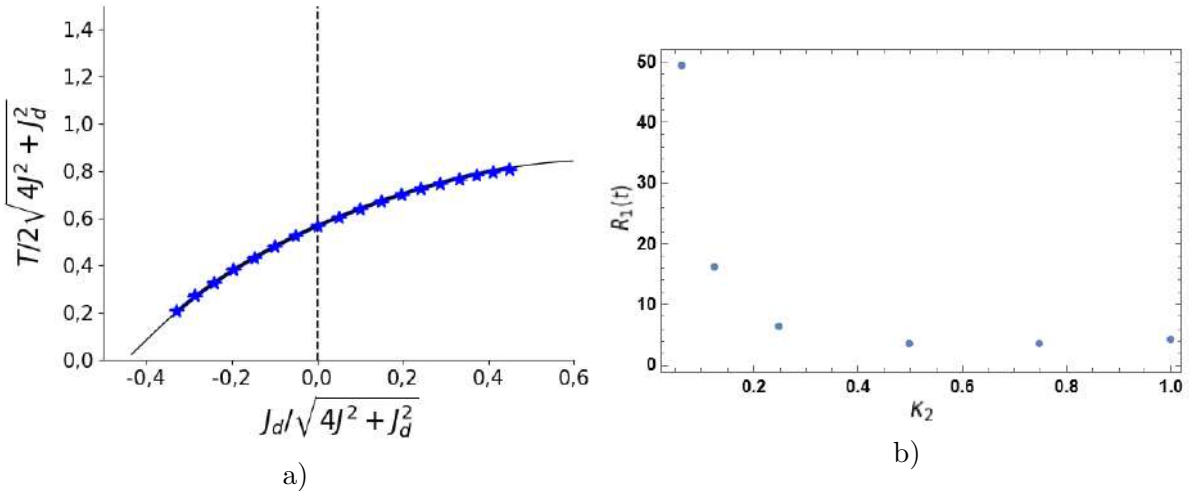


Figure 6. a) Estimations of critical temperature T_c^- on the same plot with precise critical temperature from paper [15]; b) ratio 7 vs anisotropy parameter κ_2

5. Discussion

In this work, we investigated the applicability of transfer learning within one universality class of the two-dimensional Ising model. We trained a convolutional neural network to discriminate between the two phases of an isotropic model, and then tested the pre-trained network on anisotropic samples. From the predictions of the network, we extracted estimates of the critical temperature and the critical correlation length exponent. The results showed that we estimate both quantities according to their theoretical values (within the statistical error) when transferring the training, but the region of correctness is limited.

The hypothesis is that at strong anisotropy there are non-trivial effects related to spatial oscillations occurring in the system. In the case of diagonal anisotropy, antiferromagnetic diagonal interactions are influential, leading to frustrations due to the unattainability of the ground state. In the case of orthogonal anisotropy, the increase of correction term in the correlation function leads to a different behaviour on the short scales, limiting the applicability of transfer learning. Further work on this problem will consist in testing the hypotheses put forward.

References

1. Carrasquilla J., Melko R.G. Machine learning phases of matter // *Nature Physics*. 2017. Vol. 13, no. 5. P. 431–434. DOI: 10.1038/nphys4035.
2. Chertenkov V., Burovski E., Shchur L. Finite-size analysis in neural network classification of critical phenomena // *Physical Review E*. 2023. Vol. 108, no. 3. P. L032102. DOI: 10.1103/PhysRevE.108.L032102.
3. Onsager L. Crystal statistics. I. A two-dimensional model with an order-disorder transition // *Physical Review*. 1944. Vol. 65, no. 3-4. P. 117. DOI: 10.1103/PhysRev.65.117.
4. Wang F., Landau D.P. Determining the density of states for classical statistical models: A random walk algorithm to produce a flat histogram // *Phys. Rev E*. 2001. Vol. 64. P. 056101. DOI: 10.1103/PhysRevE.64.056101.
5. Wolff U. Collective Monte Carlo updating for spin systems // *Phys. Rev. Lett.* 1989. Vol. 62, no. 4. P. 361–364. DOI: 10.1103/PhysRevLett.62.361.
6. Berg B.A., Neuhaus T. Multicanonical ensemble: A new approach to simulate first-order phase transitions // *Physical Review Letters*. 1992. Vol. 68, no. 1. P. 9. DOI: 10.1103/PhysRevLett.68.9.
7. van Nieuwenburg E.P.L., Liu Y.H., Huber S.D. Learning phase transitions by confusion // *Nature Physics*. 2017. Vol. 13, no. 5. P. 435–439. DOI: 10.1038/nphys4037.
8. Ng K.K., Yang M.F. Unsupervised learning of phase transitions via modified anomaly detection with autoencoders // *Physical Review B*. 2023. Vol. 108, no. 21. P. 214428. DOI: 10.1103/PhysRevB.108.214428.
9. Chertenkov V., Shchur L. Supervised and Transfer Learning for Phase Transition Research // *Russian Supercomputing Days*. Springer Nature Switzerland, Cham, 2024. P. 434–449. DOI: 10.1007/978-3-031-78459-0_31.
10. Houtappel R.M.F. Order-disorder in hexagonal lattices // *Physica*. 1950. Vol 16, no. 5. P. 425–455. DOI: 10.1016/0031-8914(50)90130-3.
11. Fisher M.E., Ferdinand A.E. Interfacial, boundary, and size effects at critical points // *Physical Review Letters*. 1967. Vol. 19, no. 4. P. 169. DOI: 10.1103/PhysRevLett.19.169.
12. Ferdinand A.E., Fisher M.E. Bounded and inhomogeneous Ising models. I. Specific-heat anomaly of a finite lattice // *Physical Review*. 1969. Vol. 185, no. 2. P. 832. DOI: 10.1103/PhysRev.185.832.
13. Chertenkov V., Shchur L. Universality classes and machine learning // *Journal of Physics: Conference Series*. 2021. Vol. 1740, no. 1. P. 012003. DOI: 10.1088/1742-6596/1740/1/012003.
14. Chertenkov V.I. PhD thesis, National Research University Higher School of Economics (2024).

15. Stephenson J. Ising model with antiferromagnetic next-nearest-neighbor coupling: Spin correlations and disorder points // Physical Review B. 1970. Vol. 1, no. 11. P. 4405.
DOI: 10.1103/PhysRevB.1.4405.
16. Wu T.T. et al. Spin-spin correlation functions for the two-dimensional Ising model: Exact theory in the scaling region // Physical Review B. 1976. Vol. 13, no. 1. P. 316.
DOI: 10.1103/PhysRevB.13.316.
17. Kostenetskiy P.S., Chulkevich R.A., Kozyrev V.I. HPC Resources of the Higher School of Economics // Journal of Physics: Conference Series. 2021. Vol. 1740, no. 1. P. 012050.
DOI: 10.1088/1742-6596/1740/1/012050.

Comparative Study of LoRA and Full Fine-Tuning in Large Language Models*

E.V. Surikova, E.A. Sabidaeva

National Research University Higher School of Economics

In recent years, large language model (LLM) fine-tuning has become a key area of research in the field of natural language processing (NLP). However, taking into account constraints and high costs of training, researchers are increasingly turning to PEFT [1] methods such as LoRA, which demonstrate good results. In this study, we conducted an experimental comparison of full tuning and LoRA-based tuning methods using the QWEN2-7b [2] model as the base one. We compared performance across three distinct tasks within the domain of economics and information technology: classification task, short question answering and long question answering. The results substantiate that LoRA facilitates significant reductions in computational and storage resources, thereby simplifying model deployment processes. However, full fine-tuning provides higher accuracy for all types of ML tasks considered. Additionally, we investigated the hypothesis that integrating various task types within the same domain could enhance generation quality. However, our findings indicate that combining training data from classification tasks and QA resulted in a decline in performance metrics, aligning them with those achieved by the LoRA approach.

Keywords: fine-tuning, LoRA, full-parameter, LLM.

1. Introduction

Low-rank adaptation (LoRA) [3] is a widely used method for fine-tuning large language models (LLMs). This approach utilizes low-rank matrices, enabling the preservation of a significant portion of valuable information while optimizing memory usage. In this study, we aim to assess the performance of LoRA in comparison to full fine-tuning of LLMs across various tasks, such as classification and text generation in the domains of economics and digital technologies.

The LoRA method is based on freezing the pre-trained weighting matrix $W_{pretrained} \in R^{d \times k}$, and training solely on the low-rank matrix Δ , as follows:

$$W_{finetuned} = W_{pretrained} + \Delta, \quad (1)$$

$$\Delta = AB, \quad A \in R^{d \times r}, \quad B \in R^{r \times k}, \quad (2)$$

where $A_0 \sim N(0,1), B_0 = 0$.

$W_{pretrained}$ needs to be adapted and the rank should be $r \ll d, k$, so that instead of $d \times k$ parameters, only $d \times r + r \times k$ parameters are trained, which reduces the memory and FLOPS needed to compute the gradient. For example, applying $r = 16$ LoRA to a 7B weight matrix with $d = k = 4096$ requires $< 1\%$ of the original number of parameters.

The article is structured as follows. The first section provides an overview of the data used for both classification and QA tasks. Next, we delve into the experimental phase, detailing the hyperparameter settings and the metrics obtained from the training process. In the related works section, we examine existing datasets within the relevant domains and analyze papers that compare full fine-tuning and LoRA methods. Finally, we summarize the key findings and highlight the main results of the study.

2. Data overview

2.1 Dataset for classification task

The dataset was generated using gpt4o-mini [4]. It was given a prompt which instructed the model to attribute each fragment with a class:

* This research was supported in part through computational resources of HPC facilities at HSE University.

Objective:

You are tasked with determining whether a given text contains a specific theme that is classified as "Class A". Your output should be either 1 (indicating the presence of the theme) or 0 (indicating its absence). Return 1 if the theme is present in the text, or 0 if it is not.

Example:

Input Text: "The advancements in renewable energy technology are crucial for combating climate change and fostering sustainable development."

Class: "EnergyTech",

Output: 1.

Input Text: "The latest trends in fashion highlight the importance of personal style and self-expression.",

Class: "AI",

Output: 0.

The fragments inputted to the model represent excerpts from media articles published over the past three years in the domain of economics and information technology. As a result, we formed the dataset in which each text fragment is assigned one of the nine classes: "Science, Technology and Innovation", "Telecom", "AI", "FinTech", "Digital Transformation", "EdTech", "Business", "EnergyTech", and "MedTech". The training part of the dataset comprises 9000 rows, while the test dataset contains 1800 rows. The total size of the dataset in tokens is 2M. A sample from the dataset is presented in Table 1.

Table 1. An example of data input from the dataset for the classification task

Texts	Class
Financial Services 2030 About 43% of the surveyed experts assessed ESG at the moment as a “hype”, especially promoted by marketing departments of many companies, and found it difficult to estimate the preservation of ESG agenda in the next 10 years.	FinTech

2.2 Dataset for QA task

For the QA task, we created another dataset within the same domain using textual data from StackExchange [5]. It was also enriched with the data generated with gpt4o-mini. To achieve this, we created a prompt that requested the model to generate both short and long answers based on the information from the original data:

SFOA prompt

Task:

Based on the provided Stack Exchange answer, create a short answer that captures the main point or solution presented in the answer. Your answer should be clear and to the point, ideally no longer than 2-3 sentences.

Example:

Input:

Question: "How can you improve your coding skills?"

Stack Exchange Answer: "To improve your coding skills, practice regularly by working on small projects. Collaborate with others and seek feedback on your code. Additionally, consider contributing to open-source projects to gain real-world experience."

Output:

Short answer: "To enhance your coding skills, engage in regular practice through small projects, collaborate with peers for feedback, and contribute to open-source initiatives for practical experience."

LFQA prompt

Task:

Using the provided Stack Exchange answer, create a detailed answer that expands on the ideas presented. Include examples, explanations, and any relevant context to provide a comprehensive understanding of the topic.

Example:

Input:

Question: "What is Git for?"

Stack Exchange Answer: "Using version control systems like Git allows developers to track changes in their code, collaborate with others, and manage different versions of their projects effectively."

Output:

Long answer: "Version control systems, particularly Git, are essential tools for modern software development. They enable developers to keep a detailed history of changes made to their codebase, allowing for easy tracking and rollback if necessary. For instance, when working in teams, Git facilitates collaboration by allowing multiple developers to work on different features simultaneously without overwriting each other's work. Furthermore, Git's branching capabilities allow developers to experiment with new ideas in isolated environments before merging them into the main project. Overall, mastering version control is crucial for maintaining code integrity and enhancing team productivity."

As a result, the dataset consisting of context, query, and examples of both short and long answers was generated. The training set includes 7340 rows, while the test set contains 1200 rows. The total size of the dataset in tokens for Short-Form Question Answering (SFQA) is 6.8M, and for Long-Form Question Answering (LFQA), it is 8.2M. A sample from the dataset is presented in Table 2.

Table 2. An example of data input from the dataset for the QA task

Context	Query	Short answer	Long answer
The Buenos Aires AI Plan is an overarching strategy for the generation of a positive impact through the use and development of artificial intelligence. The plan breaks down potential risks and opportunities, specifies main objectives, provides a structure for its implementation and describes the main challenges to achieve this...	What are the main objectives of the AI strategy being implemented in Buenos Aires?	The main goal is to use AI to develop the city for the benefit of the citizens and ensure the sustainability of the city.	The Future City AI Strategy, launched in August 2021, has three main objectives: 1. to use artificial intelligence to develop the city; ...

These datasets were combined in order to investigate the impact of different types of tasks on each other, as well as the amount of data. To compare full parameter fine-tuning and LoRA, the dataset from Stack Exchange was used.

The number of tokens of system prompt length for each type of task is given in Table 3.

Table 3. The number of tokens in the instruction, depending on the type of task

Task	System prompt length (tokens)
Classification	146
SFQA	65
LFQA	78

Figure 1 shows the average answer length for each type of task.

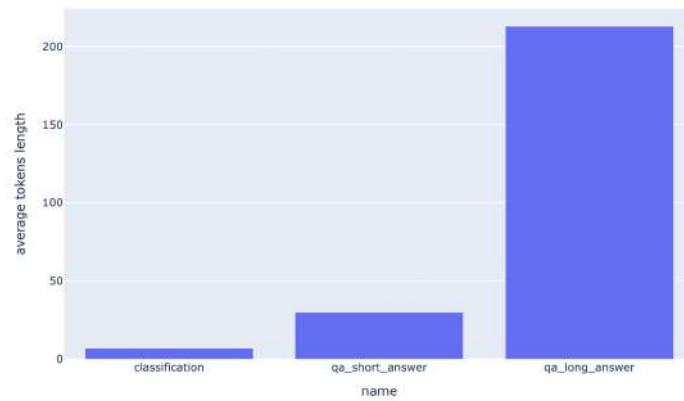


Figure 1. Golden average tokens

3. Experimental part

Qwen2-7B [6] was chosen as the base model for fine-tuning. Qwen2 language models, including pre-trained and instruction-tuned models in 5 sizes, including 0.5B, 1.5B, 7B, 57B-A14B, and 72B. Qwen2-7B supports context length of up to 131,072 tokens. Hyperparameter settings for full fine-tuning and LoRA are presented in Tables 4–5.

Table 4. Hyperparameter settings:
full parameter fine-tuning

Hyperparameter	Value
sequence_len	1024
gradient_accumulation_steps	8
micro_batch_size	1
num_epochs	2
bf16	true
optimizer	paged_adamw_8bit
lr_scheduler	cosine
learning_rate	2e-5
warmup_steps	10

Table 5. Hyperparameter settings:
LoRA fine-tuning

Hyperparameter	Value
adapter	lora
lora_r	32
lora_alpha	16
lora_dropout	0.05
Precision	fp16
gradient_accumulation_steps	4
micro_batch_size	2
num_epochs	2
learning_rate	3e-4
optimizer	paged_adamw_8bit

Table 6 demonstrates the structure of the LoRA matrix. The use of LoRA with a rank of 8 significantly reduces the number of trainable parameters compared to fine-tuning the entire original matrix. Instead of updating all $152\,064 \times 3\,584 = 543\,360\,256$ parameters, only $(152\,064 \times 8) + (8 \times 3\,584) = 1\,216\,512 + 28\,672 = 1\,245\,184$ parameters are updated. This reduction makes training faster and less resource-intensive.

Table 6. Structure of the LoRA matrix for the classification task

Matrix parameters	Sizes
Original matrix	152064 x 3584
LoRA rank	8
Additional matrix	4 x 3584
LoRA matrix A	152064 x 8
LoRA matrix B	8 x 3584

The total number of trained parameters is 1 245 184.

The experiment was conducted on 2 NVIDIA A100 80 GB GPUs (HPC Cluster "cCHARISMa").

3.1 Classification task metrics

Table 7 presents performance metrics for various classes in a classification task. The metrics include precision, recall, F1-score, and support for each class, along with overall accuracy and average scores. Qwen2:7b-instruct was chosen as the baseline model. Its metrics are given in Table 8.

It can be concluded that the models fine-tuned with LoRA and baseline model demonstrate lower quality with the class “Digital Transformation” and “Science, Technology and Innovation”.

Table 7. Metrics for fine-tuning LoRA Qwen2:
7b-instruct-clf-lora

class	precision	recall	f1-score	support
AI	0.62	0.73	0.67	200
Business	0.84	0.56	0.67	200
Digital Transformation	0.45	0.56	0.50	200
EdTech	0.93	0.9	0.91	200
EnergyTech	0.83	0.76	0.79	200
FinTech	0.76	0.8	0.78	200
MedTech	0.83	0.91	0.87	200
Science, Technology and Innovation	0.44	0.52	0.48	200
Telecom	0.8	0.58	0.68	200
accuracy			0.70	1800
macro avg	0.72	0.70	0.70	1800
weighted avg	0.72	0.70	0.70	1800

Table 8. Metrics for Qwen2:
7b-instruct

class	precision	recall	f1-score	support
AI	0.46	0.77	0.57	200
Business	0.24	0.2	0.22	200
Digital Transformation	0.28	0.2	0.23	200
EdTech	0.75	0.9	0.82	200
EnergyTech	0.74	0.8	0.76	200
FinTech	0.78	0.68	0.72	200
MedTech	0.83	0.8	0.82	200
OTHER	0	0	0	0
Science, Technology and Innovation	0.26	0.17	0.21	200
Telecom	0.57	0.28	0.37	200
accuracy			0.53	1800
macro avg	0.49	0.48	0.47	1800
weighted avg	0.54	0.53	0.52	1800

Table 9. Full parameter fine-tuning Qwen2:
7b-instruct-clf

class	precision	recall	f1-score	support
AI	0.72	0.9	0.8	200
Business	0.89	0.66	0.76	200
Digital Transformation	0.79	0.96	0.87	200
EdTech	0.93	0.99	0.96	200
EnergyTech	0.89	0.98	0.93	200
FinTech	0.95	0.95	0.95	200
MedTech	0.93	0.9	0.91	200
Science, Technology and Innovation	0.7	0.56	0.62	200
Telecom	0.91	0.78	0.84	200
accuracy			0.85	1800
macro avg	0.86	0.85	0.85	1800
weighted avg	0.86	0.85	0.85	1800

Full parameter fine-tuning shows better results on the classification task compared to LoRA and baseline model. On average F1 metrics have 0.85 (full parameter fine-tuning), 0.7 for the LoRA and 0.52 for the baseline model.

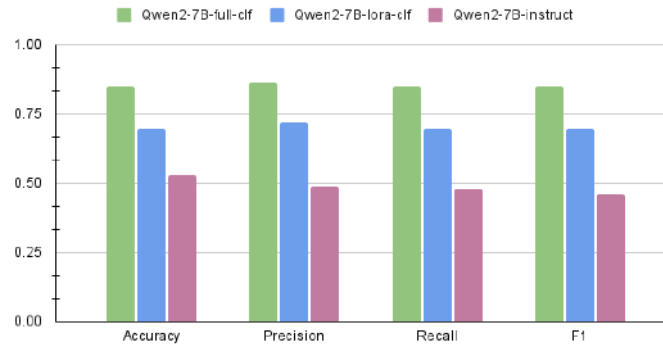


Figure 2. Metrics for classification tasks

Both full-parameter fine-tuning and LoRA exhibit unique characteristics in terms of loss function convergence. While full-parameter fine-tuning offers flexibility and potentially higher performance on complex tasks, LoRA provides efficiency and speed. Comparative analysis of the loss functions for LoRA and full-parameter fine-tuning, demonstrating that both methods exhibit comparable convergence rates (Figure 3).



Figure 3. Training loss for classification task

3.2 QA task metrics

The models were evaluated on the SFQA and LFQA tasks using offline metrics: BERTScore [7], BLEU [8], ROUGE [9] and semantic similarity [10]. The results are presented in Table 10. Average response length of the model with full parameter fine-tuning is most similar to golden responses. There is a slight difference between the models according to the BERTScore metric.

Table 10. Metrics for the QA task on short and long answers

model_name	bleu	ref_len	resp_len	rouge_1	bert_score_f1	sem_sim
qwen2_7b_qa_short_lora	0.119	9.583	9.121	0.312	0.732	0.81
qwen2_7b_qa_long_lora	0.013	336.991	160.425	0.163	0.682	0.732
qwen2_7b_qa_short_full	0.227	9.583	10.053	0.395	0.819	0.893
qwen2_7b_qa_long_full	0.02	336.991	289.002	0.151	0.679	0.89
baseline model short	0.023	9.583	11.123	0.327	0.759	0.813
baseline model long	0.015	336.991	228.266	0.136	0.662	0.854

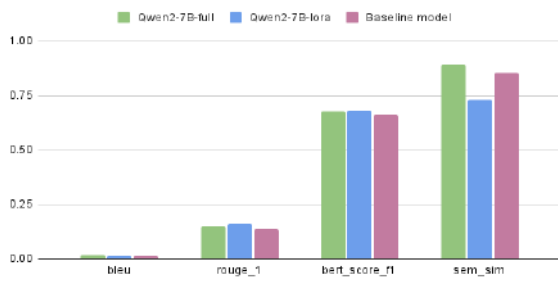


Figure 4. Metrics for QA long answer task

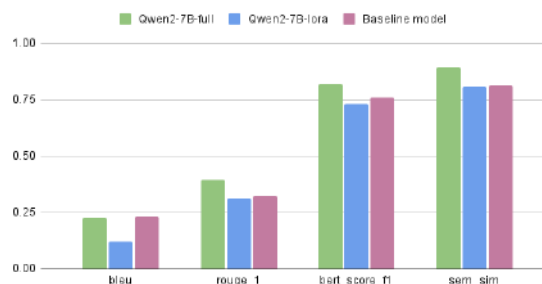


Figure 5. Metrics for QA long short task

4. Related work

4.1 Data

Several studies have been conducted in which authors have developed datasets for similar tasks. The paper [11] introduces the EconQA dataset, designed to assess the performance of LLMs in economics through multiple-choice questions. The authors present the results of ten experiments that varied both the prompts and model variants. Findings from these experiments indicate that the choice of prompt significantly influences the quality of the responses. The dataset was compiled using questions from the test bank of "McConnell and Brue Economics, 16th edition." Initially, this test bank contained approximately 6,000 questions, but after preprocessing, the number was reduced to 3,623.

The paper [12] presents the EconLogicQA benchmark, which is designed to assess the sequential reasoning capabilities of LLMs in the fields of economics, business, and supply chain management. EconLogicQA poses a more challenging problem: it requires models to distinguish and organize multiple interrelated events, reflecting the complexity of economic logic. The dataset includes a variety of scenarios extracted from economic articles that necessitate a deep understanding of both temporal and logical relationships between events. EconLogicQA effectively evaluates undergraduate students' ability to navigate the sequential complexities inherent in economic contexts. To streamline the question generation process and reduce subjectivity, labor intensity, and randomness associated with manual generation, the authors utilize GPT-4 to automatically generate questions by extracting key points from news articles.

Additionally, there are platforms such as Economiga [13], which allow users to engage in dialogue in a question-and-answer format concerning economic issues. Another example of economic data is "Explanations for CommonsenseQA: Dataset" [14].

4.2 Research on LoRA and full fine-tuning

In the study [15], researchers compared full parameter fine-tuning and LoRA-based fine-tuning that shows significant advantages in terms of training costs. The authors conducted an experimental comparison of full fine-tuning and LoRA-based fine-tuning methods using LLaMA as the base model. The results of the experiment showed that the choice of the base model, the scale of the training dataset, the number of trainable parameters, and the cost of model training are important factors.

Due to their learning ability, LLMs can generate a large number of diverse instructions. The authors of the paper translated the original data into Chinese and modified certain elements to better align with Chinese culture and context. Then, using these original data as contextual examples, prompts were created for ChatGPT to generate more examples.

To compare the quality of the models, the authors trained two models using full fine-tuning on training data of 0.6M and 2M instructions. The results demonstrate that full fine-tuning gives better experimental results than LoRA

To assess the quality of the models, the authors trained two models using full fine-tuning on datasets containing 0.6M and 2M instructions. The findings indicate that full fine-tuning yields superior experimental results compared to LoRA. In their study [16], the authors drew the following conclusions regarding the comparison between full fine-tuning and LoRA:

- 1) Full fine-tuning outperforms LoRA in terms of accuracy and effectiveness, particularly in programming and mathematics domains.
- 2) LoRA exhibits less forgetting of the original domain, providing a form of regularization.
- 3) The regularization effect of LoRA is stronger than that of typical regularization methods and contributes to maintaining diversity.

In another work [17] the authors claim that full fine-tuning (FPFT) has become the main choice for adapting language models to problems due to its excellent performance. As the LM size grows, fine-tuning all the LM parameters requires prohibitive amounts of GPU memory. Existing approaches use zero-order optimizers to save GPU memory, which potentially reduces LM performance since non-zero-order optimizers tend to converge faster on most subsequent problems.

The researchers propose a new memory-efficient, optimizer-independent approach, HiFT, which updates only a subset of parameters at each training step. HiFT significantly reduces the number of gradients and optimizer state parameters that reside in GPU memory at the same time, thereby reducing GPU memory consumption.

5. Conclusion

In conclusion, a comparison of the full parameter fine-tuning and LoRA approaches was conducted. Full parameter fine-tuning achieves superior results, as it allows all model parameters to be adapted to specific tasks, making the model versatile enough to address a wide range of problems. However, this approach comes with significant drawbacks, including high computational costs, the need for powerful GPUs, and extended training times, particularly for large models. Additionally, it is prone to overfitting on small datasets, which can diminish its generalization capabilities. After further training, the model can occupy considerable space, complicating deployment and usage.

In contrast, LoRA [16] requires fewer computational resources and less memory which allows using less powerful GPUs. The process of training is faster due to the reduced number of parameters which need updating. LoRA checkpoints are significantly smaller in size which makes storage and deployment easier.

In our study the loss of accuracy is observed compared to full parameter fine-tuning. The effectiveness of LoRA highly depends on training data quality and quantity. The research showed that full parameter fine-tuning leads to higher accuracy rates compared to LoRA on our classification and QA short and detailed answers tasks. For the classification task, the average accuracy rate for full fine-tuning was 0.85 and LoRA achieved 0.7. However, LoRA helps to reduce the time of additional training by 2 to 5 times compared to full parameter fine-tuning.

We also tested the hypothesis that different types of tasks from the same domain increase the quality of generation; in our case, combining training data for the classification task and QA led to a decrease to the level of LoRA metrics.

The choice between full fine-tuning and LoRA depends on the specific requirements of the task, the available computing resources, and the desired level of accuracy. Full fine-tuning is ideal for tasks that require maximum accuracy, while LoRA is an excellent choice for resource and time-constrained scenarios.

References

1. Han Z., Gao C., Liu J. et al. Parameter-efficient fine-tuning for large models: A comprehensive survey // CoRR. 2024. Vol. abs/2403.14608. arXiv: 2403.14608 [cs.LG]. DOI: 10.48550/arXiv.2403.14608.
2. Yang A., Yang B., Zhang B. et al. Qwen2.5 Technical Report // CoRR. 2024. Vol. abs/2412.15115. arXiv: 2412.15115 [cs.CL]. DOI: 10.48550/arXiv.2412.15115.

3. Hu E.J., Shen Y., Wallis P. et al. LoRA: Low-rank adaptation of large language models // CoRR. 2021. Vol. abs/2106.09685. arXiv: 2106.09685 [cs.CL]. DOI: 10.48550/arXiv.2106.09685.
4. Achiam J., Adler S., Agarwal S. et al. GPT-4 technical report // CoRR. 2023. Vol. abs/2303.08774. arXiv: 2303.08774 [cs.CL]. DOI: 10.48550/arXiv.2303.08774.
5. Stack Exchange. URL: <https://stackexchange.com/> (accessed: 16.01.2025)
6. Qwen2-7B. URL: <https://huggingface.co/Qwen/Qwen2-7B> (accessed: 16.01.2025)
7. Zhang T., Kishore V., Wu F. et al. BERTScore: Evaluating Text Generation with BERT // CoRR. 2019. Vol. abs/1904.09675. arXiv: 1904.09675 [cs.CL]. DOI: 10.48550/arXiv.1904.09675.
8. Papineni K., Roukos S., Ward T., Zhu W.-J. BLEU: a Method for Automatic Evaluation of Machine Translation // Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. 2002. P. 311–318. DOI: 10.3115/1073083.1073135.
9. Lin C.Y. Rouge: A Package for Automatic Evaluation of Summaries // Text Summarization Branches Out. 2004. P. 74–81.
10. Chandrasekaran D., Mago V. Evolution of Semantic Similarity - a Survey // ACM Computing Surveys (CSUR). 2021. Vol. 54, no. 2. P. 1–37. DOI: 10.1145/3440755.
11. van Patten T. Evaluating Domain Specific LLM Performance Within Economics Using the Novel EconQA Dataset // WWU Honors College Senior Projects. 2023.
12. Quan Y., Liu Z. EconLogicQA: A Question-Answering Benchmark for Evaluating Large Language Models in Economic Sequential Reasoning // CoRR. 2024. Vol. abs/2405.07938. arXiv: 2405.07938 [cs.CL]. DOI: 10.18653/v1/2024.findings-emnlp.125.
13. Economiga Question Bank. URL: <https://www.economiga.org/qbank> (accessed: 16.01.2025)
14. Aggarwal S., Mandowara D., Agrawal V. et al. Explanations for CommonsenseQA: New Dataset and Models // Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2021. P. 3050–3065. DOI: 10.18653/v1/2021.acl-long.238.
15. Sun X., Ji Yu., Ma B., Li X. A Comparative Study Between Full-Parameter and LoRA-based Fine-Tuning on Chinese Instruction Data for Instruction Following Large Language Model // CoRR. 2023. Vol. abs/2304.08109. arXiv: 2304.08109 [cs.CL]. DOI: 10.48550/arXiv.2304.08109.
16. Biderman D., Portes J., Ortiz J.J.G. et al. LoRA Learns Less and Forgets Less // CoRR. 2024. Vol. abs/2405.09673. arXiv: 2405.09673 [cs.LG]. DOI: 10.48550/arXiv.2405.09673.
17. Liu Y., Zhang Yi., Li Q. et al. Hift: A Hierarchical Full Parameter Fine-Tuning Strategy // CoRR. 2024. Vol. abs.2401.15207. arXiv: 2401.15207 [cs.LG]. DOI: 10.18653/v1/2024.emnlp-main.1015.

Enhancing Admission Inquiry Responses with Fine-Tuned Models and Retrieval-Augmented Generation

A. Virabyan

HSE University, Moscow, Russia

University admissions offices face the significant challenge of managing high volumes of inquiries efficiently while maintaining response quality, which critically impacts prospective students' perceptions. This paper addresses the issues of response time and information accuracy by proposing an AI system integrating a fine-tuned language model with Retrieval-Augmented Generation (RAG). While RAG effectively retrieves relevant information from large datasets, its performance in narrow, complex domains like university admissions can be limited without adaptation, potentially leading to contextually inadequate responses due to the intricate rules and specific details involved. To overcome this, we fine-tuned the model on a curated dataset specific to admissions processes, enhancing its ability to interpret RAG-provided data accurately and generate domain-relevant outputs. This hybrid approach leverages RAG's ability to access up-to-date information and fine-tuning's capacity to embed nuanced domain understanding. We further explored optimization strategies for the response generation logic, experimenting with settings to balance response quality and speed, aiming for consistently high-quality outputs that meet the specific requirements of admissions communications.

Keywords: admissions inquiry optimization, fine-tuning language models, retrieval-augmented generation (RAG), AI for admissions support, high-volume inquiry management, response accuracy, natural language processing (NLP), machine learning in education, admission process automation, large language models (LLMs).

1. Introduction

University admissions offices receive a vast number of inquiries from applicants daily. The quality of responses to these inquiries plays a key role in forming the first impression of the university. Responses from the admissions office are often the first interaction of a prospective student with the university. According to research [1], customer satisfaction directly affects retention, word-of-mouth, and switching intentions. Therefore, it is essential that each response be of the highest quality and professionalism. This task is challenging initially and becomes increasingly difficult as inquiry volume grows, often forcing staff to rush and dedicate less time per response.

Over the past couple of years, due to the explosion of generative artificial intelligence, more and more use cases are leaning towards large language models (LLMs). Large language models can potentially generate draft responses that admissions staff can review, edit if necessary, and send the response. While simpler rule-based chat bots and similar systems have seen application in educational contexts, they typically lack the flexibility and deep natural language understanding necessary to handle the diverse and complex inquiries common to university admissions offices. This challenge is particularly pronounced in environments like Russia, where admission rules are intricate, heavily regulated by the government, and filled with nuances that often overwhelm basic systems. Our approach, leveraging recent advancements in LLMs through the combination of retrieval-augmented generation and fine-tuning, offers a substantially more robust and scalable solution compared to these earlier methods or reliance on basic prompt engineering alone.

During the admission campaign, the admissions office of the Higher School of Economics in

Russia receives a large volume of inquiries – more than 2,000 per day. This volume often exceeds the capacity of the staff, resulting in a significant increase in response time. Therefore, the time available for each response is shortened, which can prevent staff from providing comprehensive and detailed answers, potentially hindering their ability to fully demonstrate institutional interest in the applicant.

Large language models can help admissions offices improve the quality of responses to applicants through automation. However, a limitation of mass language models from providers such as ChatGPT, Yandex GPT, GigaChat, and others is that they are not specifically trained on university admissions policies. Consequently, these models lack a comprehensive understanding of university admissions procedures and policies characteristic of experienced admissions staff. For the model to generate high-quality responses, it must develop knowledge and analytical capabilities related to university admissions campaigns. To embed domain-specific knowledge and develop analytical skills in the context of university admissions, we propose a hybrid approach combining Retrieval-Augmented Generation (RAG) with Large Language Model (LLM) fine-tuning. While many studies show that fine-tuning alone is not sufficient for effective knowledge adoption – for example, a [2] study concludes that RAG outperforms unsupervised fine-tuning in knowledge-related tasks – RAG has a limitation. In general, RAG functions as a search engine, and its performance can be highly dependent on the specific wording of user queries. This dependency can sometimes lead to suboptimal information retrieval.

2. Related Work and Theoretical Background

2.1. Retrieval-Augmented Generation (RAG)

In the context of university admissions, RAG has the potential to significantly improve response accuracy. As admissions-related inquiries are highly specific and varied, relying solely on pre-trained models can lead to vague, generalized, or even misleading responses (e.g., confusing information between universities). RAG's ability to fetch domain-specific documents, such as admissions rules, educational program catalogs, and other relevant documents and data such as specific deadlines, enables the system to generate highly tailored responses. Given the complexity and variability of the admissions process, where details such as application deadlines, program prerequisites, and eligibility criteria are key, RAG becomes particularly effective. By drawing from these specialized documents, the model can generate answers that are not generic, but instead directly relevant to the specific needs of prospective students or applicants. This is a significant advantage in environments where precision and correctness are paramount.

A recent study [3] has indicated that Retrieval-Augmented Generation (RAG) has emerged as a promising paradigm to address the aforementioned challenges. Specifically, RAG introduces an information retrieval process, which has been found to enhance the generation process by retrieving relevant information/documents from available data stores. This, in turn, has led to higher accuracy and greater robustness.

While various RAG implementations exist, in general, RAG functions by retrieving relevant documents from external knowledge bases, providing accurate, real-time, domain-specific context to LLMs [4].

2.2. Fine-Tuning in AI Models

Fine-tuning has become a fundamental technique in the optimization of AI models, particularly when applying general-purpose models to domain-specific tasks. At its core, fine-tuning involves taking a pre-trained model, which has learned general patterns from large datasets, and further training it on a smaller, domain-specific dataset. Recent research [6] implies that additional training allows the model to adapt its knowledge to better handle specialized terminology, nuances, and context relevant to the specific domain, improving its performance in tasks such

as generating responses to admissions inquiries.

Fine-tuning is particularly beneficial in situations where a model's pre-existing knowledge, obtained from a broad and general dataset, may not be sufficient to understand the intricacies of specialized areas. For example, in the context of university admissions, the model must understand complex terminology and procedures. These terms may not be sufficiently represented in general corpora, but they are essential for accurate and relevant responses in an admissions context. Through fine-tuning, the model can be retrained to focus on this specialized vocabulary, improving its ability to generate precise and contextually appropriate answers.

Research has shown that fine-tuning can lead to substantial improvements in both accuracy and relevance. For instance, Radford et al. [7] demonstrated that fine-tuning GPT models on specialized data significantly enhanced their ability to generate relevant, domain-specific text. Similarly, studies in the medical and legal fields have shown that fine-tuned models outperform generic models in understanding and responding to specialized queries [2]. The improvements observed in these domains are directly applicable to the university admissions process, where accurate, timely, and contextually aware responses are essential.

Moreover, fine-tuning is often an iterative process. It requires careful selection of a representative dataset and continuous evaluation to ensure that the model's performance improves over time. In our case, the dataset would consist of a comprehensive collection of admissions-related documents, including policies, program descriptions, application guidelines, and anonymized previous student inquiries. By fine-tuning the model on this dataset, it becomes more adept at generating responses that align with the institution's specific requirements and regulations.

2.3. Combining RAG with Fine-Tuning

The combination of RAG with fine-tuning offers the potential to significantly improve response speed and accuracy. By retrieving relevant documents in real-time, the model ensures that the information provided is up-to-date and tailored to the query. Fine-tuning further enhances this process by allowing the model to generate responses that are not only based on the retrieved data but also reflect the specific language and protocols of the admissions process. This results in responses that are not only fast but also precise, reducing the need for manual intervention and follow-up inquiries.

Furthermore, fine-tuning can help address one of the key limitations of RAG: the need to interpret and synthesize large volumes of information. In a typical RAG implementation, the retrieved documents can be lengthy or contain multiple pieces of information that need to be distilled into a concise and coherent response. Fine-tuning enables the model to better understand how to prioritize and structure this information, ensuring that the response is clear and relevant to the specific needs of the query.

In practice, the integration of RAG and fine-tuning requires careful calibration. The model must be fine-tuned on a diverse dataset that reflects the variety of inquiries encountered in the admissions process. This may include questions related to specific programs, application procedures, deadlines, and eligibility criteria. The fine-tuning process must also be iterative, with continuous evaluation to ensure that the model is learning effectively and that the quality of its responses improves over time. Additionally, regular updates to the knowledge base may be necessary to ensure that the system remains current with changes in admissions policies and procedures.

The combination of RAG with fine-tuning also enables the model to handle a broader range of inquiries. While RAG helps the system retrieve relevant data, fine-tuning ensures that the system is equipped to handle complex, domain-specific queries that may not be fully addressed by a general-purpose language model. For example, a prospective student may inquire about specific requirements for an international transfer student, which requires knowledge of both program prerequisites and visa policies. Fine-tuning on a dataset that includes these specifics allows the system to generate a response that is both accurate and comprehensive.

3. Methodology

3.1. Data Collection & Preprocessing

The data collection process involves the collection of data related to university admissions, including official documents, data from the admissions website, and previous staff responses (often in a question-and-answer format) extracted from ticketing systems. The collected data is then verified for quality and processed. This includes cleaning the data, removing irrelevant and sensitive information, and normalizing text for consistency. The processed data is stored in a structured format for efficient retrieval and is used to provide a basis for model training.

The data collection process involves the collection of data related to university admissions, including official documents such as admissions regulatory documents and data from official Q/A pages on the website. The collected data is then verified for quality and processed, which is achieved by fact-checking done by admissions staff. This includes cleaning the data, removing irrelevant and sensitive information, and normalizing text for consistency. The knowledge base is formed with the collected data described earlier.

At the same time, directly using these raw source materials, especially for fine-tuning, presents significant challenges. For example, official regulatory documents often have complicated structures; maybe Section 4.5 refers back to a definition in Section 2.9, which is hard for a model to follow directly. Web pages can also be messy, with inconsistent formatting or navigation elements mixed in. Trying to teach the model directly from this kind of unstructured, complex data is not very efficient and might lead to weird hallucinations and mistakes in understanding; it can struggle to extract the core information or understand the relationships between different pieces of text.

To address this problem and create a cleaner and more efficient dataset for fine-tuning, we used a data distillation process. We used DeepSeek R1 LLM model [8], a large language model characterized by its ability to handle extensive context. We provided DeepSeek R1 with collected source documents – regulations, web content, FAQs – as context. We then asked the model to generate relevant question-answer pairs based only on the information contained in these particular documents.

This distillation procedure resulted in a structured Alpaca-formatted dataset containing about [8,000] high quality pairs. This effectively transformed the complex, unstructured raw information into clear question and answer pairs that are much more suitable for controlled fine-tuning than the original raw documents. The raw documents themselves were indexed and used primarily for the searchable database of the RAG component.

3.2. Model Architecture

For this project, we leveraged a RAG architecture, which combines the power of a large pre-trained language model with an information retrieval system. The architecture consists of two primary components: the retriever and the generator.

A) Retriever: The retriever is responsible for fetching relevant documents or passages from an external knowledge base. In our case, the knowledge base consists of the domain-specific admissions documents collected during the data collection phase. The retriever uses an embedding-based similarity search approach (specifically, BERT embeddings) to find the most relevant documents that may contain the answers to a given query.

Documents in the knowledge base were segmented into chunks of [512] tokens with an overlap of [64] tokens to ensure semantic continuity. During retrieval, the [top 3] most relevant document chunks based on cosine similarity with the query embedding were selected.

B) Generator: The generator is a fine-tuned version of a selected pre-trained language model (Gemma 9B model [9] as the base one), adapted using [Low-Rank Adaptation (LoRA)] for [3] epochs on our distilled dataset, responsible for generating natural language responses based on

the information retrieved by the retriever. The generator's job is to process the input request and relevant documents provided by the retriever, synthesizing this information into a consistent, accurate, and context-aware response.

4. Experimental Setup and Evaluation

4.1. Experiment Design

The experiment is structured to compare four different response generation approaches:

1. **Baseline GPT Model:** A standard pre-trained language model without fine-tuning or RAG.
2. **RAG Model:** A model incorporating retrieval-augmented generation but without fine-tuning.
3. **Fine-tuned model:** A model fine-tuned on domain-specific data but *without* RAG.
4. **Fine-tuned Model with RAG:** Our proposed approach, combining RAG with domain-specific fine-tuning.

The dataset used for evaluation consists of 210 university admissions inquiries collected from historical records. Each model generated responses, which were then manually assessed for accuracy and completeness. Selected inquiries were sampled from historical records to represent a diverse range of common topics, including application procedures, deadlines, document requirements, international student queries, and program-specific questions. Manual assessment was conducted by two team members following predefined guidelines focusing on factual correctness, completeness, relevance, and adherence to university tone. The main factor was whether or not the reviewing admissions officer would have sent the generated response to the author of the appeal (where 0 – would not have sent in any form, 1 – would have sent with some edits, 2 – would have sent without edits). To determine the quality of the completed assessment, inter-rater reliability score is calculated, resulting in: Cohen's Kappa > 0.8 .

4.2. Evaluation Metrics

We use the following metrics to evaluate system performance:

- **Fact Recall (%)**: Measures the percentage of correct factual statements retrieved and used in the response.
- **Precise Data Recall (%)**: Focuses on the accuracy of responses that require specific data points such as dates and URLs.
- **User Satisfaction Score**: A subjective rating (1–10) collected from a survey of admissions employees interacting with the system.

4.3. Results and Discussion

The table below presents a comparison of the four models based on the defined metrics.

These findings emphasize the importance of combining several approaches: our results show that integrating fine-tuning with RAG significantly improves response accuracy and relevance in university admissions. The fine-tuned RAG model demonstrated strong overall performance, achieving the highest fact recall (92.7%) and user satisfaction (8.9). While the standard RAG model achieved the highest precise data recall (91.4%), the fine-tuned RAG model also performed well on this metric (88.3%) and significantly outperformed the baseline GPT and fine-tuned only

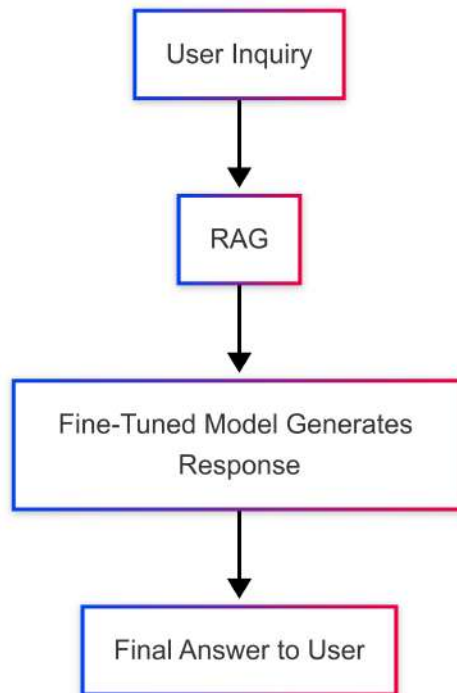


Figure 1. The flow of the proposed approach

Table 1. Performance Comparison of Response Generation Models

Model	Fact Recall (%)	Precise Data Recall (%)	User Satisfaction (1–10)
Baseline GPT	22.3	25.6	3.2
RAG Model	75.1	91.4	7.5
Fine-Tuned (No RAG)	72.7	48.3	7.9
Fine-Tuned with RAG	92.7	88.3	8.9

models across most metrics. While the fine-tuned only model performed better than the baseline GPT (particularly in fact recall), its precise data recall was significantly lower than the RAG-enhanced approaches, highlighting a key weakness. This highlights the necessity of both retrieval augmentation and domain-specific fine-tuning for optimal results. The superior performance of the combined approach likely stems from RAG’s ability to inject precise, up-to-date facts, while fine-tuning equips the model to better understand the context of admissions queries, synthesize retrieved information effectively, and adhere to the institution’s specific communication style. The lower precise data recall of the fine-tuned only model suggests that while fine-tuning improves domain understanding, it may struggle to reliably memorize and recall volatile details like specific dates or URLs without a retrieval mechanism. Although highly effective, occasional errors observed in the fine-tuned RAG model involved misinterpreting highly nuanced queries or rare edge cases not well represented in the training data.

5. Challenges and Future Directions

5.1. Current Limitations

While our approach has shown improvements, some challenges remain. Fine-tuning large models requires significant computational resources, potentially making deployment costly for smaller institutions. Loading a fine-tuned model into memory often necessitates dedicated hardware (e.g., a server with sufficient RAM/GPU memory) for each specific application domain, such as admissions. For example, if the solution is planned to be deployed across different admissions departments such as undergraduate and graduate, two different fine-tuned models might be required as the two different models need to be fine-tuned separately. Beyond cost, limitations include the residual risk of hallucination (though mitigated by RAG), challenges in handling truly ambiguous or novel questions outside the scope of the knowledge base, and the potential for inheriting biases present in the training or source data. Ensuring the system gracefully handles out-of-domain queries remains an area for attention.

5.2. Potential Improvements

Several improvements can be considered to further optimize the model. Firstly, the integration of a dynamic knowledge update mechanism can ensure that the information retrieved is up-to-date. Implementing this would likely require a more complex system, potentially including a user interface for admissions staff to review draft responses and easily update the internal knowledge base used by the RAG component.

This dynamic updating capability could also mitigate the need for frequent, resource-intensive fine-tuning cycles, especially for rapidly changing information like deadlines or specific policy details. Broad, stable knowledge could be embedded during fine-tuning, while precise, volatile data (like specific dates or URLs) would be best handled by the RAG component. Our results support this, showing RAG's strength in recalling precise data compared to the fine-tuned-only model.

5.3. Broader Implications

The successful implementation of fine-tuned RAG models in admissions could inspire broader applications in other domains. Similar techniques could be used in customer service, legal consultations, and healthcare inquiries where accuracy and timely responses are critical. Additionally, improving AI-driven communication tools could reduce the burden on human operators, allowing them to focus on more nuanced cases requiring direct human intervention.

6. Conclusion

This paper addressed the challenge faced by university admissions offices in managing high volumes of inquiries while maintaining response quality and timeliness. The core problem lies in providing accurate, context-aware, and personalized information efficiently, which is crucial for prospective student engagement but difficult under heavy workload. We proposed a hybrid solution combining Retrieval-Augmented Generation (RAG) with domain-specific fine-tuning of a large language model. RAG provides access to current, relevant information from institutional knowledge bases, while fine-tuning adapts the model to understand the nuances, terminology, and specific communication style required in the admissions context.

Our experiments compared this combined approach against baseline models, RAG-only models, and fine-tuned-only models. The results clearly demonstrated the superiority of the integrated fine-tuned RAG system. It achieved the highest overall performance, notably scoring 92.7% in fact recall and 8.9/10 in user satisfaction, significantly outperforming other configura-

tions. While the RAG-only model excelled in precise data recall (91.4%), the fine-tuned RAG model was comparable (88.3%) and offered better contextual understanding and response formulation, as reflected in user satisfaction. The fine-tuned-only model struggled with recalling precise, volatile data (48.3% recall), highlighting the necessity of RAG for factual grounding. These findings underscore that combining the strengths of RAG (accessing specific facts) and fine-tuning (understanding domain context and style) is crucial for developing effective AI tools for complex, knowledge-intensive tasks like handling admissions inquiries.

References

1. Athanassopoulos A., Gounaris S., Stathakopoulos V. Behavioural responses to customer satisfaction: an empirical study // *European Journal of Marketing*. 2001. Vol. 35, no. 5/6. P. 687–707. DOI: 10.1108/03090560110388169.
2. Ovadia O., Brief M., Mishaeli M., Elisha O. Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs // *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP 2024)*. Association for Computational Linguistics, 2024. P. 237–250. DOI: 10.18653/v1/2024.emnlp-main.15.
3. Zhao P., Zhang H., Yu Q. et al. Retrieval-augmented generation for AI-generated content: A survey // *CoRR*. 2024. Vol. abs/2402.19473. arXiv: 2402.19473 [cs.CV]. DOI: 10.48550/arXiv.2402.19473.
4. Gao Y., Xiong Y., Gao X. et al. Retrieval-augmented generation for large language models: A survey // *CoRR*. 2023. Vol. abs/2312.10997. arXiv: 2312.10997 [cs.CL]. DOI: 10.48550/arXiv.2312.10997.
5. Wang X., Wang Z., Gao X. et al. Searching for Best Practices in Retrieval-Augmented Generation // *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP 2024)*. Association for Computational Linguistics, 2024. P. 17716–17736. DOI: 10.18653/v1/2024.emnlp-main.981.
6. Mecklenburg N., Lin Y., Li X. et al. Injecting new knowledge into large language models via supervised fine-tuning // *CoRR*. 2024. Vol. abs/2404.00213. arXiv: 2404.00213 [cs.CL]. DOI: 10.48550/arXiv.2404.00213.
7. Radford A., Narasimhan K., Salimans T., Sutskever I. Improving language understanding by generative pre-training. OpenAI, 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.
8. DeepSeek-AI, Guo D., Yang D. et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning // *CoRR*. 2025. Vol. abs/2501.12948. arXiv: 2501.12948 [cs.CL]. DOI: 10.48550/arXiv.2501.12948.
9. Gemma Team: Rivière M., Pathak S., Sessa P.G. et al. Gemma 2: Improving open language models at a practical size // *CoRR*. 2024. Vol. abs/2408.00118. arXiv: 2408.00118 [cs.CL]. DOI: <https://arxiv.org/abs/2408.00118>.

Эффективный предобуславливатель для решения систем линейных уравнений, возникающих в задаче фильтрации в деформируемой поровой среде

О.С. Борщук

ООО «РН-Технологии», ФГБОУ ВО «Уфимский университет науки и технологий»

В работе исследуется эффективность совместного применения предобуславливателей типа ILU и AIPS для решения систем линейных уравнений специального вида в задачах фильтрации флюидов в пористой среде, учитывающих изменение напряженно-деформированного состояния породы. Показано, что использование адаптированного под задачу предобуславливателя до 2.8 раз ускоряет решение системы по сравнению со стандартными методами решения, используемыми в пакетах общего назначения. Приведены численные результаты распараллеливания полученного метода решения на многоядерные системы.

Ключевые слова: итерационные методы, предобуславливатели на основе неполного LU разложения, предобуславливатели на основе степенного разложения обратной матрицы AIPS, безматричное умножение.

1. Введение

В процессе решения задач фильтрации флюидов в пористой среде с учетом изменения напряженно-деформированного состояния породы возникает необходимость решения большого количества систем линейных алгебраических уравнений (СЛАУ) специального вида:

$$\begin{bmatrix} F & E \\ C & B \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}, \quad (1)$$

где E – единичная матрица, F – заполненная матрица, связанная с уравнениями геомеханики, C и B – разреженные матрицы с одинаковым шаблоном разреженности, т.е. ненулевые элементы находятся в одних и тех же местах. C и B отвечают за уравнения гидродинамики. Блоки в системе (1) обладает следующими свойствами:

- F – строится один раз и не меняется в процессе эволюционного расчета,
- C и B – меняются на каждом шаге по времени и на каждой итерации нелинейного решателя, но шаблон разреженности остается постоянным,
- f_{ij} элементы матрицы F , представимы в следующем виде:

$$f_{ij} = W(|i_x - j_x|, |i_y - j_y|) G(i_y, j_y) R(i_y), \quad (2)$$

где W, G, R – некоторые известные функции, i_x, i_y – координаты узла i в расчетной сетке.

Из первой строки системы (1) следует

$$v = -Fu. \quad (3)$$

Подставляя (3) во второе уравнение системы (1) получаем СЛАУ на неизвестный вектор u :

$$(C - BF)u = b. \quad (4)$$

Отметим, что практически все итерационные методы решения систем линейных уравнений базируются на операции умножения матрицы на вектор. Поэтому можно не вычислять матрицу $C - BF$, а последовательно вычислять: 1. $q = Cu$; 2. $w = Fu$; 3. $(C - BF)u \equiv q - Bw$. Для удобства обозначим $A = C - BF$.

Далее везде для построения матрицы предобуславливателя на основе неполного LU разложения будем использовать матрицу:

$$\tilde{A} = C - B\tilde{F}, \quad (5)$$

где $\tilde{F}$ – разряженная матрица, полученная из F отбрасыванием элементов f_{ij} , $i \neq j$, таких, что $|f_{ij}| < \varepsilon |f_{ii}|$, $\varepsilon \in (0, 1)$ – параметр, выбираемый из условия минимизации времени расчета.

В разделе 2 описаны методы предобуславливания, наиболее часто используемые для систем общего вида, и аспекты их параллельной реализации. В разделе 3 описан метод уточнения предобуславливателя на основе степенного разложения обратной матрицы, описаны принципы совмещения предобуславливателей типа ILU и AIPS и возможные сценарии их использования. В разделе 4 описан метод безматричного умножения матрицы F на вектор, показаны пути оптимизации под современные процессоры с архитектурой x86 и ARM. В разделе 5 представлены результаты численных экспериментов и сравнение эффективности предобуславливателей, в том числе в зависимости от количества потоков при расчетах на многоядерных процессорах. Численные результаты показывают, что совместное использование предобуславливателей типа неполного LU разложения для диагональных блоков без перекрытия и степенного разложения обратной матрицы AIPS значительно улучшает сходимость итерационных линейных решателей и позволяет ускорить решение системы (1) до 2.8 раз по сравнению с наиболее часто используемыми универсальными подходами.

2. Предобуславливатели на основе неполного LU разложения для параллельных вычислений

В качестве одной из составных частей предобуславливателей будем использовать матрицы, основанные на неполном LU разложении:

- ILU0 (Incomplete LU factorization with zero fill) – неполное LU разложение без расширения «шаблона разреженности». Выполняется LU-разложение, но при этом отбрасываются все «порождаемые» ненулевые элементы (т.е. остаются только те ненулевые элементы, которые расположены в тех же позициях, где были ненулевые элементы исходной матрицы) [1, 2]. Самый простой и быстрый в построении предобуславливатель в классе ILU, но на сложных задачах может не обеспечивать нужной скорости сходимости либо не приводить к сходимости вообще.
- ILUT (Incomplete LU with Threshold) – факторизация строится на основе числового порога (threshold), который определяет какие ненулевые элементы можно отбросить после вычисления значений в элементах L и U [1, 3].
- BILU0/BILUT (Block ILU0/ILUT) – блочно диагональные варианты неполного LU разложения. Используются для параллельной реализации ILU0/ILUT (рис. 1а), при этом обмены между блоками отсутствуют либо минимальны, поэтому хорошо распараллеливается, но сходимость может значительно ухудшаться с ростом числа исполнительных потоков [1, 4].
- OBILU0/OBILUT (Overlaped Block ILU0/ILUT) – блочно диагональные варианты неполного LU разложения с перекрытием. Размер перекрытия выбирается из условия минимизации влияния элементов матрицы вне перекрытия на внутренние элементы

блока (рис. 16) [5]. Позволяет повысить качество предобуславливателя (меньше итераций требуется для получения решения), но чем больше зона перекрытия, тем сильнее растет сложность построения предобуславливателя.

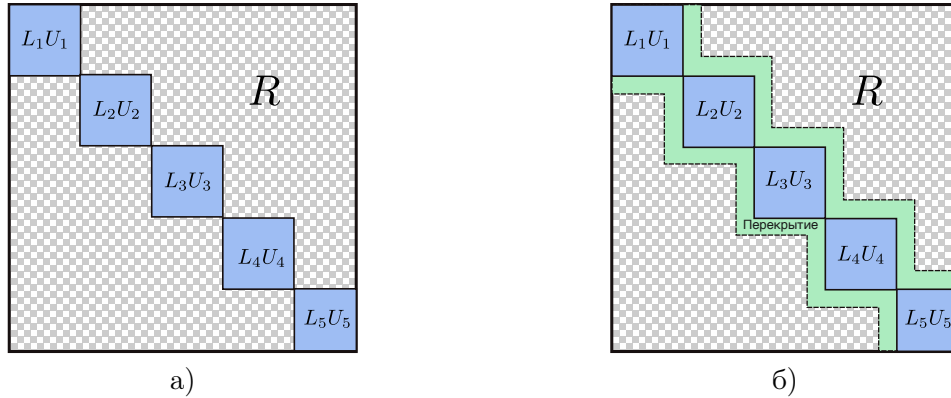


Рис. 1. Примеры декомпозиции матрицы для параллельной ILU факторизации: а) в ILU разложении участвуют диагональные блоки без перекрытия; б) в ILU разложении участвуют блоки большего размера с перекрытием, все блоки раскладываются независимо, но решение берется только по внутренним частям блоков

3. Уточнение предобуславливателя на основе степенного разложения обратной матрицы

Для улучшения сходимости итерационных методов решения СЛАУ (1) и уточнения предобуславливателей типа ILU можно использовать разложение обратной матрицы в ряд Неймана [1, 6]:

$$(E - B)^{-1} = E + B + B^2 + \dots + B^n + \dots, \quad \rho(B) < 1, \quad (6)$$

где E – единичная матрица, а $\rho(B)$ – спектральный радиус матрицы B .

Стандартным является представление матрицы A в виде [7]:

$$A = P + R = P(E + P^{-1}R) = P(E - (-P^{-1}R)), \quad (7)$$

где P – матрица предобуславливателя, R – матрица ошибки. При выполнении условия $\rho(P^{-1}R) < 1$ можно получить следующую формулу обращения предобуславливателя M_N порядка N :

$$M_N^{-1} = \left[E + \sum_{k=1}^N (-1)^k (P^{-1}R)^k \right] P^{-1}. \quad (8)$$

Вычисление M_N^{-1} тем сложнее, чем больше N , но с ростом N улучшается качество предобуславливания $A^{-1} \approx M_N^{-1}$ и снижается количество необходимых для решения линейной системы итераций. В общем случае выбор параметра N проводится эмпирически для каждого типа решаемых задач отдельно. Предобуславливатель M_N , определяемый формулой (8), принято называть AIPS (Approximation of Inverse by Power Series).

Формула (8) удобна, когда матрица R легко определима; если же R неизвестна, то может быть заменена $R = A - P$. Формула (8) в этом случае примет вид:

$$\tilde{M}_N^{-1} = \left[E + \sum_{k=1}^N Q^k \right] P^{-1}, \quad Q = E - P^{-1}A. \quad (9)$$

Если матрица предобуславливателя P построена на малой части ненулевых элементов матрицы A , например, используются три главные диагонали [7], то разница в сложности вычислений между формулами (8) и (9) не значительна, матрица R меньше матрицы A на 3 диагонали. С другой стороны, при неполном LU разложении [2] для получения матрицы R необходимо умножить матрицы $L \cdot U$, что трудоемко, при этом в матрице R может оказаться значительно больше ненулевых элементов, чем в матрице A ; в этом случае выгоднее использовать формулу (9). Предобуславливатель, полученный по формуле (9), будем называть AIPS2.

Отметим, что алгоритмы построения предобуславливателей AIPS и AIPS2 хорошо подходят для использования на параллельных процессорах, все вычислительные потоки работают независимо, пересылки данных между потоками минимальны.

Большим потенциалом обладает совмещение неполного LU разложения с уточнением предобуславливателя по формулам (8), (9). Построение и обращение предобуславливателей типа ILU существенно последовательны, а их реализации для параллельных вычислений алгебраически не эквивалентны последовательным алгоритмам и требуют больше итераций линейного решателя для достижения сходимости. При этом сходимость линейного решателя ухудшается с ростом количества вычислительных потоков. Определим эффективность параллельной блочной реализации предобуславливателей типа ILU с уточнением AIPS для компенсации ухудшения свойств блочных предобуславливателей типа ILU при росте числа вычислительных потоков.

4. «Безматричное» умножение матрицы F на вектор

Поскольку блок-матрица F плотно заполненная, то при росте размерности сложность умножения матрицы F на вектор растет как $O(n^2)$, где n – размерность матрицы. Остальные блок-матрицы – разреженные с ограниченным количеством ненулевых элементов в строке, и сложность их умножения на вектор растет как $O(n)$. Соответственно, с ростом размерности операции с матрицей F становятся доминирующими и требуют особого внимания. В литературе и пакетах программ известны «безматричные» подходы к умножению матрицы на вектор [8, 9], в этом случае формирование матрицы как единого массива значений не происходит. Обычно это выгодно, когда матрица формируется из структурированных массивов меньшей размерности; в этом случае можно формировать матрицу «на лету» маленькими блоками. Такой подход обычно требует кратно больше арифметических операций для построения результата умножения матрицы на вектор, но значительно сокращает требования к объему оперативной памяти. Если же удастся использовать векторные инструкции, например, AVX2/AVX512 [10, 11] для процессоров x86 или NEON/SVE [12, 13] для процессоров ARM, то общая скорость операции может значительно вырасти. Так же положительно на скорости вычислений сказывается локализация данных и использование КЭШ памяти процессора.

Рассмотрим матрицу F более подробно. Согласно формуле (2) на расчетной сетке размером $n = n_x \times n_y$ (рис. 2) для построения матрицы F надо вычислить W в $n_x \times n_y$ точках, G – в $n_y \times n_y$ точках, а R – в n_y точках.

Используем i, j, k в качестве индексов строк и столбцов в матрице F . Обозначим i_x, i_y – координаты в расчетной сетке по оси X и Y , соответствующие узлу i . Таким образом $i = i_x + i_y \times n_x$, аналогично для j и k : $j = j_x + j_y \times n_x$, $k = k_x + k_y \times n_x$.

Введем следующие обозначения:

$$\begin{aligned}\Delta_{ij} &= (|i_x - j_x|, |i_y - j_y|) \\ W_{\Delta_{ij}} &= W(|i_x - j_x|, |i_y - j_y|), \\ G_{ij} &= G(i_y, j_y), \\ R_i &= R(i_y).\end{aligned}$$

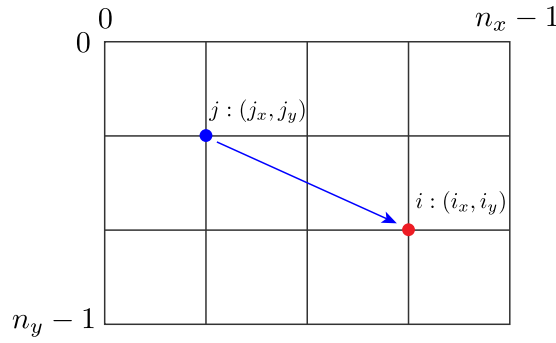


Рис. 2. Расчетная сетка размером n_x узлов вдоль горизонтальной оси X и n_y вдоль вертикальной оси Y . Расстояние между узлами по оси X равно ΔX , а по оси Y – ΔY

Внутренняя структура матрицы F по формуле (2) позволяет записать умножение i -строки матрицы на вектор в виде:

$$u_i = F_i w = \sum_{k=0}^{n-1} f_{ik} w_k = R_i \sum_{k=0}^{n-1} W_{\Delta_{ik}} G_{ik} w_k. \quad (10)$$

Использование формулы (10) позволяет сократить требования к оперативной памяти. Например, для размеров сетки $n_x = 100$, $n_y = 100$ для хранения матрицы F необходимо 800 МБайт оперативной памяти, с другой стороны при использовании формулы (10) для $W_{\Delta_{ij}}$ необходимо 80 КБайт, для G_{ij} – 80 КБайт, а для R_i – 800 Байт, т.е. приблизительно в 5 000 раз меньше. Снижение требований к оперативной памяти позволяет разместить данные в КЭШ памяти процессора, что значительно ускоряет доступ к этим данным. При этом требуемое количество арифметических операций возрастает приблизительно в 2 раза.

Результаты сравнения скорости расчета с использованием обычного алгоритма умножения плотной матрицы на вектор и по уравнению (10) показывают, что наблюдается замедление расчета в более чем 20 раз. Это не может быть связано только с возрастанием количества арифметических операций, формула (10) требует всего в 2 раза больше арифметических операций по сравнению с обычным умножением.

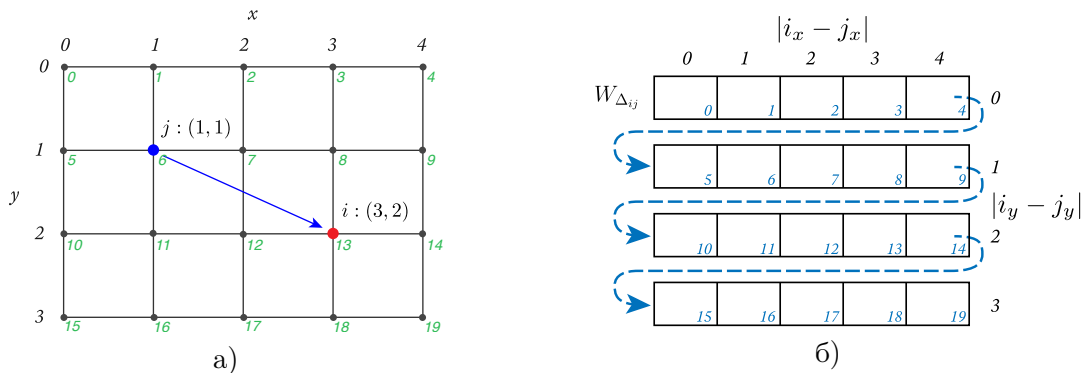


Рис. 3. Пример расчетной сетки ($n_x = 5$, $n_y = 4$). Зеленым выделены индексы узлов расчетной сетки в матрице F (узел $j = 6$, координаты $j_x = 1, j_y = 1$, узел $i = 13$, координаты $i_x = 3, i_y = 2$). На рисунке б представлено расположение элементов $W_{\Delta_{ij}}$ в оперативной памяти, синим цветом выделены индексы $l(i, j) = |i_x - j_x| + n_x \times |i_y - j_y|$ в линейном массиве

Причиной значительного замедления работы является непоследовательный доступ к данным, что приводит к большому количеству «КЭШ промахов» и затрудняет для процессора предсказание ветвлений.

Для примера рассмотрим сетку ($n_x = 5$, $n_y = 4$) (рис. 3а). Размерность матрицы F – $n = n_x \times n_y = 20$. Для хранения $W_{\Delta_{ij}}$ используется одномерный массив длиной $n_x \times n_y$ с

индексированием согласно формуле $l(i, j) = |i_x - j_x| + n_x \times |i_y - j_y|$ (рис. 3б). Рассмотрим умножение строки матрицы, соответствующей элементу сетки $i = 13$ с координатами $i_x = 3$, $i_y = 2$ (красная точка на рис. 3а) на вектор. Согласно формуле (10) последовательность доступа к массиву $W_{\Delta_{ik}}$ представлена в таблице 1.

Таблица 1. Пример изменения Δ_{ik} и линейного индекса $l(i, k)$ массива $W_{\Delta_{ik}}$ при расчете u_{13} согласно формуле (10) $i = 13$ на сетке $n_x = 5$, $n_y = 4$. Отметим, что k пробегает все значения от 0 до $n - 1 = 19$

k	0	1	2	3	4	5	6	7	8	9
(k_x, k_y)	(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(0, 1)	(1, 1)	(2, 1)	(3, 1)	(4, 1)
Δ_{ik}	(3, 2)	(2, 2)	(1, 2)	(0, 2)	(1, 2)	(3, 1)	(2, 1)	(1, 1)	(0, 1)	(1, 1)
$l(i, k)$ в массиве $W_{\Delta_{ik}}$	13	12	11	10	11	8	7	6	5	6
k	10	11	12	13	14	15	16	17	18	19
(k_x, k_y)	(0, 2)	(1, 2)	(2, 2)	(3, 2)	(4, 2)	(0, 3)	(1, 3)	(2, 3)	(3, 3)	(4, 3)
Δ_{ik}	(3, 0)	(2, 0)	(1, 0)	(0, 0)	(1, 0)	(3, 1)	(2, 1)	(1, 1)	(0, 1)	(1, 1)
$l(i, k)$ в массиве $W_{\Delta_{ik}}$	3	2	1	0	1	8	7	6	5	6

Решением проблемы с непоследовательным доступом к памяти является расширение $W_{\Delta_{ij}}$ путем зеркального отображения по оси X и Y (рис. 4).

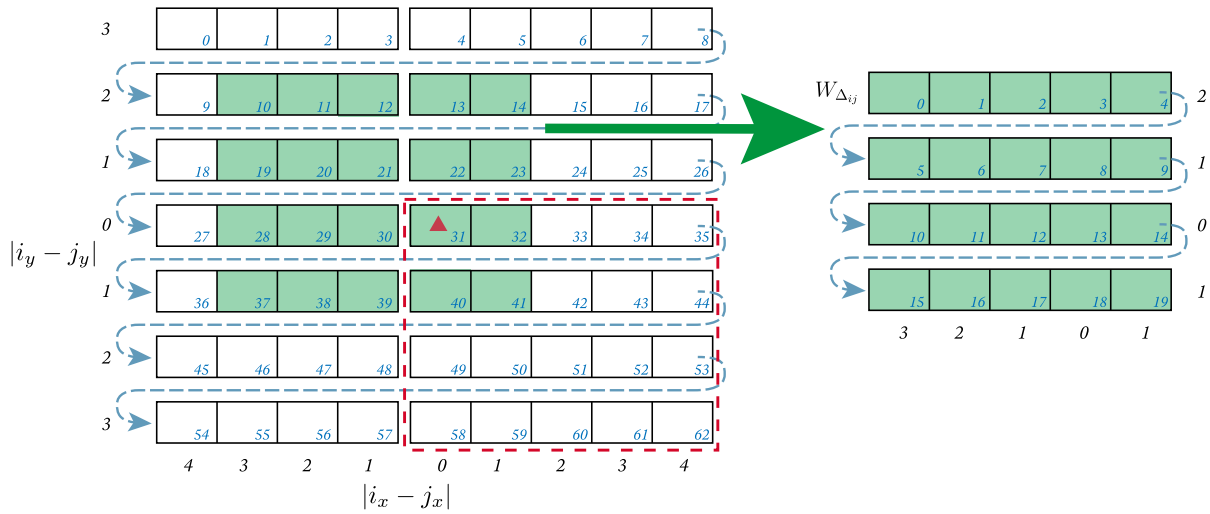


Рис. 4. Пример расширенного массива элементов $W_{\Delta_{ij}}$ в оперативной памяти, синим цветом выделены индексы l в линейном массиве. Зеленым выделена часть, необходимая для умножения строки матрицы, соответствующая узлу $i = 13$ с координатами $i_x = 3$, $i_y = 2$. Красным пунктиром обозначен исходный (до расширения) массив $W_{\Delta_{ij}}$. Элемент массива $W_{\Delta_{ij}}$ с красным треугольником – центр симметрии, т.е. элементы расширенного массива $W_{\Delta_{ij}}$ с индексами 30 и 32 совпадают, аналогично 51 и 11, 49 и 13, и т.д.

Ускорение достигается за счет использования векторных инструкций из расширения AVX2 [11] для архитектуры x86 и NEON [12] для архитектуры ARM. AVX2 позволяет обрабатывать блоки матрицы 8×8 для одинарной точности (FP32), и 4×4 для двойной точности (FP64). Реализация с использованием NEON оперирует с блоками матрицы 4×4 независимо от точности.

Итоговое ускорение относительно классического метода составило от 2 раз на матрицах размерности 500, до 3.5 раз на матрицах размерности 40 000. В дальнейшем во всех численных результатах используется «безматричное» умножение F .

5. Численные результаты

Для тестирования в качестве линейного решателя будем использовать предобусловленный метод BiCGStab [14, 15]. BiCGStab является одним из универсальных и наиболее часто используемым методом для решения разреженных несимметричных матриц.

Тестовый пример: решение задачи фильтрации флюида в деформируемой поровой среде. Всего решаемых матриц 5432, размерность матриц от 53 до 10 567, количество ненулевых элементов в матрице для предобуславливателя $\tilde{A}$ вида (5) максимум 501 011.

В качестве тестового стенда использован ноутбук Apple MacBook Pro с процессором Apple M4 Pro, архитектура ARM, 10 быстрых ядер 4.5 ГГц, пропускная способность оперативной памяти 273 ГБайт/секунду.

Будем рассматривать следующие варианты предобуславливателей:

- OBILU0 – блочный алгоритм неполного LU разложения с перекрытием. Матрица A делится на диагональные блоки по количеству параллельных потоков вычисления, блоки перекрываются (рис. 16). Размер перекрытия выбирается так, чтобы элементы вне перекрытия не влияли на внутреннюю часть блоков;
- OBILUT – блочный алгоритм неполного LU разложения с перекрытиями (рис. 16) и разложением в модификации ILUT [3], величина параметра $\tau = 1e^{-5}$ – при разложении матрица $\tilde{A}$ отбрасываются все элементы $\tilde{A}_{ij}$ такие, что $|\tilde{A}_{ij}| < \tau \|\tilde{A}_i\|_2$, где $\tilde{A}_i$ – i строка матрицы $\tilde{A}$;
- BILU0+AIPS2 – блочный алгоритм неполного LU разложения без перекрытия (рис. 1a). Для улучшения сходимости используем AIPS2 первого порядка $N = 1$ (9).

Результаты тестовых расчетов приведены в таблице 2. Как видно для системы (1) более выгоден подход BILU0+AIPS2, ускорение относительно стандартно используемых методов OBILU0 и OBILUT составляет 2.78 и 3.25 раза соответственно. Вместе с тем для подхода BILU0+AIPS2 ускорение на этапе построения предобуславливателя (Setup) составляет 6.41 на 8 вычислительных потоках, несколько хуже чем 7.65 раза для OBILU0. Данный результат удивителен, так как BILU0+AIPS2 не используется перекрытие блоков и должен ускоряться лучше OBILU0, этот факт требует дополнительных исследований. Видно, что для BILU0+AIPS2 фаза построения предобуславливателя занимает минимальное время, потому что AIPS2 не требует затрат на построение, а BILU0 проще строить, чем OBILU0, так как нет перекрытия блоков.

6. Выводы

Проведено исследование эффективности применения различных типов предобуславливателей для решения систем линейных алгебраических уравнений, возникающих в процессе решения задач фильтрации флюидов в пористой среде с учетом изменения напряженно-деформированного состояния породы. Показано, что совместное использование блочных предобуславливателей типа неполного LU разложения совместно с разложением обратной матрицы в степенной ряд позволяет ускорить поиск решения системы линейных уравнений до 2.8 раз по сравнению со стандартными, универсальными методами предобуславливания.

Дополнительно приведены подходы к использованию и оптимизации «безматричного» умножения плотной матрицы на вектор. Показано, что при наличии дополнительных знаний о построении плотной матрицы возможно значительно сократить до 50 раз требования к объему оперативной памяти, а при использовании векторных инструкций и локализации данных в КЭШ процессора – увеличить скорость операции умножения матрицы на вектор до 3.5 раза.

Таблица 2. Результаты численных расчетов. Всего матриц 5 432, размерность матриц от 53 до 10 567. **Type** – тип предобуславливателя, **NT** – количество вычислительных потоков, **Iters** – общее число итераций линейного решателя BiCGStab на решение всех матриц, **Setup** – время построения предобуславливателя в секундах, **Solve** – время решения систем в секундах, **Total** – общее время в секундах (**Setup** + **Solve**), **Speedup setup** – ускорение построения предобуславливателя, **Speedup solve** – ускорение решения систем

Type	NT	Iters	Setup (c.)	Solve (c.)	Total (c.)	Speedup setup	Speedup solve
OBILU0	1	227435	99.49	238.34	337.83	1.00	1.00
OBILU0	2	249323	49.69	145.58	195.27	2.00	1.64
OBILU0	4	258346	25.27	97.06	122.33	3.94	2.46
OBILU0	8	273876	13.01	75.91	88.92	7.65	3.14
OBILUT	1	93766	248.31	186.93	435.23	1.00	1.00
OBILUT	2	128316	120.62	117.39	238.01	2.06	1.59
OBILUT	4	160392	59.31	81.76	141.07	4.19	2.29
OBILUT	8	168459	39.39	64.42	103.81	6.30	2.90
BILU0+AIPS2	1	43004	52.06	98.77	150.83	1.00	1.00
BILU0+AIPS2	2	44324	26.36	56.18	82.54	1.97	1.76
BILU0+AIPS2	4	44891	13.90	34.58	48.48	3.75	2.86
BILU0+AIPS2	8	45206	8.12	23.82	31.94	6.41	4.15

Литература

1. Saad Y. Iterative Methods for Sparse Linear Systems, Second Edition. SIAM, Philadelphia, USA, 2003. 567 p. DOI: 10.1137/1.9780898718003.
2. Meijerink J.A., van der Vorst H.A. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix // Math. Comp. 1977. Vol. 31. P. 148–162. DOI: 10.2307/2005786.
3. Saad Y. ILUT: A Dual Threshold Incomplete LU Factorization // Numerical Linear Algebra with Applications. 1994. Vol. 1, no. 4. P. 387–402. DOI: 10.1002/nla.1680010405.
4. Smith B., Bjorstad P., Gropp W. Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge University Press, 2004. 237 p.
5. de Sturler E. Incomplete block LU preconditioners on slightly overlapping subdomains for a massively parallel computer // Applied Numerical Mathematics. 1995. Vol. 19, no. 1-2. P. 129–146. DOI: 10.1016/0168-9274(95)00077-8.
6. Chen K. Matrix Preconditioning Techniques and Applications. Cambridge Univ. Press, Cambridge, 2005. 564 p. DOI: 10.1017/CBO9780511543258.
7. Юлдашев А.В., Репин Н.В., Спеле В.В. Параллельный предобуславливатель на основе степенного разложения обратной матрицы для решения разреженных линейных систем на графических процессорах // Вычислительные методы и программирование. 2019. Т. 20, № 4. С. 444–456. DOI: 10.26089/NumMet.v20r439.
8. Kronbichler M., Kormann K. A generic interface for parallel cell-based finite element operator application// Computers Fluids. 2012. Vol. 63. P. 135–147. DOI: 10.1016/j.compfluid.2012.04.012.

9. Arndt D., Bangerth W., Davydov D. et al. The deal.II Library, Version 9.3 // Journal of Numerical Mathematics. 2021. Vol. 29, no. 3. DOI: 10.1515/jnma-2021-0081.
10. Xie B., Zhan J., Liu X. et al. CVR: Efficient Vectorization of SpMV on X86 Processors // Proceedings of 2018 IEEE/ACM International Symposium on Code Generation and Optimization (CGO'18). ACM, New York, NY, USA, 2018. DOI: 10.1145/3168818.
11. Intel Advanced Vector Extensions 10.2, Architecture Specification. 2025.
URL: <https://cdrdv2.intel.com/v1/dl/getContent/828965> (дата обращения: 18.02.2025).
12. NEON Programmer's Guide. Version: 1.0. 2013.
URL: <https://developer.arm.com/documentation/den0018/a/?lang=en> (дата обращения: 23.02.2025).
13. SVE Programming Examples. 2024.
URL: <https://developer.arm.com/documentation/dai0548/latest/> (дата обращения: 23.02.2025).
14. Barrett R., Berry M., Chan T.F. et al. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods // Mathematics of Computation. 1996. Vol. 64, no. 211. DOI: 10.2307/2153507.
15. van der Vorst H.A. Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems // SIAM Journal on Scientific and Statistical Computing. 1992. Vol. 13, no. 2. DOI: 10.1137/0913035.

Метод схем для эффективного выполнения графов задач

П.А. Васёв¹, С.В. Поршневу^{1,2}

¹Институт математики и механики им. Н.Н. Красовского
Уральского отделения Российской академии наук,

²Уральский федеральный университет им. Б.Н. Ельцина

Модель графов задач выглядит привлекательно для автоматизации программирования параллельных программ. Однако оказывается, что при ее прямом практическом применении возникают существенные накладные расходы. В настоящей работе предложено решение, как их избежать. Предлагается «метод схем». Пользователь описывает параллельную программу в виде схемы как набор взаимодействующих модулей. Каждый такой модуль определяет вычислительные процессы, которые запускаются на узлах суперкомпьютера. Взаимодействие модулей определяется связями между их портами ввода-вывода. На основе этих связей формируется следующий уровень связей – между вычислительными процессами. Взаимодействуя и реагируя на сообщения, процессы распределенно формируют и выполняют задачи графа задач. Предполагается, что данный метод будет удобен и эффективен.

Ключевые слова: суперкомпьютерные технологии, высокопроизводительные вычисления, параллельное программирование, граф задач, модели программирования.

1. Введение

Параллельное программирование – сложная задача. Чтобы справляться с ней, разрабатываются модели параллельных вычислений, которые структурируют и упрощают процесс мышления программиста. Например, разработаны модели:

- А-система Котова—Нариньяни [1] и тотальный параллелизм Котова [2];
- схема Карпа—Миллера [3, 4];
- непоследовательные процессы Петри [5]¹ и сети Кана [6];
- смешанные вычисления А.П. Ершова и идеи Л. Ломбарди о пошаговых вычислениях с неполной информацией [7];
- технология фрагментированного программирования [8, 31];
- машины абстрактных состояний Гуревича для параллельных алгоритмов [9];
- рекурсивные конвейеры [10] и модель bulk synchronous parallel [34];
- разнообразные модели Dataflow [4, 6], исчисления процессов (process calculi), например CSP Хоара [4], и другие.

Предназначение моделей вычислений состоит не только в том, чтобы человеку было удобно мыслить, разрабатывая алгоритмы. Но и в том, чтобы выраженные в этих моделях программы было возможно эффективно выполнять на машине. Под эффективностью здесь понимается соотношение затраченных аппаратных ресурсов и астрономического времени выполнения программы. Эффективность обеспечивается снижением накладных расходов, числа излишних операций, и т. п., вплоть до смены аппаратной базы. Если модель вычислений и удобна для человека, и эффективна для выполнения на машине, то такая модель «хорошая».

Меры удобства для человека бывают разные. Существуют чисто теоретические модели, которые хороши для суждений о свойствах алгоритмов, но неудобны в практическом применении (например, модель Тьюринга). Существуют модели и прикладные средства, такие как MPI, которые позволяют создавать универсальные и высокоэффективные приложения, но требуют высокой концентрации от программиста. Существуют и специализированные модели, которые

¹Непоследовательные процессы Петри (Petri non-sequential processes, signal spaces) – модель, которая существенно отличается от широко известной модели сетей Петри.

крайне удобны и вычислительно эффективны, но в то же время не могут описать произвольные вычисления (например, такова модель map reduce).

Меры эффективности также бывают разные, и зависят от машин. Раньше машины использовали для ввода данных перфокарты и ленты, и важным аспектом было последовательное чтение данных, и программы должны были это учитывать. С появлением дисков с произвольным доступом эта потребность отпала. Другой пример: современные электронные процессоры и ускорители, которые диктуют, как следует формировать программы. Но в настоящее время разрабатываются и фотонные процессоры, и в случае их успеха уже эти технологии будут определять, какие операции следует использовать, как их следует группировать и т.п.

Для программирования параллельных программ определенной популярностью пользуется модель графа задач (task graph) [11–14]. В англоязычной литературе выделяют системы, работающие по этой модели, термином asynchronous many-task systems и task-based runtime systems, а модель программирования также называют sequential task flow.

Графы задач обладают рядом преимуществ, как в плане удобства мышления, так и эффективности выполнения, но имеют и существенные недостатки: при реализации «напрямую» они неэффективны на мелкозернистых задачах (т. е. на задачах с малым числом операций).

В настоящей работе предлагается некоторое улучшение для модели графа задач, которое устраняет обозначенный недостаток. Предполагается, что оно удобно для мышления и эффективно для выполнения в настоящее время, на текущем этапе развития. В будущем ситуация поменяется, и для уровней мышления программистов будущего и техники будущего найдутся другие, более подходящие модели и их улучшения.

В основе предлагаемого улучшения лежит переосмысление процесса динамического формирования графа задач, которое далее называется «метод схем»².

2. Модель графа задач

Согласно [13], модель графа задач можно определить следующим образом.

Граф задач – это ориентированный ациклический взвешенный граф $G = (V, E)$, в котором V – множество вершин, представляющих задачи, E – множество дуг, представляющих передачу данных между задачами. Также иногда дополнительно в E включают отношение предшествования между задачами³. **Задача** определяется как список инструкций (операторов), выполняемых последовательно на одном **исполнителе** (процессоре, ускорителе и т.п.).

Задача не может начать выполнение, не получив все входные данные. Следовательно, условием запуска задачи на выполнение является завершение выполнения всех задач-предшественников на графе задач. При выполнении этого условия задача готова к выполнению, которое произойдет далее в некоторый момент времени. Выполнение одних задач может происходить параллельно с другими задачами, и параллельно с передачей данных между задачами.

Понятие графа задач связано с понятием графа алгоритма [11]. Граф задач является образом графа алгоритма, т. е. граф алгоритма можно выражать в том или ином графе задач. Другими словами, граф алгоритма можно «упаковать» в тот или иной граф задач. Смысл такой упаковки – выполнение операций графа алгоритма группами, для снижения накладных расходов.

Назначение, на каком исполнителе в какой момент времени какую задачу следует выполнять – это отдельный процесс планирования (scheduling), поведение которого задается некоторым алгоритмом, который определяется или вычислительной средой, или пользователем.

Если алгоритм планирования определяется средой, то это может подразумевать динамическое планирование, исходя из текущей нагрузки на исполнители и других параметров. Это удобно для пользователя, но влечет накладные расходы. Если алгоритм определяется пользователем, это снимает расходы на динамическое планирование, но требует, чтобы пользователь сам определил, где следует выполнять задачи. Пользователь может определить это заранее,

²В настоящей работе слово «схема» используется в общем смысле, в контексте «схема параллельных вычислительных компонент», и не связано с понятием *схем программ*, введенным основоположником теории программирования А.А. Ляпуновым.

³ На практике иногда применяются и более сложные методы описания взаимных зависимостей задач, например с указанием режима доступа к блокам данных (read-write-readwrite), см. например [18, 19].

статично, или определять динамически во время исполнения. Возможны также различные гибридные подходы, см. [16].

Преимущества модели графа задач:

1. Граф задач это удобная модель мышления для программиста. Параллельную программу в этой модели можно определять последовательным алгоритмом. Последовательные алгоритмы понятны и удобны для человека.
2. Возможность динамического и автоматического (без участия пользователя) балансирования нагрузки на множестве исполнителей.

Главное преимущество модели графа задач заключается в ее удобстве для мышления программиста. Используя метафору графа задач, возможно описание параллельного вычисления, причем формировать это описание возможно последовательным алгоритмом.

Например, такой алгоритм может быть устроен следующим образом (более подробное описание дано в [15, 17]). Пусть есть вычислительная среда, выполняющая задачи в модели графа задач. Пусть среда предоставляет операцию добавления задач. В результате выполнения этой операции немедленно формируются указатели на будущие результаты выполнения этих задач так называемые обещания (promise и future). Эти обещания можно использовать как аргументы при вызове других операций добавления задач. Заметим, что операции добавления задач в среде выглядят как обычные вызовы функций, и результаты тоже выглядят как обычные результаты срабатывания функций. Таким образом, алгоритм формирования графа задач можно записать в последовательной форме, и никаких дополнительных ухищрений не потребуется. Получается, что программист работает с привычной и удобной метафорой последовательного программирования. При этом фактическое выполнение задач может происходить параллельно, исходя из решений, принимаемых в процессе планирования (назначения задач исполнителям).

Модель графов задач настолько удобна для мышления, что она позволяет мыслить о параллельных вычислениях, выраженных и в других моделях. Например, в модели графов задач удобно представлять итеративные программы (то есть программы, проводящие вычисления над каким-то блоком данных в цикле), написанные на MPI. Так, тело внешнего цикла каждого из MPI-процессов в модели графов задач есть запуск списка задач, где каждая итерация представляется отдельной задачей. Блокирующее получение MPI-сообщений в модели графов задач выражается как запуск задач-продолжений (англ. continuations), в качестве аргументов которым указываются обещания результатов других задач, возможно с других исполнителей. Если в MPI при блокирующем получении сообщений с внешними результатами ожидание выражается явно, то в графе задач достаточно предъявить объект обещания этих результатов^{4,5}.

Среди систем на основе модели графа задач: SYCL task graph, CUDA Graphs, OpenTS, Template Task Graphs, PaRSEC, HPX, Charm++, Uintah, Kokkos, Legion, C++ Sender Library, LuNA, Dask, Flyte, TaskTorrent, OpenMP tasks, и другие. Они различаются между собой сферой применения, наличием своего языка программирования, и другими нюансами.

Также отметим систему Parallel Programming Kit разработки одного из авторов настоящей работы, которая описана в [15–17, 33] и опубликована на сайте github.com/pavelvasev/ppk.

3. Проблемы реализации модели графа задач

При реализации модели графов задач могут проявляться (и проявляются) разнообразные проблемы, среди которых в настоящей работе предлагается сфокусироваться на следующих:

⁴Отметим, что чтобы получить обещание результата вычислений с другого исполнителя (например узла суперкомпьютера), нет необходимости общаться с ним. Достаточно договориться о порядке именования такого обещания.

⁵Модель графа задач удобна для интерпретации и обычных ситуаций. Например, иногда программных интерфейсах присутствуют блокирующие вызовы, такие как работа с файлами в языке Си – `open`, `fwrite`, `fread`. При необходимости одновременного выполнения операций в таких интерфейсах пользователи вынуждены создавать нити, применять мьютексы для обмена данными между нитями, и т.п. В модели графов задач происходящее выглядит следующим образом. Блокирующий вызов операции это есть запуск двух задач: собственно операции и алгоритма продолжения. При необходимости одновременного выполнения операций создается столько задач, сколько необходимо. «Нити» при этом не нужны.

П1. Накладные расходы, связанные с передачей задач по сети, в первую очередь задержки по времени.

П2. Необходимость прямого доступа к содержимому блоков данных результатов задач из других задач без копирования памяти.

П3. Необходимость систематизации и минимизации количества аллокаций блоков данных.

П4. Необходимость предварительной инициализации (структур данных, библиотек и т.п.).

Проблема **П1** это накладные расходы, которые появляются в связи с обслуживанием задач. Если задачи формируются в некотором процессе, а выполняются в других процессах, то эту задачу надо добавить в среду, запомнить в некоторой форме, отследить момент готовности этой задачи к исполнению (если в аргументах есть обещания с результатами других задач), провести планирование и выбрать исполнителя, передать по сети процессу исполнителя, и т.п. Все эти моменты, каждый ничтожный сам в себе, в сумме накапливаются. С ростом числа задач они начинают занимать существенную часть как оперативной памяти, так и вносить существенные задержки по времени.

Всякое более менее серьезное вычисление может содержать миллиарды задач и более. Каждая задача из этого миллиарда проходит по разным узлам вычислительной среды, передается по сети и т. д. Если каждая задача содержит в себе относительно небольшое количество операций исполнителя (например 10^6 арифметических действий), а на обслуживание этой задачи уходит сопоставимое количество операций, то использование модели графа задач становится неэффективным и невыгодным.

Проблема **П2** поясняется следующим. Пусть некая задача T произвела блок данных⁶. Модель графов задач подразумевает, что этот блок данных передается на вход некоторым другим задачам. Возникает вопрос, как именно он передается: посредством копирования или без него. Если задача подразумевается к исполнению на другом узле, копирования и передачи данных по сети не избежать. Если задача подразумевается к исполнению на этом же узле, где произведен блок данных, то копирование можно делать, а можно не делать. Важный аспект заключается в том, что с учетом скоростей работы современных процессоров копирование данных в оперативной памяти это очень затратная операция, сопоставимая с самими вычислениями. Если вычисление занимает лишь несколько тактов на элемент блока данных, то получается что копирование блока данных по временным затратам сопоставимо с самими вычислениями. Поэтому существенно и важно, чтобы блоки данных в оперативной памяти при передаче между задачами не копировались.

Проблема **П3** модели графов задач связана с тем, что на практике не получается просто взять и начать решать задачи. Каждой задаче необходима оперативная память для размещения результатов ее работы. Можно получать эту память обычным способом, путем ее запроса (аллокации) у операционной системы (ОС) исполнителя. При этом:

1. Операция аллокации памяти это «дорогая» операция. Она настолько дорогая, что ее чрезмерное применение может в разы снижать производительность вычислений [32].

2. Аллоцированную память необходимо освобождать, как только она перестает быть нужной. Однако не вполне ясно, когда именно можно освобождать ту или иную память в условиях среды, когда результаты задач используются другими задачами, а граф задач строится динамически внешним алгоритмом.

Проблема **П4** подразумевает, что иногда для решения задачи необходимо провести дорогостоящую предварительную подготовку. Например, инициализировать ту или иную библиотеку, или подготовить особые структуры данных в памяти. Иногда эти операции можно было бы не повторять для решения однотипных задач. Но модель графа задач не учитывает этой особенности.

4. Обзор литературы

Принципиально проблемы **П2** и **П3** рассмотрены еще в 60х годах, см. [1, 3], и на современном уровне решение развивается например в [20, 21, 31]. Общая идея заключается в примене-

⁶Под блоком данных понимается блок оперативной памяти, в котором хранятся «тяжелые» данные – вычислительные сетки и т.п.

нии той или иной модели доступа к оперативной памяти. Вместо аллокации памяти, вводится пул буферов. Память не аллоцируется, а используется тот или иной буфер. В один момент времени буфер передается в управление одной задаче, в другой момент времени – другой задаче.

Например, в работе [21] описывается концепция на основе частичного порядка операций. Это определяет, в какой момент времени каким задачам следует выполняться. Это определяет и время, когда та или иная задача может иметь доступ к блокам памяти. Операции подразумеваются над общей памятью. Таким образом разные задачи имеют возможность доступа к одним и тем же блокам памяти, а конфликты устраняются заранее программистом, который формирует частичный порядок тем или иным алгоритмом.

Проблема **П1** решается на современном уровне по-разному. Общая идея заключается в том, чтобы так или иначе оптимизировать хранение и передачу данных о задачах на исполнители, а в идеале – и вовсе отказаться от такой передачи.

В работах [19] и [22] используется понятие *Parametrized Task Graph*, которое заключается в следующем. Граф задач представляется в форме функции, которая по аргументу (параметру) возвращает очередной участок графа. На исполнители передается не сам граф (и таким образом не сами задачи и зависимости между ними), а эта функция. Исполнители вызывают ее по мере продвижения по графу, и получают таким образом информацию о новых задачах, их выходах, и входах. Таким образом, отсутствует необходимость передачи задач по сети, а точнее задачи и граф в целом передаются в более компактной форме. Минус этого подхода – статичность функции порождения графа (т. е. неизменность по ходу вычисления) и высокая сложность ее описания для программиста, что отмечают и авторы.

В работе [18] предлагается другой подход. В нем граф порождается с помощью обычных операций добавления задач в среду, но на алгоритм добавляющий задачи накладывается ограничение: 1) алгоритм порождения графа должен быть описан в форме, которую можно передать по сети, и 2) при добавлении задачи сразу должен указываться исполнитель для нее, т. е. планирование назначения задач исполнителем ограничено моментом постановки задачи. Далее среда – разносит указанный алгоритм по исполнителям, и запускает его на каждом исполнителе. При добавлении очередной задачи исполнитель берет задачу только в случае, если она предназначена для него, а в противном случае игнорирует эту задачу.

Таким образом, как и в подходе [19], вместо графа на исполнители передается функция порождения графа. Отличие заключается в семантике и форме этой функции. В [19] это функция описывает процесс порождения графа задач и она параметризована идентификаторами, соответствующими некоторой нумерации узлов графа. А в подходе [18] эта функция существенно проще – это обычный алгоритм порождения графа посредством вызова операции добавления задач. Некоторый минус этого подхода – происходит дублирование вычисления функции порождения графа на всех исполнителях.

Ранее авторами настоящей работы был предложен метод итераций, описанный в работе [23]. Пользователь формирует подграф графа задач такой, который составляет одну итерацию вычислительного процесса. Затем пользователь имеет возможность выполнить операцию запуска этого подграфа указанное количество раз на указанном множестве исполнителей. При этом среда обеспечивает корректную связь выходных каналов задач подграфа с входными каналами задач подграфа следующей итерации. Более того, среда обеспечивает автоматизацию формирования структуры подграфа задач в оперативной памяти, перехватывая вызовы операции добавления задач в граф задач. Однако эта возможность представляется излишней, и удобнее не обеспечивать такую автоматизацию, а предлагать явные средства по формированию подграфа итерации. Кроме того, этот подход применим только для описания итеративных вычислений.

В технологии C++ Sender Library предлагается строить граф задач с помощью двух стадий: построение шаблона и запуск шаблона. Это позволяет оптимизировать передачу задач, потому что при повторном использовании одного и того же шаблона можно передавать по сети не описание задач, а идентификатор шаблона. Более того, есть операция запуска цикла *repeat_n*, которая означает запуск заданного подграфа несколько раз. Научные вычисления зачастую итеративные, и поэтому операция запуска цикла способна решить проблему накладных расходов на передачу задач и на их запуск. Аналогичный подход используется и в CUDA Graphs.

Проблема **П4** решается в проектах как правило путем введения в вычислительную модель акторов (англ. actor), например см. *chare* в Charm++. Также используется термин компонента (HPX, Dask). Задача в этом подходе решается не вычислительной средой, но конкретным экземпляром актора. Он может провести предварительную инициализацию и использовать ее результаты для решения поступающих экземпляру задач.

Все перечисленные способы в принципе решают обозначенную проблему накладных расходов графов задач. Но вопрос – какой ценой. Одни методы требуют взамен усложнения способа описания графа задач. Другие – проще в работе, но ориентированы исключительно на итеративные расчеты и фиксирует структуру одной итерации. Кроме того, методы не всегда подразумевают взаимодействия задач итераций с внешним миром.

5. Метод схем

Для решения проблем **П1-П4** предлагается следующий метод.

Используем метафору блок-схемы. С помощью записи, похожей на блок-схему, будем определять параллельный вычислительный процесс. А именно, схема – это структура, которая состоит из модулей, их портов, и связей между портами. Модули определяют вычислительные процессы, которые выполняются на исполнителях и взаимодействуют посредством связей выходных и входных каналов. А эти вычислительные процессы, в свою очередь, будут порождать и выполнять задачи из модели графа задач. Таким образом, схема определяет распределенный алгоритм динамического построения графа задач и алгоритм планирования его выполнения. Обратимся к деталям.

Исполнитель – однопоточный процесс уровня ОС. Исполнители запускаются на узлах суперкомпьютера. Исполнители можно использовать разные, исходя из оборудования и языковой среды. Например если на узле 64 ядра и 2 GPU, то можно запустить 32 исполнителя для двоичных кодов, 32 для языка Python, и 2 для GPU. Задача исполнителя – быть местом выполнения вычислительных процессов. Один исполнитель может выполнять сразу множество вычислительных процессов, благодаря их структуре, описанной далее.

Вычислительный процесс – это логический процесс, задача которого проводить конкретные вычислительные операции над входными данными и выдавать результаты.

Введем структуру вычислительного процесса. Процесс ожидает данных в форме сообщений из входных каналов (см. далее). По получению данных процесс проводит *вычисление*, и записывает результаты в выходные каналы, и возвращается в ожидание сообщений из входных каналов⁷. Вычисление непрерываемое, неблокирующееся и не длительное (подходящее значение длительности определяется экспериментально).

Благодаря такой структуре становится возможным размещение не одного, а множества вычислительных процессов на одном исполнителе. Логически, эти процессы выполняются все одновременно. Технически же они выполняются поочередно, выполняют свои вычисления по мере поступления сообщений.

Канал – это коммуникационный примитив. Поддерживаются операции: записать в канал, прочитать из канала. В канал можно записывать значения (т. е. сообщения) произвольной структуры, чтение и запись атомарны. Каналы бывают двух типов: 1) локальные по отношению к процессам, 2) глобальные, с общим на всю схему пространством имен каналов. Используя эти имена, любой процесс может взаимодействовать с любым каналом⁸.

Связь между каналами – это коммуникационный процесс, который обеспечивает передачу сообщений между двумя каналами, а именно от канала-источника к каналу-приемнику. Один

⁷ Другими словами, процесс ожидает сообщения, и реагирует на их поступление, выполняя алгоритм *реакции* (в смысле *reaction* [24]). Этот алгоритм и реализует «вычисление». Вычислительные процессы в настоящей работе похожи на процессы Хоара.

⁸ Также прорабатывается вариант, в котором вводится два типа каналов – локальные по отношению к процессам и глобальные в общем пространстве имен. В этом варианте процессы взаимодействуют только со своими локальными каналами, а общее взаимодействие определяется связями этих локальных каналов с глобальными каналами.

вычислительный процесс может записывать сообщения в некоторый канал, а другой процесс читать сообщения из некоторого другого канала. Если эти каналы соединить связью, будет происходить передача сообщений от первого канала ко второму и таким образом – от первого процесса ко второму. Ограничений на количество связей, с которыми взаимодействует канал, не вводится.

Перечисленные выше сущности принадлежат «низкому» уровню, и не присутствуют в схемах напрямую. Схемы предлагается составлять из следующих сущностей:

Модуль – это виртуальный логический процесс преобразования данных. Задача модуля – определить, на каких исполнителях какие вычислительные процессы следует запустить. Список доступных модулю исполнителей сообщается модулю внешним образом, например в момент добавления модуля в схему. Кроме того, каждому модулю соответствует набор портов ввода и набор портов вывода.

Порт – это виртуальный, логический коммуникационный примитив, предназначенный для взаимодействия модулей. Порты разделяются на входные и выходные, и «прикрепляются» к модулям. Взаимодействие модулей осуществляется посредством связей между их портами. Технически порт – это множество каналов («пучок каналов» – выражение М.О. Бахтерева). Более того, это множество определенной структуры. Эта структура может быть списком, двумерным или многомерным массивом, и т. п. Структура соответствует необходимому распределению данных в оперативной памяти узлов суперкомпьютера. При создании того или иного порта модуль определяет идентификаторы (имена) для всех входящих в него каналов, явно или неявно.

Связь между портами – это коммуникационный процесс, который обеспечивает передачу информации между двумя портами. Связываемые порты должны обладать одинаковой структурой и количеством каналов. Связь между портами реализуется как набор связей между соответствующими каналами этих портов, один к одному. Таким образом, когда пользователь устанавливает связь между портами, то на самом деле связываются каналы этих портов. Например, если порт X состоит из каналов $(x_1, x_2, \dots, x_N)$, а порт Y из каналов $(y_1, y_2, \dots, y_N)$, то связь между этими портами будет означать следующие связи каналов: $x_1 \rightarrow y_1, x_2 \rightarrow y_2, \dots, x_N \rightarrow y_N$.

Работа модулей схемы подразумевается следующая. Модуль при запуске получает разные стартовые параметры. Среди параметров – список исполнителей. Модуль запускает свои вычислительные процессы на этих исполнителях. В контексте модуля определяется:

1. Какие и сколько вычислительных процессов следует запустить.
2. Какие параметры у этих процессов.
3. Глобальные идентификаторы для входных и выходных каналов процессов.

После запуска каждый вычислительный процесс организует чтение и запись своих каналов. Посредством связей каналы процесса передают данные другим процессам, и принимают от них данные. Отметим, что процесс может считывать сразу несколько каналов, и более того, выполнять реагирующие действие только когда во всех этих каналах подано очередное значение.

Алгоритм операции записи сообщения в канал.

1. При записи сообщения в канал пишущий процесс блокируется.
2. Проводится обход всех связей, в которых этот канал является источником. Для каждой связи вызывается операция записи сообщения в канал-приемник.
3. Проводится обход всех заявок от процессов на чтение этого канала.
4. Если читающий процесс работает на этом же исполнителе, то ему передается управление. Он выполнит свои реакции, и вернет управление, согласно структуре вычислительных процессов введенной ранее.
5. Если читающий процесс работает не на этом исполнителе, то сообщение пересылается ему методами межпроцессного или сетевого взаимодействия.
6. После обработки всех связей и заявок на чтение, управление возвращается процессу, который записал сообщение в канал.

Благодаря такому алгоритму, вычислительные процессы, находящиеся на одном исполнителе, имеют прямой и бесконфликтный доступ к памяти друг друга. Сообщение может содер-

жать указатели на блоки данных процесса. Поскольку пишущий процесс блокируется, то читающий процесс имеет полный доступ к его памяти, в том числе и на запись.

Шаг 4 алгоритма можно реализовать эффективно, сравнимо по затратам с вызовом функции. Таким образом передача сообщений между процессами на одном исполнителе, включая передачу больших блоков данных, будет производиться быстро.

Описание метода схем завершено. Проиллюстрируем его применение на рис. 1–3.

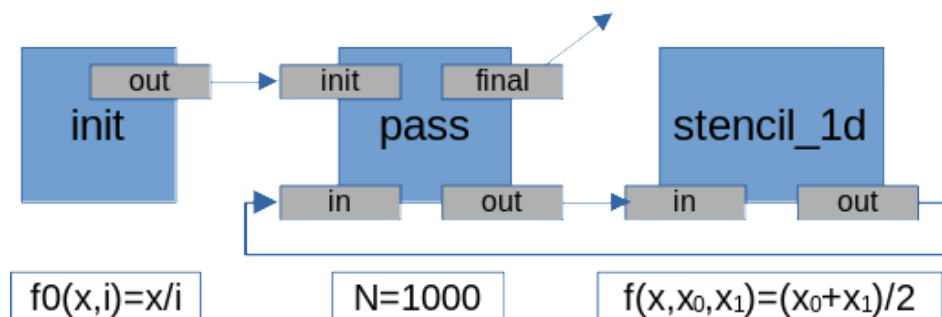


Рис. 1. Схема параллельной программы по вычислению функции одного переменного на регулярной сетке по явной схеме (см. раздел Испытание метода). Синие прямоугольники – модули, белые – параметры модулей, серые – порты, стрелки – связи портов. Модуль `init` генерирует начальные данные и записывает их в свой порт `out`. Модуль `pass` разово передает данные из порта `init`, а затем из порта `in` N раз – в порт `out`. Модуль `stencil_1d` реализует явную схему, вычисляя функцию f , заданную в параметре. Результат `stencil_1d` замкнут на модуль `pass`. В итоге работы схемы на порту `final` появляется результат вычисления N итераций

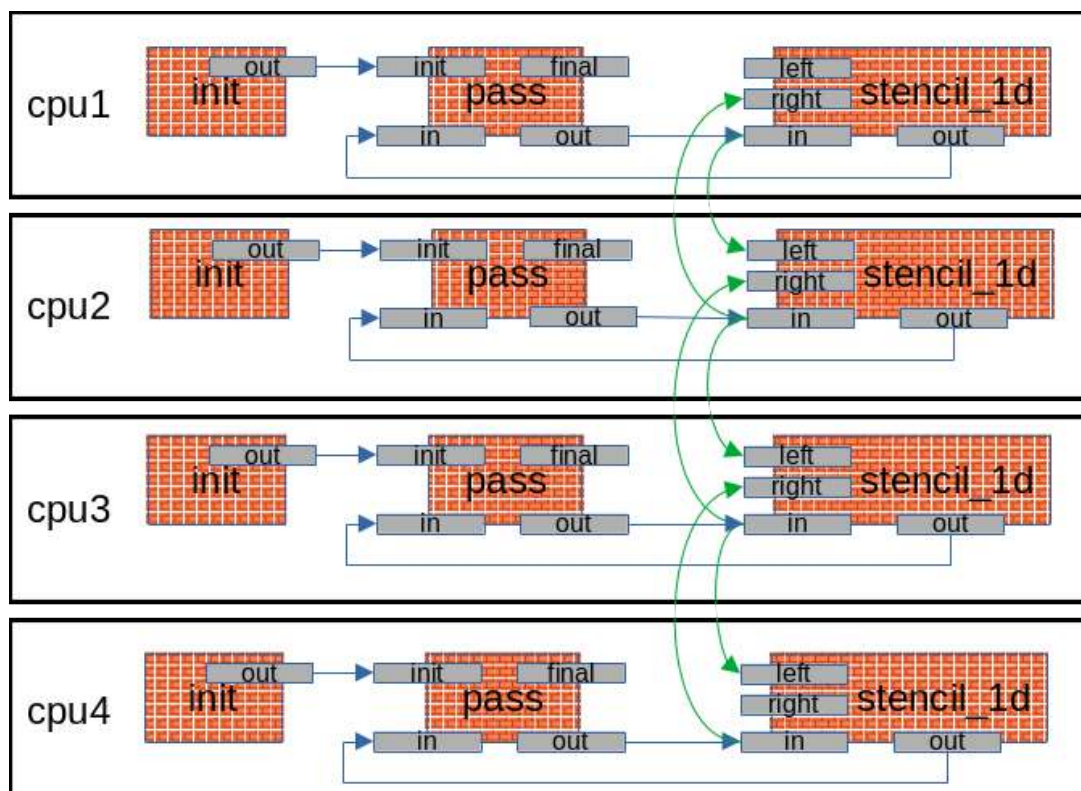


Рис. 2. Набор вычислительных процессов, порожденных схемой из рис. 1. Для иллюстрации показаны процессы для 4х исполнителей (cpu1..4). Кирпичные прямоугольники – вычислительные процессы, серые – каналы, стрелки – связи между каналами.

Каждый исполнитель получил по процессу от каждого модуля. Исходя из связей портов образованы связи между каналами процессов. Также модуль `stencil_1d` наладил внутренние связи своих процессов для передачи граничных значений (зеленые стрелки)

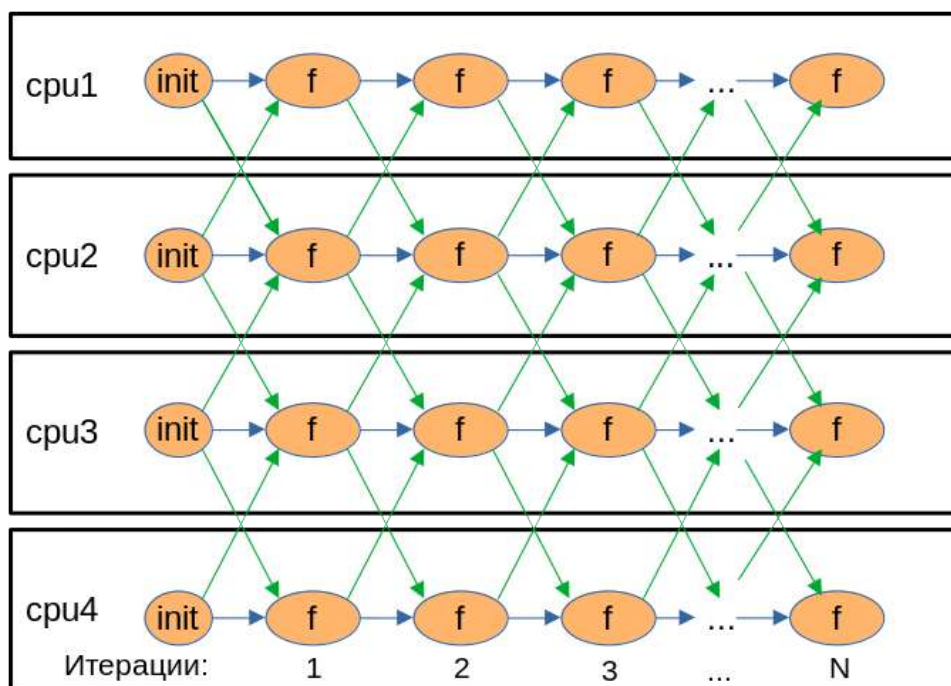


Рис. 3. Граф задач, порождаемый вычислительными процессами из рис. 2. Показано два типа задач: 1) *init* – задача формирует начальные данные, и 2) *f* – задача рассчитывает очередную итерацию на блоке данных. Задачи типа *f* «порождаются» и выполняются процессами *stencil_1d*, причем локально, на своих исполнителях

5.1 Связь схем и графов задач. Решение проблемы накладных расходов

Вычислительные процессы реагируют на поступление сообщений в каналах, выполняя те или иные алгоритмы. Эти алгоритмы (алгоритмы реакций на сообщения) по сути являются алгоритмами задач из модели графов задач. А процессы по сути являются генераторами задач.

Запуск процессов на исполнителях и связь их каналов определяет **распределенный** (в смысле без координирующего центра) процесс формирования графа задач. Сигналом к «добавлению» очередной задачи в граф служит сообщение в канале вычислительного процесса.

Добавленная таким образом задача тут же и «решается» – в алгоритме реакции на сообщение. Результаты ее работы записываются в исходящий канал, что приводит к срабатыванию реакций других вычислительных процессов – граф задач динамически развивается.

Таким образом, решается проблема накладных расходов **П1** на передачу задач. «Добавление» задач в граф задач в методе схем происходит неявно и распределенно, локально по отношению к исполнителям этих задач.

5.2 Прямой доступ к оперативной памяти процессов

Проблема **П2** – возможность прямого доступа к оперативной памяти с результатами выполнения задач – решена в представленном методе. Если два вычислительных процесса назначены на один исполнитель, то при передаче сообщения от первого процесса ко второму данные передаются по указателю, и могут быть свободно прочитаны или изменены. Благодаря зафиксированной структуре вычислительных процессов (см. выше) проблема одновременного доступа к памяти исключается.

5.3 Работа с блоками данных

В методе схем имеются следующие удобные условия:

1. Все блоки данных всех вычислительных процессов, размещенных в некотором исполнителе, находятся в оперативной памяти процесса ОС этого исполнителя.

2. Процессы в рамках одного исполнителя выполняются последовательно, обмениваясь данными через ссылки.

Исходя из этого, на текущий момент среди возможных вариантов просматривается следующее решение проблемы **ПЗ**.

1. Вычислительный процесс может аллоцировать память у ОС для блока данных, который затем может отправить в исходящем сообщении.
2. После этой отправки начинается циркуляция блока данных между реакциями процессов текущего исполнителя. Она может быть продолжительной, например блок может передаваться в цикле между процессами (как в программе раздела 6). В конце концов такая циркуляция заканчивается, и блок становится «свободным». Процесс-аллокатор может использовать его повторно или вернуть ОС.
3. Допускается использовать входящий блок данных для записи результата вычисления реакции процесса. Это позволяет сократить объем используемой памяти. Для обеспечения этого требуются специальные меры, с приоритетом доступа для читающих процессов и одним пишущим процессом в конце.

5.4 Предварительная подготовка

Проблема **П4** по предварительной подготовке решается в представленном методе схем естественным образом. Вычислительный процесс может провести всю необходимую предварительную подготовку по загрузке библиотек, подготовке структур данных и т.п. И затем проводить неограниченное число вычислений на базе этих подготовленных сущностей. В этом смысле вычислительный процесс ведет себя также, как акторы из обзора литературы (см. выше).

6. Испытание метода

Для испытания метода проведен вычислительный эксперимент. Его материалы и коды программ опубликованы по адресу github.com/pavelvasev/ppk/tree/scdays2024/experiments/trial-v2. Впервые этот эксперимент был представлен в работе [23], в настоящей работе он дополнен.

Проводится итеративное вычисление:

- Вычисляется функция одного переменного на регулярной сетке по явной схеме.
- Используется формула: $p_{next} = (p_{left} + p_{right})/2 + \text{Math.random}(1)$.
- Это вычисление реализует упрощенную версию параллельного вычисления аппроксимации динамики нелокального уравнения неразрывности нелинейной марковской цепью [30]. Полная версия, выполненная на модели графов задач, описана в [15].

Сетка одномерная. Элементы сетки – числа с плавающей точкой типа float32.

Вычисление реализовано в последовательном варианте и в нескольких параллельных вариантах для разных моделей программирования:

- Последовательное вычисление (**seq**). Обычная последовательная программа.
- Граф задач (**task-graph**). Используется прямая реализация графа задач со статическим назначением задач на исполнители. Прямая реализация подразумевает, что пользователь запускает последовательную программу, которая связывается со средой вычисления и ставит задачи для графа задач посредством операций этой среды вычисления.
- Ручное распараллеливание (**manual**). Исполнители обмениваются данными посредством обмена сообщениями.
- Метод итераций для графа задач [23] (**iter-graph**). Используется та же реализация графа задач, что и выше в методе графа задач (см. task-graph), для которой добавлены средства формирования подграфа итераций.
- Метод схем, представленный в настоящей работе (**schema**). Используется та же реализация графа задач, для которой добавлены средства формирования схем и их распространения на исполнители.

Параллельные программы используют один и тот же метод распараллеливания: сетка, как одномерный массив, разбивается на P одинаковых частей; на каждой итерации вычисление над

частями проводится параллельно; перед началом следующей итерации происходит обмен данными граничных значений этих частей.

Запуски всех программ выполнены как полная комбинация следующих параметров:

- Количество исполнителей – 2, 4, 8, 16. Исполнители работали на одной машине, каждому было выделено по одному ядру. Процессор Ryzen 1700x, 16 ядер.
- Размер вычислительной сетки: 100 тысяч, 1 миллион, и 10 миллионов. Применение небольшой сетки порождает «мелкозернистые» задачи, и зерна тем меньше, чем больше количество исполнителей.
- Количество итераций: 1 тысяча, 10 тысяч.

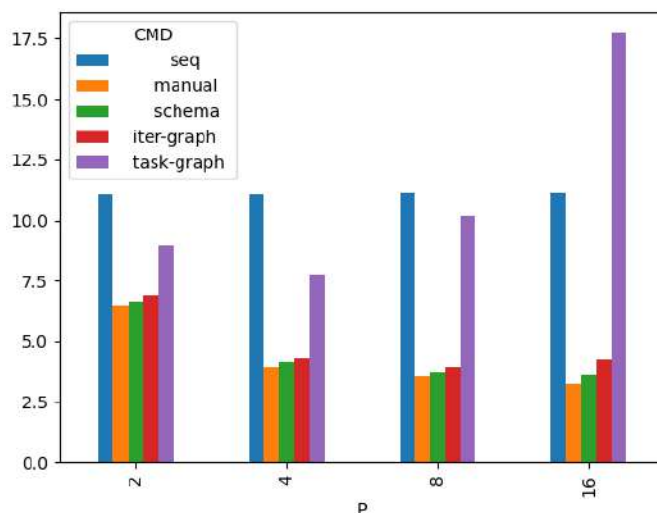


Рис. 4. Время работы программ на сетке 100 тысяч ячеек. По оси X – количество задействованных исполнителей, по оси Y – время в секундах. Цвет – модель программирования.

Сетка 100 тысяч ячеек дает наиболее мелкозернистые задачи, особенно с увеличением количества исполнителей. При 16 исполнителях зерно наиболее мелкое, и визуально заметна чувствительность моделей (а точнее их реализаций) к размеру зерна. Например, реализация графа задач показала производительность даже худшую, чем последовательное вычисление

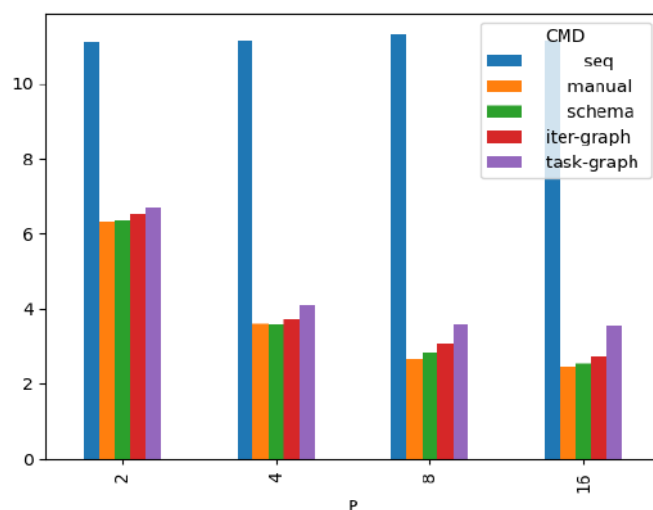


Рис. 5. Время работы программ на сетке 1 млн ячеек. По оси X – количество задействованных исполнителей, по Y – время в секундах. Цвет – модель программирования. Зерна задач крупнее, и время работы разных моделей вычислений стабилизировалось, по сравнению с зернами сетки 100 тысяч ячеек

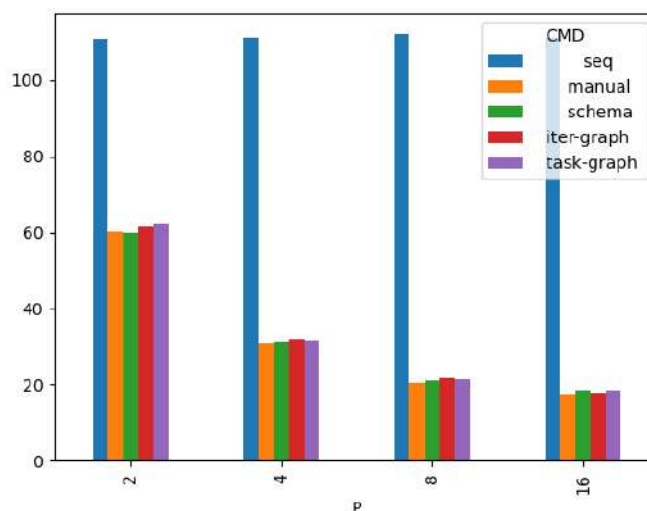


Рис. 6. Время работы программ на сетке 10 млн ячеек. По оси X – количество задействованных исполнителей, по Y – время в секундах. Цвет – модель программирования. Наблюдается схожесть работы методов во времени, зерна наиболее крупные и поэтому накладные расходы минимальны

Результаты испытания представлены на рис. 4–6. Выводы следующие:

1. Ручное распараллеливание показало себя наиболее эффективно. Это ожидаемо: программа была специально написана так, чтобы уменьшить накладные расходы.
2. Метод схем в среднем дал наиболее эффективное вычисление среди других испытанных методов автоматизации параллельного программирования, лучше остальных приближаясь по эффективности вычислений к ручному распараллеливанию.
3. Хорошо проявились накладные расходы при работе графа задач, что видно на рис. 4.
4. По видимому, есть диапазоны «зернистости» задач, где разные методы хороши или плохи. Другими словами, эффективность программ, сгенерированных методом, есть функция от размера (зерна) задачи.

В качестве перспектив дальнейших испытаний метода рассматриваются:

1. Более сложные в плане коммуникации задачи. Текущий тест относительно прост в плане коммуникации. Необходимы тесты на двумерных, трехмерных регулярных сетках; неструктурированных сетках; и на графах.
2. Существенно большее количество исполнителей, в том числе работающих одновременно на разных узлах суперкомпьютера.

7. Анализ метода и перспективы его развития

Принципиальные недостатки метода схем:

- **Статическое назначение на исполнители.** В методе подразумевается статическая балансировка нагрузки. Вместе с тем модель графов задач имеет возможность динамической балансировки, которая приносит преимущества, такие как распределение нагрузки исходя из текущей загруженности исполнителей, возможность гибко разделять суперкомпьютер между пользователями, и другие. Однако назначение задач, передача решений об этих назначениях по сети на исполнители, излишняя передача данных между задачами по сети вносят существенные задержки, которые и призван ликвидировать метод схем.
- **Неустойчивость к отказам узлов.** Для современных вычислений, а особенно это применимо к вычислениям экза-масштаба, уже получены оценки того, что вычислительные узлы будут выходить из строя часто, в том числе во время вычислений [25]. Метод схем в своей постановке не учитывает этой проблемы. И пока нет понимания, возможна ли

разработка эффективного дополнения, которое бы обеспечило отказоустойчивость в этой ситуации.

Устранимые недостатки метода схем:

- **Сложность асинхронного взаимодействия схем.** Метод схем удобен для вычислений, когда данные между процессами передаются на регулярной основе. Однако существуют ситуации, когда передача данных требуется лишь иногда. Например, это обычная ситуация в сценарии онлайн-визуализации [26]. При такой визуализации пользователь может запросить данные в произвольный момент времени. Технически это означает, что данные из некоторых модулей основной вычислительной схемы по внешней команде необходимо передать на входные порты некоторой схемы визуализации. При этом необходимо обеспечить, чтобы данные были переданы согласованно, из соответствующих вычислительных итераций. Это удастся реализовать, но требует специальных мер, основанных на подсчете сообщений в заданных каналах, т. е. номеров итераций.

Среди преимуществ метода схем выделяются следующие:

- **Высокая производительность вычислений.** По сути, метод обеспечивает наполнение вычислительных процессов уровня ОС (процессы исполнителей) путем стыковки алгоритмических слоев, определяемых вычислительными процессами модулей. Эта стыковка обеспечивает передачу данных между этими слоями путем передачи управления в рамках одного процесса ОС, что эквивалентно вызову функций с передачей данных по указателям. Что по эффективности эквивалентно тому, как если бы эти процессы программировались вручную.
- **Динамическое управление содержанием вычисления.** Схему можно менять во время выполнения, и это влечет изменение состава выполняющихся вычислительных процессов в процессах ОС исполнителей, и меняет их взаимодействие между собой.
- **Возможность абстрагирования схем.** Написанную схему можно зафиксировать в качестве нового модуля. Таким образом, метод схем отвечает всем критериям выразительности языков программирования [27]. В нем есть базовые элементы (модули), средства их комбинирования (связи между портами модулей), и операция абстрагирования.
- **Удобное мета-программирование.** Программа, формирующая схему, может быть последовательной, что удобно для мышления. Она может быть написана на удобном для работы языке, например на Python или даже в форме разметки типа YAML. При этом вычислительные процессы, которыми схема управляет, могут быть оптимизированные, например написанные на Си или для GPU.

Среди перспектив развития метода, помимо устранения упомянутых выше недостатков, представляются полезными следующие:

1. Более строгая модель работы с памятью блоков данных (раздел 5.3).
2. Ввод модели сериализации объектов. Если объект передается вычислительному процессу в том же процессе ОС, то его можно передать по указателю. Если же объект передается по сети, то его необходимо сериализовать (в исходящий поток согласно используемой технологии передачи данных) и затем десериализовать при получении. Эти аспекты должны быть проработаны. Сейчас это решается таким образом, что передаются объекты, заранее согласованные со средой выполнения, такие как массивы чисел.
3. Автоматизация назначения процессов модулей на исполнители: сейчас это делается «вручную», путем указания исполнителей модулям. Здесь видится большая перспектива, в том числе на основе сбора статистики, искусственного интеллекта и т.д.
4. Возможность подключения программ, заданных методом схем, к параллельным программам, написанным в других моделях вычислений. Для этого достаточно вынести абстракцию «порта» на внешний уровень, как это сделано в ADIOS2 [29].
5. Повышение эффективности реализации, в частности использование суперкомпьютерных технологий передачи данных (InfiniBand, разные RDMA), например посредством библиотеки UCX.

Вероятно не все эти возможности стоит встраивать в сам метод, но стоит предложить такой программный интерфейс, который позволял бы добавлять их внешним образом.

8. Заключение

Модель графов задач автоматизирует программирование процессов, составляющих параллельные программы. Например, в технологии MPI эти процессы программируются вручную, и программист для каждого процесса описывает, что и когда он делает (когда считает, когда посылает сообщения, что делает при получении сообщений и т. п.). В модели графов задач эти же самые процессы формируются автоматически, исходя из тех задач что ставятся управляющим алгоритмом. При этом управляющий алгоритм может быть последовательным, что существенно упрощает программирование параллельной программы.

Но оказывается, что при подобной автоматизации возникают существенные накладные расходы. Эти расходы при определенных ситуациях оказываются таковы, что параллельная программа может работать даже медленнее, чем последовательная (см. пример на рис. 4).

В настоящей работе предложено возможное решение, как избежать этих накладных расходов. Предлагается использовать «метод схем». Его суть заключается в том, что пользователь описывает параллельную программу как набор взаимодействующих модулей. Каждый такой модуль определяет вычислительные процессы, которые запускаются в процессах уровня ОС на узлах суперкомпьютера. Взаимодействие модулей определяется связями между их портами. Эти связи в свою очередь формируют связи между вычислительными процессами, а точнее между их каналами коммуникации. Таким образом описание схемы, составленное из модулей и связей между ними, определяет поведение комплекса процессов ОС на суперкомпьютере. При этом внутри этих процессов, на теоретическом уровне, работает модель графов задач.

Метод схем является абстракцией более высокого уровня, чем графы задач. По сути он является другой моделью вычислений, со своими сущностями и правилами их взаимодействия. Полагается, что эта модель сможет эффективно порождать динамический граф задач, избегая при этом обременительных накладных расходов.

Отметим связь метода схем с моделью акторов, применяемую например в системе HPX, Charm++ и других. Актор соответствует вычислительному процессу. Модуль схемы определяет свои вычислительные процессы, то есть как бы задает целую группу акторов. Эта группа взаимодействует с другими группами акторов, заданных другими модулями схемы. Взаимодействие определяется связями портов, и как следствие связями каналов «акторов»⁹.

Получается, что метод схем есть способ управления «ансамблем из групп акторов».

Отметим идейную связь предложенного метода с промышленными системами. В проекте Логос-Платформа [28] предлагается модель организации междисциплинарных вычислений на основе связи вычислительных программ посредством внедрения адаптеров (пользовательских функций) программ друг в друга. При этом программы могут быть написаны в произвольных моделях программирования. Метод схем организует подобную связь на уровне модулей. Однако у модулей в настоящей работе нет возможности реагировать на подключение их портов к другим модулям, что необходимо для рассылки адаптеров, и вероятно эту возможность следует предусмотреть.

В проекте ADIOS2 [29] моделей вычислений не предлагается, но предлагается способ связи параллельных программ. Этот способ соответствует предложенному понятию связи между портами модулей. В ADIOS2 к связям добавлена модель времени для итеративных вычислений (beginStep/endStep), в предложенном методе этого пока нет. Связи программ в ADIOS2 многофункциональны, например могут поддерживать перераспределение данных (MxN data transfer). В методе схем это достигается явно, посредством ввода перераспределяющего модуля.

Авторы благодарят коллег, в общении с которыми и сформировалась настоящая работа. Особую благодарность авторы выражают за постановку задачи [30], которая сыграла роль мотива представленного исследования.

⁹Взаимодействие посредством связей каналов обеспечивает лучшую гибкость, то есть настраиваемость системы, чем адресная посылка сообщений и удаленный вызов процедур, которые часто применяются в программировании.

Литература

1. Котов В.Е., Нариньяни А.С. Асинхронные вычислительные процессы над памятью // Кибернетика. 1966. № 3. С. 64–71.
2. Котов В.Е. Проблемы развития параллельного программирования // Труды Всесоюзного симпозиума «Перспективы системного и теоретического программирования». Новосибирск, 1979. С. 58–72.
3. Karp R.M., Miller R.E. Parallel program schemata // J. Comp. Syst. Sci. 1969. Vol. 3, no. 2. P. 147–195. DOI: 10.1016/S0022-0000(69)80011-5.
4. Mikloško J., Kotov V.E. Algorithms, Software and Hardware of Parallel Computers. Springer, Berlin, Heidelberg, 1984. 395 p. DOI: 10.1007/978-3-662-11106-2.
5. Einar S. Non-Sequential Processes and Concurrency Theory // Carl Adam Petri: Life and Science. 2015. P. 77–87. DOI: 10.1007/978-3-662-48093-9.
6. Kahn G. The semantics of a simple language for parallel programming // Information Processing, Proceedings of the 6th IFIP Congress 1974, Stockholm, Sweden, August 5-10, 1974. P. 471–475.
7. Ершов А.П. Смешанные вычисления // В мире науки. 1984. № 6.9.
8. Ахмед-Заки Д.Ж., Лебедев Д.В., Малышкин В.Э., Перепёлкин В.А. Автоматизация конструирования распределенных программ численного моделирования в системе LuNA на примере модельной задачи // Проблемы информатики. 2019. № 4 (45). С. 53–64. DOI: 10.24411/2073-0667-2019-00017.
9. Blass A., Gurevich Y. Abstract state machines capture parallel algorithms: Correction and extension // ACM Trans. Comput. Log. 2008. Vol. 4, no. 4. P. 578–651. DOI: 10.1145/937555.937561.
10. Куприянов Б.В. Оценка и оптимизация производительности рекурсивного конвейера // Автоматика и телемеханика. 2020. Т. 5. С. 6–25.
11. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
12. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем: Учебное пособие. Нижегородский гос. ун-т им. Н. И. Лобачевского. Нижний Новгород: Издательство Нижегородского государственного университета, 2001. 122 с.
13. Прихожий А.А. Распределенная и параллельная обработка данных // Белорусский национальный технический университет, кафедра «Программное обеспечение вычислительной техники и автоматизированных систем». Минск: БНТУ, 2016. 91 с.
14. Stevenson S., Jones S., Oh F. Enabling Dynamic Control Flow in CUDA Graphs with Device Graph Launch. 2022. Nvidia developer blog. URL: <https://developer.nvidia.com/blog/enabling-dynamic-control-flow-in-cuda-graphs-with-device-graph-launch/>.
15. Васёв П.А. Визуализация работы алгоритма планирования параллельных задач // Труды 33-ей Международной конференции по компьютерной графике и машинному зрению ГрафиКон 2023, 19-21 сентября 2023 г., Москва. С. 341–353. DOI: 10.20948/graphicon-2023-341-353.
16. Васёв П.А. Гибридное планирование графов задач // Сборник тезисов докладов 12-го Национального суперкомпьютерного форума НСКФ-2023, Россия, Переславль-Залесский, ИПС имени А.К. Айламазяна РАН, 28 ноября – 1 декабря 2023 года.
17. Vasev P. A Computational Model for Interactive Visualization of High-Performance Computations // Supercomputing. RuSCDays 2023. Vol. 14389 / eds. by V. Voevodin, S. Sobolev, M. Yakobovskiy, R. Shagaliev. Springer, Cham, 2023. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-49435-2_9.
18. Castes C., Agullo E., Aumage O., Saillard E. Decentralized in-order execution of a sequential task-based code for shared-memory architectures. Research Report RR-9450, Inria Bordeaux - Sud Ouest. 2022. 30 p.

19. Hoque R., Herault T., Bosilca G., Dongarra J. Dynamic task discovery in PaRSEC: a data-flow task-based runtime // *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA '17)*. Association for Computing Machinery, New York, NY, USA, 2017. Article 6. P. 1–8. DOI: 10.1145/3148226.3148233.
20. Арыков С.Б., Малышкин В.Э. Алгоритмы конструирования асинхронных программ заданной степени непроцедурности методом группировки // *Вестник Новосибирского государственного университета. Серия: Информационные технологии*. 2009. Т. 7, № 1. С. 3–15.
21. Арыков С.Б. Асинхронная модель вычислений над общей памятью // *Вестник Уфимского государственного авиационного технического университета*. 2016. Т. 20, № 3 (73). С. 114–121.
22. Cambier L., Qian Y., Darve E. TaskTorrent: a Lightweight Distributed Task-Based Runtime System in C++ // *2020 IEEE/ACM 3rd Annual Parallel Applications Workshop: Alternatives To MPI+X (PAW-ATM)*, Atlanta, GA, USA, 2020. P. 16–26. DOI: 10.1109/PAWATM51920.2020.00007.
23. Васёв П.А. Метод итераций для графов задач // *Параллельные вычислительные технологии – XVIII всероссийская конференция с международным участием, ПаВТ'2024, г. Челябинск, 2-4 апреля 2024 г. Короткие статьи и описания плакатов*. Челябинск: Издательский центр ЮУрГУ, 2024. С. 182. DOI: 10.14529/pct2024.
24. Lohstroh M., Menard C., Bateni S., Lee E.A. Toward a Lingua Franca for Deterministic Concurrent Systems // *ACM Trans. Embed. Comput. Syst.* 2021. Vol. 20, no. 4. Article 36. DOI: 10.1145/3448128.
25. Четверушкин Б.Н., Якобовский М.В. Вычислительные алгоритмы и архитектура систем высокой производительности // *Препринты ИПМ им. М.В.Келдыша*. 2018. № 52. 12 с. DOI: 10.20948/prepr-2018-52.
26. Vasev P. Analyzing an Ideas Used in Modern HPC Computation Steering // *2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBREIT)*, Yekaterinburg, Russia, 2020. P. 1–4. DOI: 10.1109/usbereit48449.2020.9117685.
27. Abelson H., Sussman G.J., Sussman J. The Elements of Programming // *Structure and Interpretation of Computer Programs*. MIT Press/McGraw-Hill, Cambridge, 1996.
28. Надуев А.Г., Черевань А.Д., Лебедева А.С. Архитектура программного модуля «ЛОГОС ПЛАТФОРМА» // *Вопросы атомной науки и техники. Серия: Математическое моделирование физических процессов*. 2022. № 4. С. 55–63. DOI: 10.53403/24140171_2022_4_55.
29. Klasky S. et al. ADIOS 2: The Adaptable Input Output System. A framework for high-performance data management // *SoftwareX*. 2020. Vol. 12. DOI: 10.1016/j.softx.2020.100561.
30. Averboukh Y. Lattice approximations of the first-order mean field type differential games // *Non-linear Differ. Equ. Appl.* 2021. Vol. 28, no. 65. DOI: 10.1007/s00030-021-00727-2.
31. Киреев С.Е., Литвинов В.С. Анализ исполнения фрагментированных программ на основе факторов SLOW // *Параллельные вычислительные технологии – XVIII всероссийская конференция с международным участием, ПаВТ'2024, г. Челябинск, 2-4 апреля 2024 г. Короткие статьи и описания плакатов*. Челябинск: Издательский центр ЮУрГУ, 2024. С. 94–107. DOI: 10.14529/pct2024.
32. Durner D., Leis V., Neumann T. On the Impact of Memory Allocation on High-Performance Query Processing // *Proceedings of the 15th International Workshop on Data Management on New Hardware (DaMoN'19)*. ACM, 2019. Article 21. P. 1–3. DOI: 10.1145/3329785.3329918.
33. Васёв П.А. Среда параллельного программирования Parallel Programming Kit // *Тезисы докладов Национального суперкомпьютерного форума (НСКФ-2024), 26-29 ноября 2024, Институт программных систем им. А.К. Айламазяна РАН, Переславль-Залесский*.
34. Marquer Y., Gava F. Algorithmic Completeness of BSP Languages. Technical Report Laboratoire d'Algorithmique, Complexité et Logique, Université Paris-Est Créteil. 2018.

Комплекс средств высокоуровневого синтеза «Тесей» для многокристальных реконфигурируемых вычислительных систем

А.И. Дордопуло¹, И.И. Левин^{1,2}, В.А. Гудков^{1,2}, А.А. Гуленок¹

¹Общество с ограниченной ответственностью «НИЦ супер-ЭВМ и нейрокомпьютеров»,
г. Таганрог, Россия

²Федеральное государственное автономное образовательное учреждение высшего
образования Южный федеральный университет, г. Таганрог, Россия

В статье рассматривается программный комплекс «Тесей», преобразующий последовательную программу на языке высокого уровня С в конфигурационные файлы программируемых логических интегральных схем (ПЛИС). В отличие от известных средств высокоуровневого синтеза, создающих из последовательной программы автономное IP-ядро, «Тесей» создает параллельно-конвейерное решение задачи для заданного пользователем ресурса реконфигурируемой вычислительной системы (РВС), содержащей множество ПЛИС, соединенных пространственной коммутационной системой. Для этого компоненты комплекса строят информационный граф из последовательной программы, преобразуют его в кадровую структуру, которая формальными методами в автоматическом режиме без участия пользователя адаптируется к заданной конфигурации РВС. Отличительной особенностью подхода является существенно меньшее по сравнению с распараллеливающими компиляторами число анализируемых вариантов, что позволяет сократить время портирования последовательных программ. Описана новая программа «Ариадна» для синтеза стратегий преобразования программ разных предметных областей и разработанные решения для распараллеливания косвенной адресации, часто встречающейся в последовательных программах.

Ключевые слова: высокоуровневый синтез, HLS, трансляция программ, язык С, редукция производительности, реконфигурируемая вычислительная система, программирование многопроцессорных вычислительных систем.

1. Введение

Средства высокоуровневого синтеза (High Level Synthesis) или HLS-компиляторы [1, 2] предназначены для преобразования программ на одном из языков высокого уровня в конфигурационные файлы специализированных аппаратных средств на языках описания аппаратуры HDL (Hardware description language). Использование HLS-компиляторов позволяет снизить требования к квалификации разработчиков для вычислительных систем с реконфигурируемой элементной базой, например, построенных на основе программируемых логических интегральных схем (ПЛИС). HLS-компиляторы предназначены для быстрой разработки проектов, где важна скорость создания конфигурационных файлов, а не эффективность кода и/или сокращение аппаратных затрат. Большинство средств высокоуровневого синтеза преобразуют вычислительно трудоемкий фрагмент программы на языке С в сложный функциональный блок (IP-ядро), представляющий собой цифровой автомат или специализированный процессор, выполняющий вычисления быстрее за счет прямой реализации информационных зависимостей и свойств архитектуры ПЛИС даже при существенной (по сравнению с микропроцессорами) разнице в тактовой частоте работы.

Автоматизация создания IP-ядра HLS-компилятором облегчает портирование вычислений в архитектуру ПЛИС, но не гарантирует их эффективную реализацию, т.к. организация потоков данных возлагается на программиста, а средства масштабирования и синхронизации решения (хотя бы в пределах одного кристалла) отсутствуют. Для реконфигурируемых вычислительных систем

(PBC) [3], содержащих множество ПЛИС, связанных пространственной коммутационной системой, настройка синхронизации и согласования одновременной работы множества IP-ядер является сложным и трудоемким процессом, сравнимым по сложности с разработкой нового схемотехнического проекта.

Для преобразования последовательных программ на языке C в стандарте ISO/IEC 9899:1999 в решения многокристальных PBC с автоматической синхронизацией информационных и управляющих сигналов для всех используемых ПЛИС предназначен описываемый в настоящей статье комплекс средств высокоуровневого синтеза «Тесей» [4, 5].

В статье представлены результаты работ по созданию программного комплекса высокоуровневого синтеза [4, 5] с автоматическим преобразованием последовательных программ к доступному аппаратному ресурсу PBC: описана новая программа «Ариадна» для синтеза стратегий преобразования программ разных предметных областей и разработанные решения для распараллеливания косвенной адресации, часто встречающейся в последовательных программах. Статья организована следующим образом. Во втором разделе представлен краткий обзор известных HLS-компиляторов и их особенностей. Описание новой структуры комплекса и взаимосвязей между его компонентами, возможностей и ограничений представлено в третьем разделе. В четвертом разделе описана новая программа синтеза стратегий «Ариадна». В пятом разделе представлено описание разработанных преобразований для распараллеливания косвенной адресации. В заключении обобщаются полученные результаты и формулируются направления дальнейших исследований.

2. Краткий обзор проблемной области

Среди наиболее распространенных средств высокоуровневого синтеза можно выделить академические (DWARV [6], BAMBU [7], LEGUP [8]) и коммерческие (CatapultC, Vivado HLS [9], Vivado Vitis [10]) HLS-компиляторы, подробные обзоры возможностей, ограничений и отличительных характеристик которых представлены в [1, 2, 11–13]. Несмотря на различия в методах преобразования и форматах входных и выходных данных, большинство перечисленных компиляторов обладает следующими общими чертами:

- аппаратный ресурс для синтезируемого проекта ограничен одним кристаллом ПЛИС;
- для получения эффективного решения необходима выполняемая разработчиком предварительная разметка кода (например, директивами `#pragma` в Vivado HLS и Vitis), что требует детальных знаний о структуре и семантике транслируемой программы и синтезируемой реализации;
- использование и управление синтезированным IP-ядром осуществляется программистом самостоятельно, без поддержки со стороны среды разработки.

Синтез HLS-компиляторами однокристалльных решений связан с широким распространением специализированных или отладочных плат, содержащих одну микросхему ПЛИС. В отличие от них комплекс «Тесей» разрабатывался для многокристальных PBC, что изначально предполагало создание не единичного IP-ядра, а поддержки различных форм организации вычислений для большого количества вычислительных устройств, требующих синхронизации для совместной согласованной работы. Синтез таких решений существенно сложнее, поскольку требует анализа и выбора рациональных вариантов из множества возможных, поэтому по сложности сравним с автоматическим распараллеливанием.

Комплекс средств высокоуровневого синтеза «Тесей», при схожей с Xilinx Vivado HLS и Vitis функциональности, обладает следующими основными отличиями:

- преобразование осуществляется на основе анализа информационных зависимостей входной программы, а не директив ручной разметки кода, аналогичных `#pragma` в Vivado HLS;
- для высокоуровневого синтеза и масштабирования его результатов параметры параллелизма кадровой структуры сокращаются, а не распараллеливаются, как для последовательной программы;
- произвольный доступ к памяти в последовательной программе преобразуется к регулярным операциям чтения-записи для потоков данных;
- эффективность синтезируемого решения может быть повышена различными способами организации вычислений (макроконвейер, конвейер в конвейере);

- автоматическая адаптация прикладной задачи к архитектуре и конфигурации заданной пользователем PBC;
- автоматическая синхронизация информационных и управляющих сигналов для всех используемых кристаллов ПЛИС;
- поддержка задач разных прикладных областей.

Входная программа на языке C для текущей версии комплекса «Тесей» должна соответствовать определенным требованиям:

- программа должна быть написана без использования операторов безусловного перехода goto, конструкций union и struct, функций strlen и функций работы с памятью (malloc/calloc, mem-cpy и др), а также некоторых директив условной компиляции для кроссплатформенных программ вида #if (Def_WinOrLin == 1);
- для определения входных и выходных переменных в тексте программы должны **обязательно** присутствовать процедуры загрузки и сохранения данных с фиксированными именами (load_data/save_data, load_data32/save_data32, load_data64/save_data64), соответствующие заданному шаблону;
- желательно избегать использования операторов break и return для немедленного выхода из циклов, а также использования переменных с косвенной адресацией, которую нельзя рассчитать на этапе трансляции, для счетчиков циклов и индексации массивов. Информационная эквивалентность для входных программ с такими операторами не гарантируется;
- желательно, чтобы текст входной программы был написан в процедурном стиле с наибольшей наглядностью, без применения возможных в языке C оптимизаций для реализуемого алгоритма.

3. Структура программного комплекса «Тесей»

Для эффективной реализации задач различных проблемных областей комплекс средств высокоуровневого синтеза «Тесей», в отличие от своих предыдущих версий [4, 5], дополнен компонентом синтеза стратегий преобразования параллельно-конвейерной программы «Ариадна». Структура новой версии комплекса приведена на рис. 1.

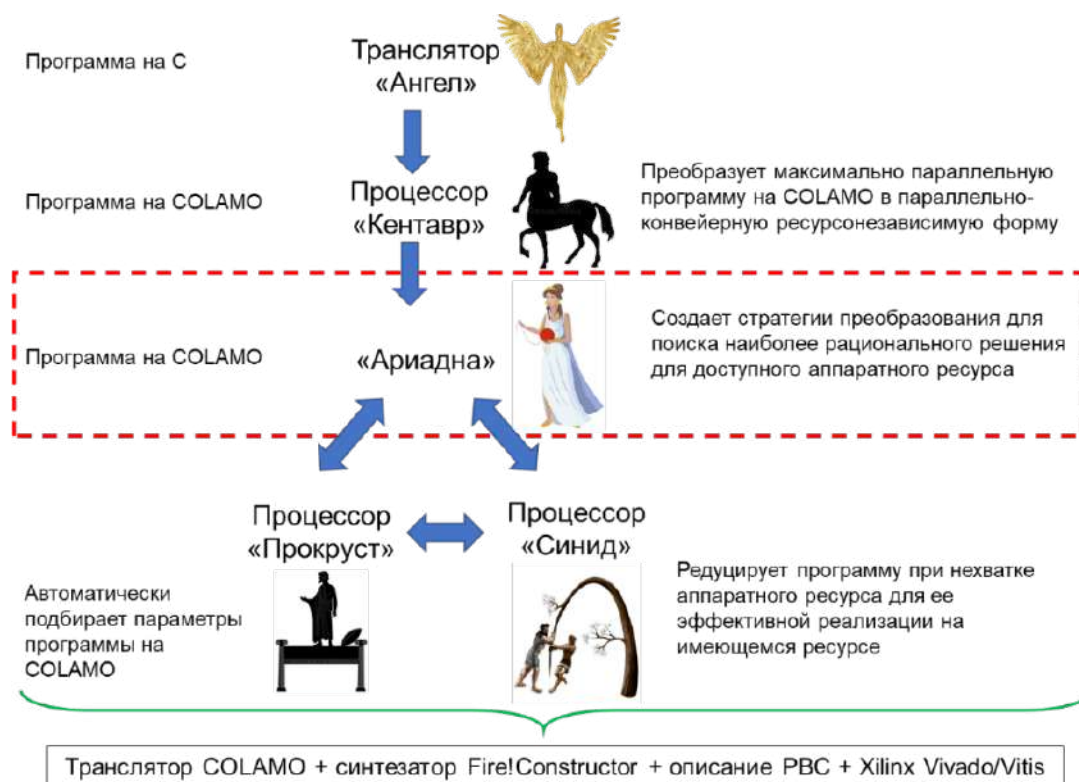


Рис. 1. Структура взаимодействия компонентов программного комплекса

«Ариадна» выполняет часть функций, ранее реализованных в «Прокрусте», и ряд новых преобразований для синтеза рациональных способов организации вычислений (макроконвейер, конвейер в конвейере). Первые этапы преобразования последовательной программы не претерпели изменений: как и в предыдущих версиях, «Ангел» преобразует входную программу на языке C в максимально параллельную кадровую структуру, описанную конструкциями языка программирования с неявным описанием параллелизма COLAMO, а «Кентавр» преобразует максимально параллельную кадровую структуру в ресурсонезависимую параллельно-конвейерную программу на языке COLAMO. Далее «Ариадна» определяет возможные стратегии преобразования ресурсонезависимой программы для поиска наиболее рационального решения на доступном аппаратном ресурсе PBC. Согласно стратегиям, синтезированным «Ариадной», «Прокруст» преобразует ресурсонезависимую программу и подбирает параметры каждой кадровой структуры для ее рациональной реализации в архитектуре заданной PBC. Далее, из всех синтезированных вариантов параллельно-конвейерной программы на COLAMO «Ариадной» выбирается наиболее рациональный по времени решения и занимаемому аппаратному ресурсу. «Синид», редуцирующий производительность кадровой структуры, выполняет вспомогательную роль и вызывается при необходимости либо «Прокрустом», либо «Ариадной», если его участие предусмотрено синтезированной стратегией.

Результатом работы «Ариадны» и «Прокруста» является программа на языке высокого уровня COLAMO с заданным (или наиболее близким к нему) уровнем реальной производительности на доступном вычислительном ресурсе. Транслятор языка программирования COLAMO транслирует программу в конфигурационные файлы ПЛИС многокристалльных PBC, а синтезатор многокристалльных решений Fire!Constructor выполняет автоматическую синхронизацию распределенных по разным кристаллам фрагментов решения. После этого синтезатор Xilinx Vivado создает загрузочные конфигурационные файлы (*.bit) для каждого кристалла.

4. Стратегии преобразования кадровых структур: программа «Ариадна»

Входными данными для «Ариадны» является полученная от «Кентавра» параллельная программа на языке COLAMO со структурной реализацией базового подграфа, у которой все константы масштабирования кадровой структуры равны 1. «Ариадна» оценивает аппаратные затраты на реализацию этой программы в PBC по всем компонентам ПЛИС (LUT, FlipFlop, DSP, BRAM и др.) и определяет критический аппаратный ресурс для решаемой задачи. Кроме этого, выполняется анализ интервала обработки данных, количества каналов распределенной памяти и латентности. Далее, в соответствии с методикой преобразования кадровых структур [4], после выбора критического ресурса «Ариадна» определяет последовательность (или стратегию) выполнения преобразований входной кадровой структуры с учетом особенностей ее информационных зависимостей и базового подграфа.

Для выбора преобразований «Ариадна» анализирует динамику нелинейного изменения аппаратных затрат $A_{CS} = (a_1^{CS}, a_2^{CS}, \dots, a_n^{CS})$ кадровой структуры при изменении основных параметров быстродействия кадровой структуры $P_{CS} = (p_1, p_2, \dots, p_n)$ с учетом информационных зависимостей внутри и между подграфами (фрагментами) задачи. В соответствии с методикой [4], «Ариадна» анализирует параметры быстродействия кадровой структуры P_{CS} : L_i^{it} – число информационно-независимых подграфов в слое F_{ij} , It_i^{it} – число итераций в функции Φ_i , Ch_i^{it} – число внешних каналов кадровой структуры, Op_i^{it} – число устройств кадровой структуры FG^{it} , ρ_i^{it} – разрядность обрабатываемых данных, способ аппаратной реализации базового подграфа g_i и интервал обработки данных I . Каждый из этих параметров, в зависимости от предметной области, структуры и размерности решаемой задачи, может являться определяющим для критического аппаратного ресурса преобразуемой программы. Выбор первого изменяемого параметра быстродействия, как правило, определяет остальные преобразования стратегии.

Для выбранного критического аппаратного ресурса «Ариадна» определяет границы и рациональный шаг изменения параметров быстродействия кадровой структуры

$P_{CS} = \{L_i^{it}, It_i^{it}, Ch_i^{it}, Op_i^{it}, \rho_i^{it}, g_i\}$, а также список возможных и/или необходимых оптимизационных преобразований (конвейер в конвейере, макроконвейер, автоподстановка) для синтеза потенциально рациональной кадровой структуры. Таким образом, «Ариадна» определяет последовательность выполнения и основные параметры преобразований. «Прокруст» выполняет все преобразования кадровой структуры в соответствии с переданной ему стратегией как единую транзакцию, чтобы сократить накладные временные расходы на многократный вызов программы. Результатом реализации стратегии является создаваемая «Прокрустом» кадровая структура.

При наличии нескольких созданных «Ариадной» стратегий результирующие кадровые структуры помещаются в список, для всех элементов которого анализируются эффективность и время решения задачи, после чего выбирается наиболее рациональная кадровая структура, которая и является решением.

Одной из основных задач при формировании стратегии являются исключение заведомо неэффективных направлений движения в пространстве кадровых структур [5] и сокращение количества синтезируемых вариантов, каждый из которых должен удовлетворять заданным критериям эффективности – производительность не должна быть ниже 60 % от пиковой.

Основная стратегия преобразования состоит в структурной реализации базового подграфа и последующем изменении влияющих на параллелизм и занимаемый ресурс параметров кадровой структуры. При масштабировании сначала изменяется количество слоев – до ограничений максимального числа каналов в системе, затем – итерации – до ограничений доступного аппаратного ресурса, потом выполняется сокращение интервала обработки данных с помощью автоподстановки и/или других оптимизационных преобразований. Эта стратегия не является непременно обязательной к выполнению, она может быть изменена в зависимости от типа и предметной области преобразуемой задачи. Выбор такого порядка выполнения преобразований обусловлен следующими причинами:

- масштабирование (распараллеливание) по слоям линейно повышает производительность;
- распараллеливание по итерациям повышает производительность близко к линейной, но зачастую требует буферизации, которая в ряде случаев может достигать половины длины обрабатываемого потока данных;
- типовые методы сокращения интервала обработки данных – автоподстановка и замена переменных – существенно, в три–четыре раза увеличивают аппаратные затраты, а производительность при этом растет нелинейно.

Если в результате выполнения всех преобразований стратегии полученное решение удовлетворяет требованиям занимаемого ресурса и получен заданный уровень производительности, то цель считается достигнутой, и дальнейшие преобразования прекращаются. В противном случае, если заданный уровень производительности не достигнут, происходит изменение приоритетов выполнения преобразований и переход к следующей стратегии.

Если базовый подграф кадровой структуры входной параллельной программы поместился в доступный аппаратный ресурс, но не достигнут заданный уровень производительности (обычно 60 %), необходимо снова определять стратегии. Порядок выполнения преобразований значительно зависит от типа и предметной области решаемой задачи.

Например, при реализации вложенных конвейеров сокращение интервала обработки данных становится даже более эффективным, чем распараллеливание по слоям. Причиной этого является линейный рост производительности решения практически без дополнительных аппаратных затрат (добавляется лишь цепочка регистров в цепи обратной связи). Поэтому для задач, содержащих такие фрагменты, первым преобразованием стратегии целесообразно выполнять оптимизацию интервала обработки данных.

При решении задач цифровой обработки сигналов распараллеливание по слоям при увеличении аппаратного ресурса, как ни странно, приводит к нелинейному (сублинейному) росту производительности. Это связано с тем, что увеличение числа каналов обработки требует динамической коммутации, что усложняет не только коммутацию, но и адресацию памяти, что, в свою очередь, способствует разрывам информационных потоков. Поэтому порядок выполнения преобразований для таких задач целесообразно изменить по сравнению с базовой стратегией и первым преобразованием выполнить редукцию по числу каналов связи.

В ряде сложных задач полная реализация базового подграфа со всеми операционными вершинами требует огромного числа информационных каналов, что не позволяет применять распараллеливание по слоям. В этом случае даже при достаточности аппаратного ресурса для реализации базового подграфа его необходимо редуцировать, чтобы сократить количество каналов, которые являются критическим ресурсом.

Стратегия преобразования кадровой структуры входной параллельной программы не фиксирована и в значительной степени определяется информационной структурой задачи и типом информационных зависимостей. Так, для целого ряда задач символьной обработки характерно отсутствие итераций, но число слоев крайне велико, поэтому преобразования итерационной структуры для таких задач лишены смысла. При этом в задачах математической физики рациональная реализация итерационной составляющей кадровой структуры является залогом эффективности решения всей задачи. При решении задач цифровой обработки сигналов зачастую первым преобразованием является редукция количества входных информационных каналов с переходом к м-кадрам и последующим масштабированием полученного решения для эффективного использования аппаратного ресурса.

В ряде случаев аппаратного ресурса РВС может быть недостаточно для структурной реализации базового подграфа. Тогда к базовому подграфу применяются редукционные преобразования [5], что позволяет впоследствии использовать методы масштабирования редуцированной кадровой структуры для сокращения времени решения задачи. Если необходима редукция функционального подграфа задачи, «Ариадна» рассчитывает требуемый коэффициент редукции и вызывает вспомогательный компонент – программу преобразования фрагментов кадровой структуры на языке COLAMO в микро-кадры «Синид». В текущей версии программы «Ариадна» реализованы не все указанные в [5] преобразования – в частности, не поддерживается редукция по разрядности обрабатываемых операндов, которая дает ощутимый для быстродействия эффект только для достаточно узкого класса задач.

Таким образом, в качестве первого преобразования кадровой структуры в стратегии может использоваться изменение любого из пяти параметров параллелизма (L, It, Ch, Op, I), что и определяет различие стратегий. Оценкой сверху для общего количества возможных стратегий тогда является $5! = 120$, т.е. их число достаточно невелико даже для полного перебора современными средствами вычислительной техники. На практике, в силу ортогональности преобразований кадровой структуры [5], количество возможных стратегий существенно меньше и едва ли превышает 20. Используемый в «Ариадне» алгоритм синтеза стратегий реализован в соответствии с методикой преобразования (портации) кадровой структуры [4], причем выбор последовательности выполняемых преобразований зависит от информационных зависимостей в структуре задачи, параметров быстродействия кадровой структуры и доступного аппаратного ресурса. Для оптимизации алгоритма синтеза стратегий также использовался многолетний опыт разработки схемотехнических решений специалистов НИЦ СЭ и НК.

Сравнительно небольшое общее число возможных стратегий преобразования кадровых структур позволяет существенно сократить время анализа кадровых структур и время синтеза **гарантированно** рационального решения прикладной задачи: если при полном переборе всех стратегий не был достигнут заданный уровень производительности, будет выбрано лучшее из проанализированных решений.

Сравнение результатов высокоуровневого синтеза шести задач символьной обработки и четырех задач линейной алгебры без использования «Ариадны» и с ее применением показало, что количество синтезируемых конвейерных реализаций не сокращается, а время синтеза для ряда задач сокращается на 15–50 %.

5. Структурная реализация косвенной адресации

Структурная реализация косвенной адресации при потоковой обработке является одной из самых «неудобных» задач, эффективная реализация которой затруднена, поскольку произвольный порядок следования адресов памяти нарушает необходимую для конвейерной организации вычислений регулярность работы контроллеров распределенной памяти. В наиболее общем случае структурная реализация косвенной адресации для получения элемента массива X по косвен-

ному адресу $Y[i]$ представляет собой каскад последовательно соединенных мультиплексоров, количество которых определяется длиной массива. Схема структурной реализации косвенной адресации приведена на рис. 2. Получение элемента $X[Y[i]]$ из массива X сводится к последовательному сравнению поступившего адреса $Y[i]$ с адресами всех элементов массива X , в результате которого сработает только один из мультиплексоров в цепочке. Как видно из рис. 2, структурная реализация косвенной адресации неизбежно приводит к нерациональному использованию аппаратного ресурса и существенному снижению производительности РВС.

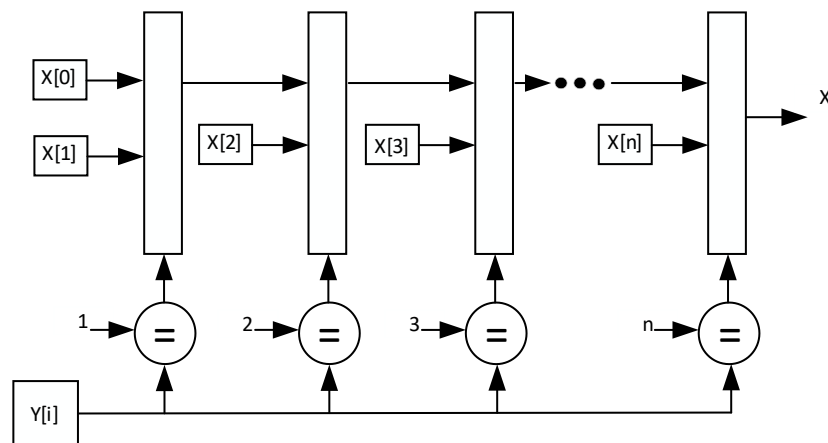


Рис. 2. Структурная реализация косвенной адресации для получения элемента из массива X по косвенному адресу $Y[i]$

Вместе с тем, на практике нередко встречаются частные случаи, когда для выражений, содержащих косвенную адресацию, возможно сократить время обращения к элементу массива и даже распараллелить. Наиболее частым примером является суммирование с накоплением, когда выполняется доступ к одному и тому же элементу массиву с совпадающими адресами чтения и записи: $X[Y[i]] := X[Y[i]] + \text{const}$. Выражения такого вида в ряде случаев могут быть распараллелены по адресам, по данным или по адресам и данным одновременно.

Распараллеливание косвенной адресации по адресам состоит в одновременной обработке двух и более адресов $Y[i]$ для доступа к нескольким «копиям» исходного массива X (блокам $X0$ и $X1$ на рис. 3). При структурной реализации этого метода цепочка мультиплексоров заменяется внутренней памятью, количество блоков которой определяется степенью распараллеливания адресного массива Y . Необходимым условием применения метода распараллеливания по адресам является существование нейтрального элемента для операции в выражении с косвенной адресацией (нейтральный элемент оставляет любой другой элемент неизменным при применении к ним бинарной операции). Для операции арифметического сложения нейтральным элементом является 0, а для операции умножения – единица. Существование нейтрального элемента для выполняемой операции необходимо для предварительного заполнения его значением блоков внутренней памяти $X0$ и $X1$ на рис. 3.

При одновременной подаче адресов $Y0[i]$ и $Y1[i]$ выполняется чтение текущих значений из блоков памяти $X0$ и $X1$, их обработка (в рассматриваемом примере – сложение с константой K) и запись результата по тому же адресу в блоки внутренней памяти $X0$ и $X1$. Адреса в каналах $Y0$ и $Y1$ могут совпадать, совпадающие адреса будут обработаны информационно-независимо, но для получения итогового корректного результата, по завершению обработки всех адресов $Y0[i]$ и $Y1[i]$ необходимо объединить накопленные в блоках $X0$ и $X1$ частичные результаты – на рис. 3 это выполняется с помощью сумматора во втором ярусе. Учитывая возможную начальную инициализацию массива X , далее необходимо объединить его начальные значения с объединенными частичными результатами, что выполняет сумматор в последнем ярусе.

Для реализации предложенного метода распараллеливания косвенной адресации необходимы как дополнительные каналы распределенной памяти для одновременной подачи адресов Y , так и дополнительные аппаратные затраты в виде блоков внутренней памяти по размеру исходного массива.

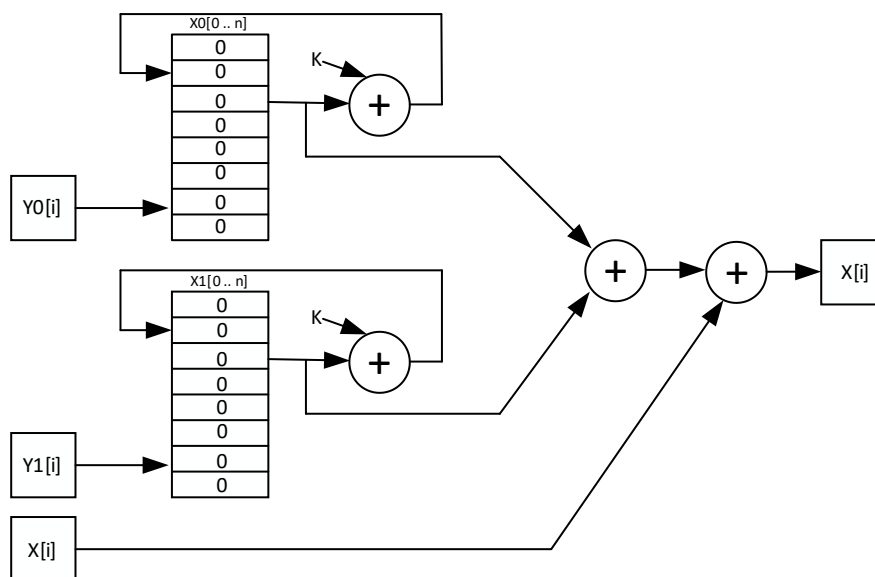


Рис. 3. Структурная реализация косвенной адресации при распараллеливании по адресам

Альтернативой рассмотренному методу является распараллеливание по данным, схема структурной реализации которого приведена на рис. 4. В отличие от рассмотренного на рис. 3 случая, исходный массив X делится на несколько непересекающихся частей, каждая из которых размещается в отдельном блоке внутренней памяти. Количество блоков внутренней памяти соответствует степени распараллеливания и совпадает с количеством входных адресных каналов Y . В отличие от предыдущей схемы подаваемые адреса $Y0[i]$ и $Y1[i]$ подаются не напрямую, а буферизируются в зависимости от своего значения с помощью коммутатора. Буферизация необходима для разрешения коллизии одновременного доступа к памяти, когда одновременно пришедшие адреса обращаются к одному и тому же блоку внутренней памяти.

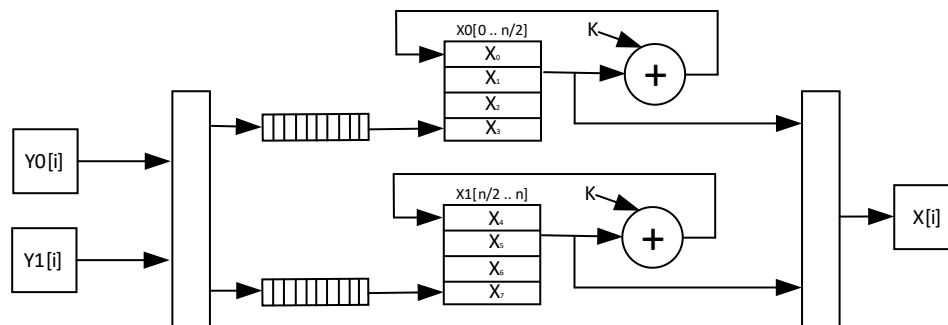


Рис. 4. Структурная реализация косвенной адресации при распараллеливании по данным

Представленная на рис.4 схема требует дополнительных аппаратных расходов на буферизацию, причем рекомендуемый размер каждого буфера в общем случае равен $M/2$, где M – размерность массива Y . Если распределение адресов в массиве Y заранее неизвестно или существенно отличается от нормального, то использование буферов меньшей длины теоретически может привести к их переполнению. Поэтому при выборе рационального размера буфера, с одной стороны, необходимо минимизировать размер буфера для сокращения аппаратных затрат, а с другой – обеспечить корректность работы, исключив возможность их переполнения. Как правило, размер буфера обычно подбирается эмпирически для каждой прикладной программы с учетом характеристик распределения адресов в массиве Y . В комплексе «Тесей» в настоящий момент реализован вариант с буфером размером $M/2$ как в наибольшей степени обеспечивающий корректность обработки данных.

Распараллеливание косвенной адресации для вычислительных операций с накоплением результата (суммирование, умножение) является одним из инструментов «Ариадны» для оптимизации организации вычислений и синтеза рациональной кадровой структуры.

6. Заключение

В комплексе средств высокоуровневого синтеза «Тесей», в отличие от известных HLS-компиляторов, входная программа на языке C преобразуется автоматически, без ручной разметки кода или иных указаний пользователя, а результатом является рациональное решение задачи для многокристальной PBC с автоматической синхронизацией информационных и управляющих сигналов.

В настоящей статье описана новая программа «Ариадна», выполняющая синтез стратегий преобразования кадровой структуры для поиска рационального решения исходной прикладной задачи. В отличие от предыдущих версий комплекса [4, 5] «Ариадна» выбирает преобразования кадровой структуры адаптивно, учитывая особенности ее информационных зависимостей и базового подграфа, что позволяет синтезировать рациональные кадровые структуры для различных предметных областей: символьной обработки, линейной алгебры, цифровой обработки сигналов.

Направления дальнейших исследований связаны с расширением классов и диапазонов решаемых задач и сокращением ряда требований к исходному тексту программ на C, в частности, поддержки косвенной адресации во входной программе и операторов динамического выделения памяти для массивов.

Литература

1. Nane R., Sima V.-M., Pilato C. et al. A Survey and Evaluation of FPGA High-Level Synthesis Tools // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2016. Vol. 35, no. 10. P. 1591–1604. DOI: 10.1109/TCAD.2015.2513673.
2. Numan M.W., Phillips B.J., Puddy G.S., Falkner K. Towards Automatic High-Level Code Deployment on Reconfigurable Platforms: A Survey of High-Level Synthesis Tools and Toolchains // IEEE Access. 2020. Vol. 8. P. 174692–174722. DOI: 10.1109/ACCESS.2020.3024098.
3. Каляев И.А., Левин И.И. Реконфигурируемые вычислительные системы на основе ПЛИС: монография. Ростов-на-Дону: Издательство ЮНЦ РАН, 2022. 474 с. ISBN 978-5-4358-0232-0
4. Дордопуло А.И., Левин И.И., Гудков В.А., Гуленок А.А. Программный комплекс высокоуровневого синтеза конфигурационных файлов для многокристальных реконфигурируемых вычислительных систем // Параллельные вычислительные технологии (ПаВТ'2023): Короткие статьи и описания плакатов. Материалы XVII всероссийской научной конференции с международным участием, Санкт-Петербург, 28-30 марта 2023 года. Челябинск: Издательский центр ЮУрГУ, 2023. С. 133–142.
5. Дордопуло А.И., Левин И.И., Гудков В.А., Гуленок А.А. Программные средства высокоуровневого синтеза для многокристальных реконфигурируемых вычислительных систем // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 3. С. 5–21. DOI: 10.14529/cmse220301.
6. Nane R., Sima V.-M., Olivier B. et al. DWARV 2.0: A CoSy-based C-to-VHDL Hardware Compiler // FPL. 2012. P. 619–622. DOI: 10.1109/FPL.2012.6339221.
7. Pilato C., Ferrandi F. Bambu: A Modular Framework for the High Level Synthesis of Memory-intensive Applications // FPL. 2013. P. 1–4. DOI: 10.1109/FPL.2013.6645550.
8. Canis A., Choi J., Aldham M. et al. LegUp: High-Level Synthesis for FPGA-based Processor/Accelerator Systems // ACM FPGA. 2011. P. 33–36. DOI: 10.1145/1950413.1950423.
9. Make Slow Software Run Fast with Vivado HLS. URL: <https://www.xilinx.com/publications/xcellonline/run-fast-with-Vivado-HLS.pdf> (дата обращения: 23.03.2023)
10. Vitis Unified Software Platform Documentation. Application Acceleration Development. URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug1393-vitis-application-acceleration.pdf (дата обращения: 23.03.2023).

11. Kamkin A.S., Chupilko M.M., Lebedev M.S. et al. Comparison of High-Level Synthesis and Hardware Construction Tools // Trudy ISP RAN/Proc. ISP RAS. 2022. Vol. 34, no. 5. P. 7–22. DOI: 10.15514/ISPRAS-2022-34(5)-1.
12. Liang Y., Rupnow K., Li Y. et al. High-Level Synthesis: Productivity, Performance, and Software Constraints // Journal of Electrical and Computer Engineering. 2012. Article 1. DOI: 10.1155/2012/649057.
13. Zhou Y., Gupta U., Dai S. et al. Rosetta: A Realistic High-Level Synthesis Benchmark Suite for Software Programmable FPGAs // Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '18). Association for Computing Machinery, New York, NY, USA, 2018. P. 269–278. DOI: 10.1145/3174243.3174255.

Оценка степени манипулируемости известных процедур агрегирования для большого числа альтернатив и участников

Ю.В. Зонтов, А.А. Иванов, Д.С. Карабекян, В.И. Якуба

НИУ ВШЭ

В данной работе рассматривается проблема манипулирования, возникающая во время процедуры голосования. Представлены реализованные подходы к переработке существующих последовательных алгоритмов для обобщенных скоринговых процедур агрегирования, а также оценок степени манипулируемости известных процедур агрегирования для большого числа участников и альтернатив с использованием подхода параллелизма данных на суперкомпьютерном кластере ВШЭ. Обсуждаются полученные результаты расчетов и пути решения выявленных трудностей.

Ключевые слова: скоринговые процедуры, процедуры агрегирования, манипулируемость.

1. Введение

Манипулирование – это ситуация, которая возникает при голосовании, когда участник или группа участников записывают в бюллетень неискренние предпочтения, чтобы достичь для себя более выгодного для них результата голосования.

Процедура агрегирования – это правило вычисления результата голосования на основе выборов участников голосования.

Около 50 лет назад было доказано, что любая недиктаторская процедура агрегирования является манипулируемой [1, 2], после чего возник вопрос, какая процедура агрегирования является наименее манипулируемой [3]. Для решения этой задачи в литературе чаще всего используются два подхода: первый подход предполагает выведение точных [4] или приближенных [5] формул, второй подход предполагает использование компьютерного моделирования для поиска наименее манипулируемых процедур агрегирования [6, 7].

Для анализа степени манипулируемости используется индекс Нитцана-Келли (НК-индекс) [8, 9]:

$$NK = \frac{P_0}{P},$$

где P_0 – количество всех профилей, в которых хотя бы один участник успешно манипулирует, P – число всех возможных профилей предпочтений участников. Таким образом, НК-индекс – это доля всех манипулируемых профилей.

Для вычисления индекса Нитцана-Келли необходимо для каждого профиля сгенерировать все возможные попытки манипулирования, затем, если хотя бы одна попытка манипулирования является успешной, то есть результат агрегирования для манипулирующего участника или группы участников лучше согласно их расширенным предпочтениям, то профиль помечается как манипулируемый. Если для профиля не существует ни одной успешной попытки манипулирования, то он помечается как неманипулируемый.

В данной работе рассматривается модель, в которой n участников голосования выбирают среди m альтернатив. У каждого участника есть предпочтение, выраженное линейным порядком, на множестве альтернатив, то есть одно из $m!$ предпочтений. Профиль P – это совокупность участников голосования и их предпочтений. Процедура агрегирования $C(P)$ ставит в соответствие заданному профилю результат процедуры голосования.

Существует два подхода к тому, каким может быть результат голосования $C(P)$: это одиночный и множественный выбор. В рамках концепции одиночного выбора предполагается, что

если между двумя или более альтернативами возникла ничья, то используется некоторый дополнительный механизм для определения победителя.

Один из популярных вариантов – использование алфавитного (лексикографического) принципа устранения несравнимости. При одиночном выборе процедура голосования дает на выходе одну альтернативу – победителя. Например, если альтернативы $\{a\}$ и $\{b\}$ набрали по 4 голоса, и возникла ничья, то выбор будет $\{a\}$ по алфавитному принципу.

В модели множественного выбора предполагается, что в случае возникновения ничьи между двумя или более альтернативами, все такие альтернативы будут входить в результат коллективного выбора. Например, если альтернативы $\{a\}$ и $\{b\}$ набрали по 4 голоса, то выбор будет $\{a, b\}$.

В литературе известно несколько десятков процедур агрегирования. В данной работе рассматриваются 5 основных групп процедур: Scoring Rules (Скоринговые), Majority Relation-Based Rules (Мажоритарные), Value Function Rules (Основанные на функции полезности), Tournament Matrix Rules (Турнирные), Q-Paretian Rules (q-Паретовские), а также обобщенные скоринговые правила.

В качестве примеров скоринговых правил рассмотрим две классические процедуры агрегирования: правило относительного большинства и правило Борда.

Правило относительного большинства: В выбор входят альтернативы, которые являются лучшими для наибольшего числа участников голосования.

Правило Борда: Каждой альтернативе x ставится в соответствие число, равное мощности множества альтернатив, худших, чем x при выбранных предпочтениях участника. Сумма данных значений по всем участникам называется рангом Борда для альтернативы x . В выбор входят альтернативы с максимальным рангом.

Исходных предпочтений участника (например, $a > b > c$) может оказаться недостаточно для сравнения результатов процедуры голосования до и после манипулирования. Например, возникает вопрос, если до манипулирования результат был $\{b\}$, а после он стал $\{a, c\}$, то стал ли этот результат лучше или хуже для участника? Является ли более предпочтительным получить гарантированно вторую лучшую альтернативу или ничью между лучшей и худшей альтернативами? Для ответа на этот вопрос в литературе используются расширенные предпочтения, которые являются линейным порядком на множестве всех возможных результатов процедуры агрегирования.

Для случая 3 альтернатив это 4 метода построения расширенных предпочтений (при предположении, что исходные предпочтения участника $a > b > c$):

1. Лексимин

$$\{a\} > \{a, b\} > \underline{\{b\}} > \underline{\{a, c\}} > \underline{\{a, b, c\}} > \{b, c\} > \{c\}$$

2. Лексимакс

$$\{a\} > \{a, b\} > \underline{\{a, b, c\}} > \underline{\{a, c\}} > \underline{\{b\}} > \{b, c\} > \{c\}$$

3. Рискофил

$$\{a\} > \{a, b\} > \underline{\{a, c\}} > \underline{\{a, b, c\}} > \underline{\{b\}} > \{b, c\} > \{c\}$$

4. Рискофоб

$$\{a\} > \{a, b\} > \underline{\{b\}} > \underline{\{a, b, c\}} > \underline{\{a, c\}} > \{b, c\} > \{c\}$$

В литературе используются две наиболее популярные модели вероятностей профилей: Impartial Culture (IC) и Impartial Anonymous Culture (IAC).

В модели IC предпочтения участников равновероятны. Для случая n участников и m альтернатив у каждого участника возможно одно из $m!$ предпочтений, значит, общее количество профилей равно $(m!)^n$. Для случая 3 альтернатив и 100 участников всего будет $6.5 * 10^{77}$ различных профилей.

В модели IAC предполагается, что профили равновероятны с учетом анонимности, то есть два профиля считаются одинаковыми, если их можно получить переименованием участников.

В модели IAC используются «единогласные профили», то есть профили, когда все участники имеют одинаковые или почти одинаковые предпочтения, будут более вероятны, чем в модели IC. Количество профилей в модели IAC для случая n участников и m альтернатив равно

$C_{m!+n-1}^n$. Для случая $n = 100$ и $m = 3$ общее количество профилей в ИАС составит 96.5 миллионов.

Стандартный механизм расчета манипулируемости состоит из следующих шагов [10]:

1. Генерация профилей;
2. Для каждого профиля – генерация всех возможных попыток манипулирования, то есть всех возможных изменений предпочтений манипулирующим участником или группой участников;
3. Подсчет нового результата коллективного выбора для каждой попытки манипулирования;
4. Расчет индекса NK: если хотя бы одна попытка манипулирования привела к более хорошему результату, то профиль считается манипулируемым, и неманипулируемым в противном случае.

Для случая ИС их количество равно $(m!)^n$, поэтому в предыдущих исследованиях в этой области, например [11], генерируются не все профили, а 1 миллион случайных профилей. Вторая проблема – высокая вычислительная сложность проверки всех профилей на манипулируемость. Вычислительная сложность равна количеству всех профилей умноженному на количество всех возможных попыток манипулирования и на алгоритмическую сложность вычисления результата процедуры голосования для профиля.

В работе [10] показана оценка сложности для метода с генерацией 1 миллиона случайных профилей, для случая 3 альтернатив и 100 участников время работы алгоритма составляло 63 часа на персональном компьютере. Если бы расчеты производились для 96.5 млн профилей (общее число профилей в модели ИАС для случая 100 участников), то расчет только одного случая со 100 участниками занял бы более 6000 часов (более 8 месяцев).

2. Постановка задачи

Как следует из предыдущего раздела, одной из ключевых проблем использования компьютерного моделирования для оценки степени манипулируемости процедур агрегирования является высокая сложность необходимых расчетов. Такая оценка предполагает решение задачи проверки множества профилей предпочтений участников на возможность успешного манипулирования. При этом количество всех возможных профилей предпочтений, как следствие, и сложность вычислений, быстро возрастают с ростом числа участников и альтернатив, что требует использования эффективных вычислительных схем [10].

Помимо вычислительной нагрузки, с ростом числа участников и альтернатив возрастает требуемый объем постоянной и оперативной памяти. Размерность задачи оказывается слишком высока для решения на персональных компьютерах, поэтому мы использовали суперкомпьютерный кластер ВШЭ [12] для проведения расчетов.

В качестве исходных материалов в данной работе выступали существующие однопоточные реализации алгоритмов расчета [10] оценки манипулируемости, разработанные сотрудниками Международного центра анализа и выбора решений на языке C#.

На первом этапе работы был выбран курс на переписывание данного программного кода на язык C++ с целью наиболее эффективного использования возможностей как суперкомпьютерного кластера, так и известного быстродействия самого языка. Данный подход применялся для алгоритмов индивидуального манипулирования применительно к числу альтернатив более 5 и показал хорошие перспективы внедрения, однако были выявлены важные недостатки, в первую очередь дополнительные временные затраты на сверку результатов расчета старой и новой версии программ, различия в количестве и устройстве доступных библиотек и структур данных для языков C++ и C#.

На втором этапе работы, при решении задачи распараллеливания алгоритма расчета оценки коалиционной манипулируемости обобщенных скоринговых правил, был выбран альтернативный подход. Было решено сделать запрос на установку на суперкомпьютерном кластере дополнительных программных модулей, а именно фреймворка .net core [13]. Данный подход позволил упростить и ускорить процесс переработки существующих программ для использования на суперкомпьютерном кластере.

3. Программная реализация и результаты вычислительных экспериментов

3.1 Расчет оценки манипулируемости для числа альтернатив более 5

Разработанная версия расчетной программы реализует подход к распараллеливанию на уровне данных (параллелизм данных). В рамках решения задачи определения манипулируемости профиля предпочтений для группы участников происходит создание «пула объектов» [14] класса `std::thread` библиотеки STL [15], которые последовательно отбирают из набора (1 миллион) профилей первый необработанный и после завершения работы возвращаются в «пул».

Данный подход показал превосходство над наивным подходом, когда полный набор профилей разделяется на равные доли между всеми вычислительными потоками. Он позволил устранить проблему простоя отдельных потоков (вычислительных ядер), который возникал из-за особенностей алгоритма вычисления степени манипулируемости правила агрегирования (НК-индекса). Алгоритм останавливается при нахождении первой модификации профиля, допускающей успешное манипулирование, и позволил довести средний уровень загрузки вычислительных ядер до 98%.

Также при реализации алгоритма удалось реализовать подход без блокировок, т.к. каждый поток сохраняет результат лишь в ассоциированную с ним (связь организуется по числовому идентификатору) ячейку памяти.

Данный подход хорошо показал себя при расчете до 30 участников для 3–7 альтернатив. На рис. 1 и далее приведены примеры расчетов для Правила относительного большинства (классической процедуры агрегирования, при использовании которой выбирается альтернатива, которая занимает первое место в предпочтениях максимального количества участников) и модели расширенных предпочтений Лексимин [16].

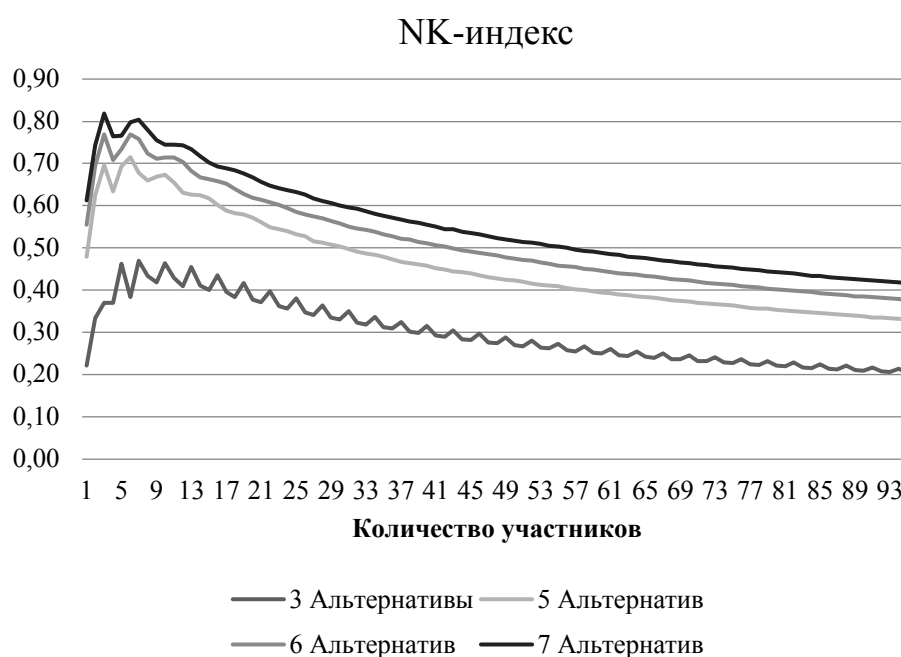


Рис. 1. График зависимости степени манипулируемости процедуры агрегирования (НК-индекс от числа участников для различного числа альтернатив) на примере Правила относительного большинства и расширенных предпочтений Лексимин

Благодаря примененным подходам к распараллеливанию алгоритмов, в рамках данной работы был впервые построен сравнительный график зависимости НК-индекса от числа участников для числа альтернатив более 5 (на примере Правила относительного большинства и расширенных предпочтений Лексимин). График представлен на рис. 2.

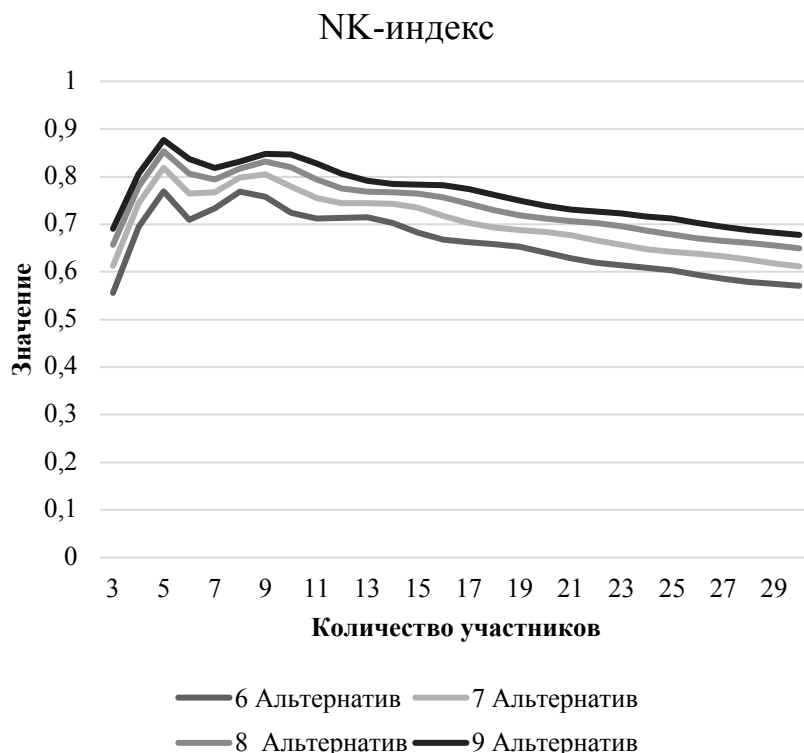


Рис. 2. График зависимости степени манипулируемости процедуры агрегирования (НК-индекс от числа участников для различного числа альтернатив) на примере Правила относительного большинства и расширенных предпочтений Лексимин до 9 альтернатив

Однако организация программы, при которой каждый задействованный вычислительный узел (32 вычислительных ядра) производил расчет для одного заданного числа участников, привело к тому, что каждый последующий расчет занимает больше времени, чем предыдущий (см. рис. 3).



Рис. 3. График зависимости времени расчета на одном узле (32 ядра) для Правила относительного большинства и 9 альтернатив от количества участников

Время расчетов также существенно возрастает с ростом числа альтернатив, что связано с увеличением размерности пространства профилей, когда для каждого из 1 миллиона профилей, производится перебор $(m!)^n$ – 1 вариаций, где m – количество альтернатив, а n – количество

участников. В табл. 1 и на рис. 4 приведены данные зависимости времени расчета НК-индекса для 25 участников, Правило относительного большинства, расширенное предпочтение Лексин-мин.

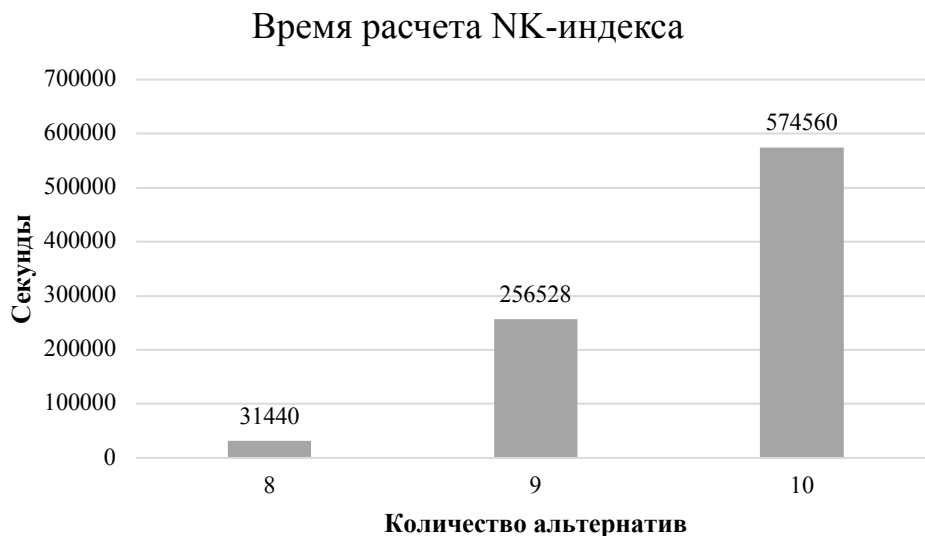


Рис. 4. График зависимости времени расчета на одном узле (32 ядра) для Правила относительного большинства и 25 участников

Таблица 1. Зависимость времени расчета на одном узле (32 ядра) для Правила относительного большинства и 25 участников от числа альтернатив.

*Для 10 участников приведена оценка времени, т.к. расчет завершился по таймауту

Количество альтернатив	Время расчета НК-индекса	Время расчета НК-индекса (секунды)
8	8ч 44м 00с	31440
9	2д 23ч 15м 28с	256528
10	Более 3д 7ч 48м 00с	Более 287280 (574560*)

Таким образом, даже для относительно небольшого числа участников 32 ядра узла вычислительного кластера затрачивают дни машинного времени для расчета индекса манипулируемости на 1 миллионе профилей. Данное ограничение не позволяет использовать существующий подход для выполнения расчетов для числа альтернатив более 9 и числа участников более 30 за приемлемое время.

Следует отметить, что основные временные затраты при работе алгоритма приходятся на перебор большого количества вариаций исходного вектора профиля предпочтений, каждая из которых подвергается проверке на манипулируемость. Представляется целесообразными перейти от распараллеливания на уровне данных к распараллеливанию на уровне алгоритма (параллелизм инструкций). Альтернативным, и более эффективным подходом, представляется использование распараллеливание на GPU с использованием технологии CUDA [17].

3.2 Расчет оценки коалиционной манипулируемости обобщенных скоринговых правил

При решении данной задачи расчет оценки манипулируемости для обобщенных скоринговых правил подсчета означает вычисление НК-индекса не 1, 2 или нескольких, а 6089 вариантов скоринговых процедур, возникающих в результате вариации параметров. Для каждой такой процедуры необходимо рассчитать результат для каждого из профилей предпочтений, а также для возможных вариаций каждого профиля каждым из участников с целью получения наилучшего для себя результата агрегации. При этом требования к ресурсам возрастают с ростом числа участников, поскольку помимо увеличения длины профиля предпочтений увеличивается и

число возможных коалиций. Все это приводит к тому, что задача оценки коалиционной манипулируемости обобщенных правил подсчета очков требовательна как к вычислительным ресурсам компьютера, так и к ресурсам памяти.

Поскольку для вероятностной модели Impartial Culture общее количество профилей составляет $(m!)^n$, что для случая 3 альтернатив и 40 участников дало бы порядка $13 \cdot 10^{30}$ профилей, что сделало бы невозможным расчеты на современных компьютерах, мы использовали метод генерации всех возможных профилей для вероятностной модели Impartial Anonymous Culture с дальнейшим пересчетом вероятностей профилей и результатов в Impartial Culture (были использованы алгоритмы и формулы из [18]). Общее количество профилей в Impartial Anonymous Culture оценивается как $C_{m!+n-1}^n$, для случая $m=3$ альтернатив и $n=40$ участников это дает $C_{m!+n-1}^n = 1221759$ профилей и занимает 193 мегабайта на жестком диске, тогда как в случае $n = 100$ полное количество профилей $= 96,5 \cdot 10^6$ и занимает 15,5 гигабайт.

Расчет для одного множества обобщенных скоринговых правил (включает 6089 различных правил) даже для 20 участников на персональном компьютере в однопоточной реализации алгоритма занял около 30 часов, что, учитывая очень быстрый рост алгоритмической сложности сделал невозможным расчет за разумное время случая, например, 40 участников. Все вышесказанное указывает на необходимость создания параллельной реализации алгоритма и ее запуска на суперкомпьютере.

Программа использует параллелизм данных в том смысле, что исходный массив профилей предпочтений распределяется между несколькими потоками программы, каждый из которых выполняется на отдельном вычислительном ядре в пределах одного узла кластера, что также соответствует модели параллелизма с общей памятью.

Однако, как и в случае необобщенных скоринговых правил, простое разделение исходного числа профилей поровну между потоками не подходит для задачи оценки манипулируемости, поскольку расчет завершается при нахождении первой вариации профиля, приводящей к выигрышу для данного агента и выбранного набора параметров. Это может привести к простоям отдельных вычислительных ядер, поскольку они могут завершить свою работу раньше остальных, а специфика выполнения задач на кластере не позволяет повторно использовать такие ядра до полного завершения расчета. Для решения этой проблемы необходимо организовать обработку массива данных с использованием “пула объектов” потоков.

Для решения этой задачи на языке C# мы использовали метод `Parallel.ForEach` с опцией разбиения «Без буферизации», реализованный в библиотеке `.Net Task Parallel Library (TPL)` [19]. Отключение буферизации при разбиении массива обрабатываемых данных (списка векторов – профилей предпочтений) необходимо для устранения возможной утечки памяти, связанной с тем, что механизм разбиения, используемый по умолчанию для структуры данных списка, может при некоторых условиях неограниченно выделять память.

Еще одна важная проблема, которую было необходимо решить при распараллеливании алгоритма, была минимизация необходимости синхронизации потоков. В результате приведенного преобразования нам удалось достичь 88% загрузки вычислительных ядер в процессе вычислений.

Программа использует параллелизм задач в том смысле, что применяется пакетный режим суперкомпьютерного кластера. Для ее работы мы подготовили файл скрипта для команды **sbatch**, в котором запрашиваются различные ресурсы, подготавливается среда, а затем осуществляется запуск расчетной программы для диапазона значений числа участников.

5. Заключение

В результате данной работы были разработаны параллельные версии расчетных программ для оценки манипулируемости скоринговых правил. Программа для расчета степени манипулируемости для 10 скоринговых правил перенесена на язык программирования C++, оптимизирована и распараллелена с использованием возможностей библиотеки STL и Intel oneAPI Toolkit, что сделало возможными расчеты для случая более 5 альтернатив.

В рамках данной работы был впервые построен сравнительный график зависимости NK-индекса от числа участников для числа альтернатив более 5 (на примере Правила большинства и расширенного предпочтения Лексимин).

Произведены расчеты для отдельных скоринговых правил для числа альтернатив до 7 и числа участников до 100, а также для числа альтернатив до 9 для числа участников до 30. Произведены расчеты оценки коалиционной манипулируемости обобщенных скоринговых правил для числа участников до 50.

Результаты расчетов доступны в виде интерактивных графиков на сайте Manipulability of Social Choice Rules Project (в разработке, URL: <https://demo.moscowlab.info>).

Несмотря на достигнутые результаты, в ходе работы мы столкнулись с проблемой, состоящей в том, что с ростом числа участников и альтернатив вычислительная сложность задачи возрастает настолько, что подход с использованием параллелизма данных становится неэффективным. Например, расчет оценки коалиционной манипулируемости для 100 участников для одного семейства обобщенных скоринговых правил не смог завершиться в течение 7 дней (расчет выполнялся на 6 вычислительных ядрах) и был прекращен по таймауту. Аналогично расчет оценки индивидуальной манипулируемости для Правила относительного большинства для случая 10 альтернатив не смог завершиться за приемлемое время даже для малого числа участников.

Представляются перспективными доработка подхода с использованием параллелизма задач для распределения вычислений для заданного числа агентов между узлами (с использованием модели распределенной памяти), а также дальнейшая доработка алгоритма для использования параллелизма инструкций, в том числе с использованием технологии CUDA.

6. Благодарности

Исследование поддержано Международным центром анализа и выбора решений НИУ ВШЭ. Исследование выполнено с использованием суперкомпьютерного комплекса НИУ ВШЭ [12].

Литература

1. Gibbard A. Manipulation of voting schemes // *Econometrica*. 1973. Vol. 41. P. 587–601. DOI: 10.2307/1914083.
2. Satterthwaite M. Strategy-proofness and Arrow's conditions: existence and correspondence theorems for voting procedures and social welfare functions // *Journal of Economic Theory*. 1975. Vol. 10. P. 187–217. DOI: 10.1016/0022-0531(75)90050-2.
3. Chamberlin J.R. An investigation into the relative manipulability of four voting systems // *Behavioral Science*. 1985. Vol. 30, no. 4. P. 195–203. DOI: 10.1002/bs.3830300404.
4. Diss M., Tselikhovski B.: Manipulable outcomes within the class of scoring voting rules // *Math. Soc. Sci.* 2021. Vol. 111. P. 11–18. DOI: 10.1016/j.mathsocsci.2021.02.002.
5. Favardin P., Lepelley D. Some Further Results on the Manipulability of Social Choice Rules // *Soc Choice Welfare*. 2006. Vol. 26. P. 485–509. DOI: 10.1007/s00355-006-0106-2.
6. Aleskerov F., Kurbanov E. Degree of manipulability of social choice procedures // Alkan A. et al. (eds.) *Current Trends in Economics. Studies in Economic Theory*. Berlin Heidelberg, N.Y.: Springer, 1999. Vol. 8. P. 13–27.
7. Keller O., Hassidim A., Hazon N. New Approximations for Coalitional Manipulation in Scoring Rules // *Journal of Artificial Intelligence Research*. 2019. Vol. 64. P. 109–145. DOI: 10.1613/jair.1.11335.
8. Nitzan S. The vulnerability of point-voting schemes to preference variation and strategic manipulation // *Public Choice*. 1985. Vol. 47. P. 349–370. DOI: 10.1007/BF00127531.
9. Kelly J. Almost all social choice rules are highly manipulable, but few aren't // *Social Choice and Welfare*. 1993. Vol. 10. P. 161–175. DOI: 10.1007/BF00183344.

10. Иванов А.А. Эффективные вычислительные схемы расчета манипулируемости процедур агрегирования // Информационные технологии и вычислительные системы. 2020. № 2. С. 38–50.
11. Aleskerov F., Karabekyan D., Sanver R., Yakuba V. On manipulability of positional voting rules // *SERIEs: Journal of the Spanish Economic Association*. 2011. Vol. 2, no. 4. P. 431–446. DOI: 10.1007/s13209-011-0050-y.
12. Kostenetskiy P.S., Chulkevich R.A., Kozyrev V.I. HPC Resources of the Higher School of Economics // *Journal of Physics: Conference Series*. 2021. Vol. 1740, no. 1. P. 012050. DOI: 10.1088/1742-6596/1740/1/012050.
13. Microsoft .Net Core Framework. URL: <https://learn.microsoft.com/ru-ru/dotnet/fundamentals/> (дата обращения: 29.01.2025)
14. Freeman A. The Object Pool Pattern. 2015. P. 137–156. DOI: 10.1007/978-1-4842-0394-1_7.
15. Meyers S. Effective STL: 50 specific ways to improve your use of the standard template library. GBR: Addison-Wesley Longman Ltd., 2001.
16. Barbera S, Bossert W, Pattanaik P. Ranking Sets of Objects // S. Barbera, P.J. Hammond, C. Seidl (eds.): *Handbook of Utility Theory*, vol. 2. Boston: Kluwer Academic Publishers, 2004.
17. Nickolls J., Buck I., Garland M., Skadron K. Scalable Parallel Programming with CUDA: Is CUDA the parallel programming model that application developers have been waiting for? // *Queue*. 2008. Vol. 6, no. 2. P. 40–53. DOI: 10.1145/1365490.1365500.
18. Иванов А.А. Алгоритмы расчета точных значений индексов манипулируемости для случая трех альтернатив // Журнал Новой экономической ассоциации. 2022. Т. 5, № 57. С. 14–23. DOI: 10.31737/2221-2264-2022-57-5-1.
19. Task Parallel Library. URL: <https://learn.microsoft.com/ru-ru/dotnet/standard/parallel-programming/task-parallel-library-tpl> (дата обращения: 29.01.2025)

Эволюционная разработка многопоточных программ с применением процедурно-параметрической парадигмы программирования

П.В. Косов¹, А.И. Легалов¹, В.С. Васильев²

¹НИУ «Высшая школа экономики»,

²Сибирский федеральный университет

При создании параллельных программ, используемых во встроенных системах и в Интернете вещей зачастую применяются методы гибкой разработки. Одним из подходов к повышению гибкости является расширение функциональности за счет добавления новых программных объектов без изменения ранее написанного кода. Использование процедурно-параметрической парадигмы программирования позволяет во многих ситуациях безболезненно и независимо добавлять как данные, так и функции, предоставляя дополнительные возможности и технические приемы для разработки параллельных программ. В работе рассматривается использование данной парадигмы для многопоточного программирования на процедурно-параметрическом расширении языка программирования C.

Ключевые слова: параллельное программирование, многопоточное программирование, процедурно-параметрическая парадигма программирования, полиморфизм, гибкая разработка программного обеспечения.

1. Введение

При разработке программного обеспечения учитываются различные критерии качества. Не все они непосредственно связаны с параллелизмом, что обуславливается различными требованиями, предъявляемыми к конечному продукту. Вместе с тем, при создании параллельных программ обычно используются дополнительные библиотеки, которые прямо или косвенно могут влиять на достижение требуемых критериев. Это необходимо учитывать при применении различных подходов и парадигм программирования.

Расширение программ без изменения ранее написанного кода или с его минимальными изменениями является одним из критериев, определяющих гибкую разработку. Его важность обуславливается неполным знанием об окончательной функциональности программ во время их создания и начальной эксплуатации, а также необходимостью ускорить выход продуктов на рынок за счет инкрементального наращивания функциональности в каждом последующем релизе. В подобных ситуациях добавление новых конструкций, изменяющих уже написанный код, зачастую ведет к появлению неожиданных ошибок и непредсказуемому поведению. Ситуация может дополнительно усугубляться при использовании взаимодействующих параллельных процессов. Как следствие это ведет к увеличению стоимости разработки и сопровождения программной системы.

Одним из подходов, поддерживающих эволюционное расширение является динамическое связывание альтернативных программных объектов, обеспечивающее гибкое изменение данных во время выполнения. Он определяет специфику динамического полиморфизма, который отличается от статического полиморфизма, ориентированного на выявление альтернатив на этапе компиляции. Динамический полиморфизм, широко используется в объектно-ориентированном программировании (ООП). Основным решением в объектно-ориентированных (ОО) языках со статической типизацией является совместное использование наследования и виртуализации. Наследование поддерживает идентичность интерфейсов родительского и дочернего классов, а виртуализация позволяет подменять реализации методов в потомках. Подобный подход используется во многих современных ОО языках

программирования. В качестве примера можно привести языки C++ [1], Java [2] и другие. Поддержка динамического полиморфизма на основе статической утиной типизации включена и в процедурные языки. В Go [3] это механизм интерфейсов, позволяющий использовать для обработки альтернатив функции, связанные со структурами данных. Близкий механизм на основе типажей реализован в языке программирования Rust [4]. Вместе с тем, все эти подходы напрямую не поддерживают эволюционного расширения программ в случае множественного полиморфизма (мультиметодов) [5].

2. Процедурно-параметрическая парадигма

Еще один подход к реализации динамического полиморфизма предложен в рамках процедурно-параметрической парадигмы программирования (4П) [6]. Она предлагает гибкое эволюционное расширение как данных, так и функций, используя методы формирования обобщений и их специализаций на основе параметрического механизма. Подход может быть интегрирован в уже существующие языки процедурного и функционального программирования, а также использоваться при разработке новых языков. В настоящее время процедурно-параметрический (ПП) механизм реализован как расширение языка программирования C [7]. Данный язык широко используется в предметных областях, где ОО подход не является эффективным. Поэтому его дальнейшее развитие в направлении, связанном с поддержкой гибкой разработки программного обеспечения может оказаться перспективным.

В настоящее время методы кодирования, применяемые при разработке программ на языке C, могут приводить к ошибкам за счет отсутствия в ряде случаев контроля за типами данных во время компиляции. Такие ситуации возникают из-за того, что язык поддерживает произвольные и неконтролируемые преобразования типов. Например, в библиотеке потоков POSIX threads [8] многие функции для передачи параметров и возврата результатов используют указатели на `void`. Это приводит к ненадежному и неконтролируемому преобразованию типов. Подобный стиль кодирования снижает безопасность программ, хотя зачастую и позволяет достичь высокой производительности. Другой проблемой языка программирования C является отсутствие жесткой зависимости между альтернативными данными и идентифицирующими их признаками, что наблюдается, например, при использовании объединений (`union`). Признаки альтернатив могут произвольно использоваться программистом и приводить к ошибкам, выявляемым только во время выполнения.

Многие из этих проблем эффективно решаются в ОО языках программирования, а также в Go и Rust за счет использования динамического полиморфизма. ПП парадигма, наряду с решением аналогичных проблем, позволяет еще и эффективно поддерживать множественный полиморфизм, обеспечивая эволюционное расширение мультиметодов. Моделирование конструкций, расширяющих язык C, подтвердило возможности их реализации [7]. На основе данных, полученных в ходе моделирования, процедурно-параметрические конструкции были интегрированы в семейство компиляторов `clang` [9]. Описание предлагаемых расширений, обновляемое по мере вводимых изменений, размещено в сети Интернет [10].

3. Сопоставление объектно-ориентированного и процедурно-параметрического подходов

Основное отличие ПП подхода по поддержке динамического полиморфизма заключается в ином техническом решении, обеспечивающем формирование отношений между альтернативными данными и функциями как на уровне языковых конструкций, так и при их реализации в семантической модели языка. Это отличие можно рассмотреть в сравнении с методами поддержки ОО полиморфизма. Общие схемы, демонстрирующие отличие ОО и ПП подходов, представлены на рис. 1.

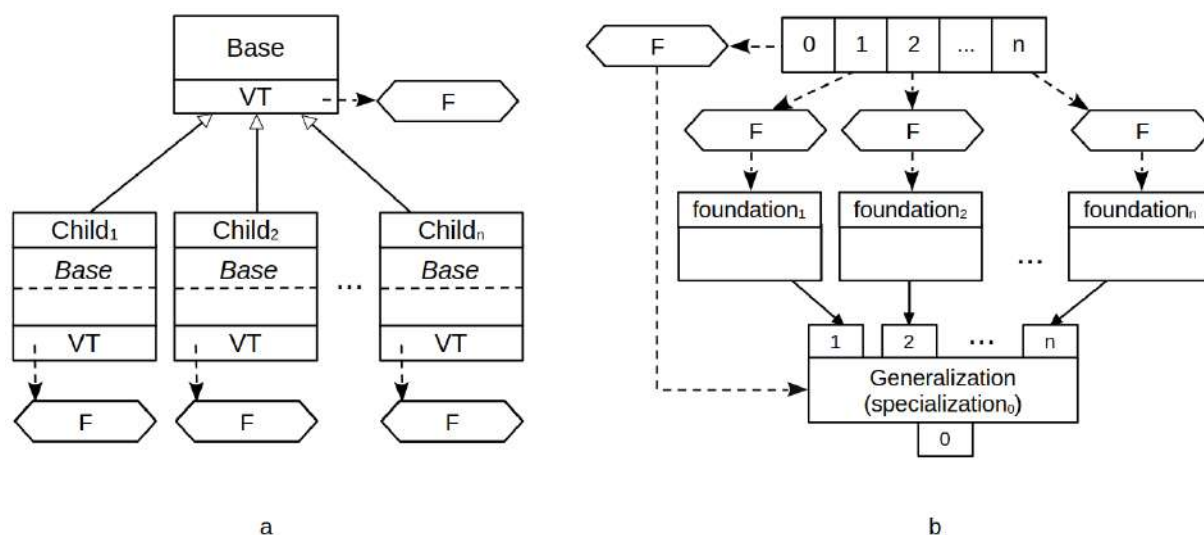


Рис. 1. Реализация объектно-ориентированного (а) и процедурно-параметрического (b) полиморфизма

Объектно-ориентированный полиморфизм (рис. 1а) формируется путем совместного использования наследования и виртуализации. Виртуализация при этом обычно реализуется за счет таблиц виртуальных методов (VT). Таблица базового класса *Base*, выступающего в роли обобщения, содержит указатели на один или несколько методов (*F*), которые обычно переопределяются в виртуальных таблицах производных классов, являющихся специализациями обобщения (*Child₁–Child_n*). При этом каждый производный класс, расширяя базовый класс, имеет собственный тип, что выражается в уникальном имени класса (*Child_i*). Производные классы формируются независимо друг от друга. Это позволяет эволюционно расширять альтернативные программные объекты, каждый из которых, при наличии одинаковых интерфейсов, может иметь иную функциональность, обеспечивая тем самым реализацию динамического ОО полиморфизма. Однако добавление нового виртуального метода требует модификации всей иерархии классов. Также данный механизм напрямую не поддерживает множественного полиморфизма (мультиметодов), для реализации которых ОО решением является использование диспетчеризации, не способствующей безболезненному расширению классов. Зачастую для гибкой реализации мультиметодов в ОО языках используют дополнительный код, который не связан с вычислениями, определяемыми решаемой задачей, и предназначен только для организации дополнительных взаимодействий [11, 12].

В случае ПП подхода реализация отношений между обобщением (*Generalization*) и его специализациями осуществляется иным способом (рис. 1b). Процедурно-параметрический полиморфизм обеспечивает формирование альтернатив (*specialization₁–specialization_n*), используя для этого основы специализаций, которые являются независимыми абстрактными типами данных (*foundation₁–foundation_n*). Добавление альтернативных специализаций к обобщающему их типу данных может осуществляться в произвольные моменты времени и в различных единицах компиляции. Формируемые при этом композиции являются специализациями обобщения и принадлежат к тому же типу, что и обобщающий тип (*Generalization*). Основа специализации трактуется как подтип обобщения. Само обобщение также является одной из специализаций, не содержащей основу (*specialization₀*), но имеющей свои внутренние данные. Механизм идентификации специализаций локализован внутри каждого обобщения с использованием внутренней индексации (параметрических индексов).

Обработка сформированных альтернатив осуществляется с использованием внешних

независимых специальных функций — обработчиков специализаций. Они расширяют обобщающую их функцию. Обработчики специализаций, как и сами специализации, могут добавляться независимо друг от друга. Механизм идентификации специализаций используется для автоматического выбора обработчиков через параметрические таблицы, каждая из которых связана со своим набором обобщающих функций. Добавление новых обобщающих функций и их обработчиков может осуществляться в любой момент времени независимо от уже существующих данных и функций, что характерно для добавления любых новых функций при процедурном подходе. Эти функции могут содержать произвольное число обобщающих аргументов, обеспечивая тем самым безболезненное расширение мультиметодов [7].

В предлагаемом расширении языка С основами специализаций могут быть разнообразные типы данных 2. Их можно сформировать с использованием базовых типов данных, а также на основе именованных и неименованных структур. Помимо этого специализации могут формироваться на основе указателей на различные типы данных. Для идентификации подключаемых основ специализаций используются признаки, задаваемые идентификаторами. Они являются уникальными внутри каждого из формируемых обобщений. Также в качестве признака может выступать имя типа используемой основы специализации, что может трактоваться как применение в качестве признака имени типа. Подробное описание реализованных ПП расширений языка программирования С представлено в [10].

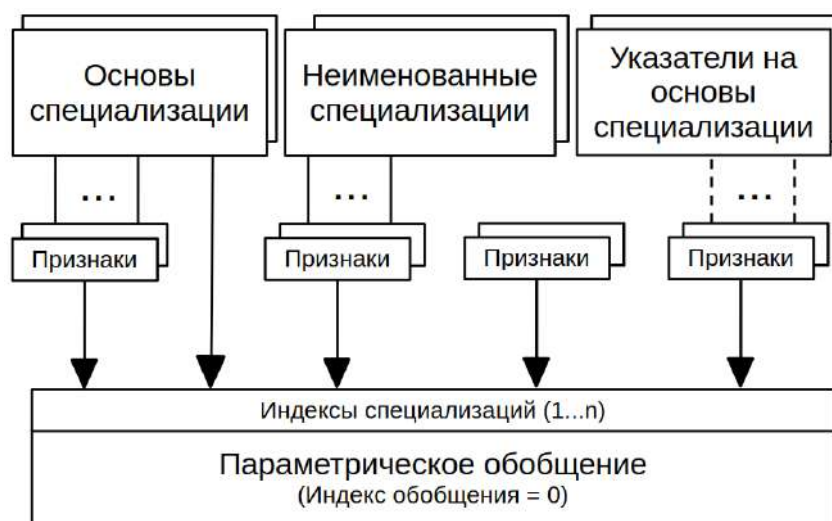


Рис. 2. Варианты процедурно-параметрических специализаций в расширении языка С

Примеры, демонстрирующие различные варианты прямого эволюционного расширения как для ОО, так и для ПП подходов приведены в каталоге *evolution* репозитория на Гитхаб [17].

4. Формирование оберток вокруг потоковых функций

Несмотря на появление новых перспективных языков, применение языка С в параллельном программировании остается популярным, что связано с его эффективным отображением на современные архитектуры. Использование языка актуально для встроенных систем, а также для приложений с критическим временем реакции на внешние события. Предлагаемые процедурно-параметрические расширения, на наш взгляд, позволяют повысить надежность и гибкость при создании многопоточных программ за счет создания соответствующих оберток над ненадежными конструкциями, а также за счет минимизации изменений в уже написанном коде.

Использование указателей на пустой тип `void` ведет к разыменованию типов и потере контроля над ними со стороны компилятора. Вместе с тем этот прием широко распространен при программировании на языке C и используется в различных библиотеках. Например, библиотека POSIX Threads [8] использует такой подход в функциях создания потока `pthread_create` и ожидания его завершения `pthread_join`:

```
int pthread_create( pthread_t *thread,
                  const pthread_attr_t *attr,
                  void *(*start_routine) (void *),
                  void *arg
);

int pthread_join( pthread_t thread, void **retval );
```

В связи с этим при вызове таких функций часто приходится осуществлять ненадежные преобразования типов данных.

4.1. Формирование объектно-ориентированной обертки

Преодоление этих недостатков в случае ОО программирования осуществляется за счет формирования абстрактной обертки, которая позволяет избавиться от прямого применения ненадежных конструкций. Пример простой обертки, реализованной в виде соответствующего класса [13], представлен ниже:

```
class Thread {
public:
    virtual ~Thread () {}
    virtual void run () = 0;
    int start () {
        return pthread_create(
            &_ThreadId, nullptr,
            Thread::thread_func, this );
    }
    int wait () {
        return pthread_join( _ThreadId, NULL );
    }
protected:
    pthread_t _ThreadId;
    static void* thread_func(void* d) {
        (static_cast <Thread*>(d))->run();
        return nullptr;
    }
};
```

На основе этой обертки можно создавать потоки, которые решают разнообразные задачи. В качестве примера рассмотрим класс, обеспечивающий вычисление периметра прямоугольника. Он определяет функционал соответствующего потока за счет наследования от базового класса `Thread`:

```
class ThreadRect : public Thread {
public:
    ThreadRect (int xx, int yy): x{xx}, y{yy} {}
    virtual void run () {
        p = double(2*(x+y));
    }
    void print(const char* str) {
        std::cout << str << ": Perimeter=" << p << "\n";
    }
protected:
    int x;
    int y;
    double p;
};
```

Использование этого и других подобных классов позволяет формировать и объединять множество потоков, избегая прямого использования функций библиотеки POSIX Threads:

```
...
ThreadRect thread1{3, 5};
ThreadRect thread2{13, 15};
thread1.start(); thread2.start();
thread1.wait(); thread2.wait();
thread1.print("Thread_1");
thread2.print("Thread_2");
...
```

Подобным образом в языке C++ реализована и более сложная обертка, определяющая стандартную библиотеку `thread` [14].

4.2. Формирование процедурно-параметрической обертки

Аналогичную обертку, подменяющую указатели на `void` на обобщения, обеспечивающие контроль над типами данных за счет специализаций, можно сформировать и при использовании ПП программирования. Одним из эквивалентных вариантов является создание обобщенной структуры, в которой специализации могут использоваться для передачи аргументов в формируемый поток и возврата из него результата.

```
typedef struct ThreadData {pthread_t threadId;}<> ThreadData;
```

Для работы с этой структурой можно создать абстрактную обобщенную функцию `RunThread`, не имеющую тела. На ее основе будут создаваться обработчики специализаций, формирующие требуемые потоковые функции. Данная функция в качестве обобщенного аргумента получает структуру с входными параметрами, передаваемые в поток, а также возвращаемыми из него:

```
void RunThread <ThreadData* d>() = 0;
```

Промежуточная потоковая функция `ThreadFunc`, скрытая от программиста внутри обертки, будет запускать различные обработчики специализаций этой обобщенной функции, формируемые программистом:

```
void* ThreadFunc(void* d) {
    RunThread<(Thread*)d>();
    return NULL;
}
```

Данная функция используется в качестве аргумента функции запуска потока, которая размещается внутри обертки `StartThread`, получающей данные, используемые для вычислений и возврата полученного результата:

```
int StartThread(ThreadData* td) {
    return pthread_create(td->threadId, NULL, ThreadFunc, td);
}
```

Слияние, обеспечивающее ожидание и завершение запущенных потоков, как и другие аналогичные функции библиотеки, тоже можно реализовать в соответствующей обертке:

```
int WaitThread(ThreadData* td) {
    return pthread_join(td->threadId, NULL);
}
```

Подобную обертку, используя процедурно-параметрическую парадигму, можно создать и над библиотекой `threads.h`, появившейся в стандарте языка C11 [15].

Рассмотренные оберточные функции можно применять для формирования различных потоков, рассматриваемых в качестве специализаций обобщения `RunThread`. В частности, использование прямоугольника и определение данных, необходимых для вычисления его периметра, будут выглядеть следующим образом:

```
typedef struct Rectangle{
    int x, y; // Sides of a rectangle
} Rectangle;

typedef struct RectPerimeter {
    Rectangle r;    // Rectangle
    double p;       // Perimeter as result
} RectPerimeter;

// Formation of specialization for a generalized thread
ThreadData + <RectPerimeter>;
```

Обработчик специализации, вычисляющий периметр, может быть реализован следующим образом:

```
void RunThread<ThreadData.RectPerimeter *rp>() {
    rp->p = (double)((rp->r.x+rp->r.y)*2);
}
```

Он получает в качестве параметра сформированную специализированную структуру, возвращая в поле периметра полученный результат. Функция, осуществляющая непосредственный вывод значения периметра, выделяет из нее данные, размещенные в структуре `RectPerimeter`:

```
void PrintRectPerimeter(RectPerimeter* rp, const char* str) {
    printf("Perimeter of %s = %f\n", str, rp->p);
}
```

Главная функция, запускающая несколько потоков, скрывает особенности библиотеки POSIX Thread аналогично тому, как это делается и в ОО программе:

```
...
struct ThreadData.RectPerimeter thread1;
thread1.threadId = 0;
thread1.@r.x = 3;
thread1.@r.y = 5;
struct ThreadData.RectPerimeter thread2;
thread2.threadId = 1;
thread2.@r.x = 7;
thread2.@r.y = 4;

StartThread((ThreadData*)&thread1);
StartThread((ThreadData*)&thread2);

WaitThread((ThreadData*)&thread1);
WaitThread((ThreadData*)&thread2);

PrintRectPerimeter(&(thread1.@), "Thread_1");
PrintRectPerimeter(&(thread2.@), "Thread_2");
...
```

После формирования данных, передаваемых в потоки, производится их запуск в функциях (`StartThread`). Ожидание завершения потоков происходит на функциях `WaitThread`, а затем осуществляется вывод периметров. Символ «@» отделяет данные специализации от основной структуры, описывающей поток.

5. Эволюционное расширение многопоточных ПП программ

Использование ПП парадигмы позволяет добавлять в многопоточную программу, как новые альтернативные данные, так и новые функции, осуществляющие их обработку. В этом плане использование методов эволюционного расширения практически ничем не отличаются от их применения в последовательных ПП программах. Однако в случае многопоточности допускается параллельная обработка с сохранением гибкости по расширению ранее написанного кода.

5.1. Добавление новых альтернативных данных

Рассмотрим особенности расширения программ на следующем простом примере. Пусть имеется массив хранящий различные геометрические фигуры, которые изначально могут быть прямоугольниками и треугольниками. Необходимо вычислить периметр каждой из фигур после чего подсчитать общую сумму. Для вычисления периметра каждой фигуры предполагается использование отдельного потока.

Прямое применение традиционного процедурного подхода в языке C обычно ведет к представлению фигур либо с использованием объединений, либо применяется разыменовывание типов за счет использования указателей на пустой тип (`void*`). В первом случае ключ, определяющий тип фигуры инструментально не связан с хранимой фигурой. Во втором варианте приходится использовать разыменовывание типов. В обоих случаях имеются проблемы, связанные с обеспечением надежности программы. В случае объединений описание обобщенной фигуры, задающей прямоугольник и треугольник, можно представить следующим образом:

```
typedef struct Rectangle {
    int x, y;
} Rectangle;

typedef struct Triangle {
    int a, b, c;
} Triangle;

typedef enum key {RECTANGLE, TRIANGLE} key;
typedef struct Figure {
    key k;
    union {
        Rectangle r;
        Triangle t;
    };
} Figure;
```

Вычисление периметра в этом случае обычно реализуется с использованием единой централизованной функции, в которой по ключу осуществляется выбор хранимой фигуры:

```
double FigurePerimeter(Figure *s) {
    switch(s->k) {
        case RECTANGLE:
            return RectanglePerimeter(&(s->r));
            break;
        case TRIANGLE:
            return TrianglePerimeter(&(s->t));
            break;
        default:
            return 0.0;
    }
}
```

Если в разрабатываемую процедурную программу потребуется добавить еще одну или несколько разновидностей фигур, например круг, то ранее написанный код придется изменить. Это затрудняет эволюционную разработку и последующее сопровождение. Проблемы, связанные с изменением обработчиков, особенно проявляются при большом числе используемых в них альтернативных данных. Следует также отметить, что подобное решение в случае ООП позволяет без труда добавлять новые альтернативы за счет производных классов и обеспечивать для них формирование соответствующих методов, используя ОО полиморфизм. Но при этом добавление новых методов ведет к изменению как базового, так и производных классов.

Процедурно-параметрическое расширение языка C аналогично по возможностям ОО подходу. Оно обеспечивает гибкое добавление как новых данных, так и новых обработчиков специализаций, так как альтернативы, являющиеся специализациями уже существующего общего типа формируются независимо друг от друга. Это же касается и обработчи-

ков специализаций, каждый из которых независимо расширяет обобщенную функцию. При использовании ранее приведенных описаний прямоугольника и треугольника обобщенная фигура будет формироваться на основе структуры, в которой отсутствуют специализации:

```
typedef struct Figure {} <> Figure;
```

Далее независимо друг от друга формируются специализации этого обобщения, определяющие фигуру–прямоугольник и фигуру–треугольник:

```
Figure + < rect: Rectangle; >;  
Figure + < trian: Triangle; >;
```

Каждая из специализаций может быть описана в своем заголовочном файле, что при добавлении следующих фигур, обеспечивает их независимость и неизменность ранее написанного кода. Например, добавление круга будет связано только с его описанием и зависимостью от основы и обобщенной фигуры:

```
typedef struct Circle {  
    int r;  
} Circle;  
  
Figure + < circ: Circle; >;
```

Аналогичным образом в независимых единицах компиляции осуществляется формирование обобщенной функции вычисления периметра и ее различных обработчиков, что позволяет постепенно и эволюционно наращивать функциональность программы:

```
double FigurePerimeter<Figure *f>() = 0;  
  
void FigurePerimeter<Figure.rect *f>() {  
    RectanglePerimeter(&(f->@));  
}  
  
void FigurePerimeter<Figure.trian *f>() {  
    TrianglePerimeter(&(f->@));  
}
```

Для вычисления периметра в многопоточном режиме необходимо сформировать структуру, позволяющую передавать в потоковую функцию аргументы и возвращать результат. Эта структура используется в качестве специализации потоковой функции, вычисляющей периметр обобщенной фигуры:

```
typedef struct FigurePerimeterData {  
    int iThread;    // Thread id  
    Figure *f;      // Pointer to figure  
    double p;       // Figure perimeter as result  
} FigurePerimeterData;  
  
struct ThreadData + <FigurePerimeterData;>;  
  
void RunThread<ThreadData.FigurePerimeterData *figurePerimeter>() {  
    figurePerimeter->@p = FigurePerimeter<figurePerimeter->@f>();  
}
```

Для хранения различных геометрических фигур можно создать простейший контейнер, например, на основе одномерного массива:

```
enum {max_len = 100};  
  
typedef struct Container {  
    int len;  
    struct Figure *cont[max_len];  
} Container;
```

Обработка всего массива может быть организована запуском потоков для каждого из его элементов:

```
double CalcFigurePerimetersInContainer(Container *c) {
    struct ThreadData.FigurePerimeterData perimeterArray[c->len];
    struct ThreadData.FigurePerimeterData *pElement;
    for(int i = 0; i < c->len; ++i) {
        // Specialization of the memory area
        // allocated for the thread data
        init_spec(ThreadData.FigurePerimeterData, (perimeterArray+i));
        pElement = (perimeterArray+i);
        pElement->iThread = i;
        pElement->@f = c->cont[i];
        pElement->@p = 0.0;
        StartThread((ThreadData*)(perimeterArray+i));
    }

    double perimeterSum = 0.0;
    for(int i = 0; i < c->len; ++i) {
        pElement = (perimeterArray+i);
        WaitThread((ThreadData*)(perimeterArray+i));
        perimeterSum += pElement->@p;
        printf("The thread %d: perimeter = %lf\n",
            pElement->iThread, pElement->@p);
    }
    printf("Common Figures Thread Perimeter = %lf\n", perimeterSum);
    return perimeterSum;
}
```

В представленном примере вспомогательный массив переменных, передаваемых в потоковую функцию, формируется на стеке, что позволяет избежать динамического выделения памяти с использованием специальной функции `create_spec`. Однако каждая из формируемых специализаций, размещаемая в массиве `perimeterArray`, имеет свой признак, который не устанавливается автоматически. Поэтому для формирования признаков специализаций используется специальная функция инициализации `init_spec`, аргументами которой являются тип специализации и адрес области памяти, выделяемой под инициализируемую специализацию. Описание дополнительных функций, используемых для поддержки процедурно-параметрического программирования в расширении языка C, приведено в [?].

5.2. Добавление новых функций

При добавлении в ПП программу новой функциональности, например, для вычисления площади геометрических фигур в уже сформированный код также не нужно вносить дополнительных изменений создаются только новые функции, обеспечивающие вычисление площади по аналогии с вычислением периметра. Для формирования новых потоков создается обработчик специализации, содержащий данные, необходимые для вычисления площади. В этой ситуации процедурно-параметрический подход является более гибким по сравнению с объектно-ориентированным, при котором добавление нового метода ведет к изменению интерфейсов ранее написанных классов, а также включением в них новых реализаций, обеспечивающих вычисление площадей фигур уже описанных в программе.

6. Многопоточность и множественный полиморфизм

Аналогичным образом ПП подход позволяет расширять функции от нескольких полиморфных аргументов, а также добавлять новые мультиметоды в программу, формировать новые специализации, которые осуществляют обработку этих мультиметодов [7]. Использование многопоточного программирования в этом случае также не мешает гибкому расширению функциональности программ.

В качестве примера параллельного выполнения мультиметодов можно рассмотреть добавление функции осуществляющей вывод отношений между различными парами геометрических фигур, размещенных в контейнере. Для запуска каждого потока используется функция `TreadContainerMultimethodOut`. Она во вложенном цикле формирует данные для каждой из пар массива фигур. После чего запускает стартовую функцию потока `StartThread`.


```
// Output of pairs of figures from container
// to the specified thread using a multimethod
void TreadContainerMultimethodOut(Container *c, FILE* ofst) {
    struct ThreadData.MultimethodData mmArray[c->len][c->len];
    struct ThreadData.MultimethodData *pElement;
    int threadId = 0;
    for(int i = 0; i < c->len; i++) {
        for(int j = 0; j < c->len; j++) {
            init_spec(ThreadData.MultimethodData, &mmArray[i][j]);
            pElement = &mmArray[i][j];
            pElement->iThread = threadId;
            pElement->c = c;
            pElement->figureIndex1 = i;
            pElement->figureIndex2 = j;
            pElement->ofst = ofst;
            pElement->p = 0.0;
            StartThread((ThreadData*)&mmArray[i][j]);
            threadId++;
        }
    }
    // Assembling the results returned by threads
    for(int i = 0; i < c->len; i++) {
        for(int j = 0; j < c->len; j++) {
            pElement = (&mmArray[i][j]);
            WaitThread((ThreadData*)&mmArray[i][j]);
        }
    }
}
```

Основа специализации используется для передачи в потоки информации об индексах каждой из фигур формируемой пары:

```
// The structure of data transmitted to a thread processing
// a one of combination of multimethod
typedef struct MultimethodData {
    int iThread; // Thread ID
    Container* c;
    // The figure container
    int figureIndex1; // The index of the first figure
    int figureIndex2; // The index of the second figure
    FILE* ofst;
} MultimethodData;
```

Специализация, сформированная с использованием этой основы, обеспечивает использование обобщенного обработчика, запускающего поток:

```
// Specialization for the thread used in the multimethod
ThreadData + <MultimethodData>;
```

Переопределяемая поточная функция RunThread, использует сформированную специализацию и вызывает функцию MultimethodLauncher, осуществляющую непосредственный вызов одного из обработчиков мультиметодов, доступный через обобщающую функцию Multimethod.

```
// Implementation of a specialization handler that runs in a thread.
void RunThread<ThreadData.MultimethodData *mm>() {
    MultimethodLauncher(mm->iThread,
                        mm->c, mm->figureIndex1,
                        mm->figureIndex2, mm->ofst);
}

// A function that starts the handler for figures combination
void MultimethodLauncher(int iThread,
    Container *c,
    int i, int j, FILE* ofst) {
    sleep(rand()%3); // The delay for generating a random execution time
    printf("%d_[%d,%d]:_", iThread, i, j);
    Multimethod<c->cont[i], c->cont[j]>(ofst);
}
```

Сам мультиметод, формируется на основе абстрактной обобщенной функции и множества отдельных обработчиков специализаций, определяющих все необходимые отношения между геометрическими фигурами, реализованными в программе.

```
// A generalizing function defining an abstract multimethod
void Multimethod<Figure* f1, Figure* f2>(FILE* ofst) = 0;

// Specialization handler for two rectangles
void Multimethod<Figure.rect* r1, Figure.rect* r2>(FILE* ofst) {
    fprintf(ofst, "Two Rectangles Combination\n");
}

// Specialization handler for rectangle and triangle
void Multimethod<Figure.rect* r1, Figure.trian* t2>(FILE* ofst) {
    fprintf(ofst, "Rectangle - Triangle Combination\n");
}

// Specialization handler for triangle and rectangle
void Multimethod<Figure.trian* t1, Figure.rect* r2>(FILE* ofst) {
    fprintf(ofst, "Triangle - Rectangle Combination\n");
}

// Specialization handler for two triangles
void Multimethod<Figure.trian* t1, Figure.trian* t2>(FILE* ofst) {
    fprintf(ofst, "Two Triangles Combination\n");
}
```

Добавление новой фигуры и ее обработчиков специализаций, обеспечивающих взаимодействие с другими фигурами осуществляется без изменения ранее написанного кода. Аналогичным образом возможно и добавление новых мультиметодов, реализующих другие алгоритмы взаимодействия между фигурами.

7. Заключение

Предлагаемое расширение языка программирования C конструкциями, обеспечивающими инструментальную поддержку процедурно-параметрической парадигмы программирования, позволяет разрабатывать гибкие и эволюционно расширяемые параллельные программы. При этом обеспечивается добавление как новых альтернативных данных, так и функций без изменения ранее написанного кода. Эволюционное расширение возможно и в случае множественного полиморфизма. Реализация проекта осуществляется путем интеграции в компилятор clang [9], что позволяет изменять как синтаксический анализатор, так и абстрактное синтаксическое дерево, адаптируя их под новые конструкции. Проект реализуется под открытой лицензией и размещен на Гитхаб [16]. Примеры, демонстрирующие возможности процедурно-параметрической парадигмы программирования, включающие и использование многопоточности, также размещены на гитхаб [17].

Литература

1. Gregoire M. Professional C++. John Wiley & Sons, 2018. 1122 p.
2. Sciore E. Java Program Design. Apress Media, 2019. 1122 p.
3. Freeman A. Pro Go: The Complete Guide to Programming Reliable and Efficient Software Using Golang. Apress, 2022. 1105 p.
4. Blandy J., Orendorff J., Tindall L.F. Programming Rust. O'Reilly Media, 2021. 735 p.
5. Эдджер Дж. C++: библиотека программиста. СПб.: ЗАО «Издательство Питер», 1999. 320 с.

6. Легалов А.И. Процедурно-параметрическая парадигма программирования. Возможна ли альтернатива объектно-ориентированному стилю? Красноярск: 2000. Деп. рук. № 622-B00 Деп. в ВИНТИ 13.03.2000. 43 с.
URL: <http://www.softcraft.ru/ppp/pppfirst/>. (дата обращения: 31.01.2025).
7. Легалов А.И., Косов П.В. Расширение языка C для поддержки процедурно-параметрического полиморфизма // Моделирование и анализ информационных систем. 2023. Т. 30, № 1. С. 40–62.
DOI: 10.18255/1818-1015-2023-1-40-62.
8. Butenhof D.R. Programming with POSIX Threads. Addison-Wesley, 1997. 398 p.
9. Clang: a C language family frontend for LLVM. URL: <https://clang.llvm.org/> (дата обращения: 31.01.2025).
10. Легалов А.И., Косов П.В. Процедурно-параметрическое расширение языка программирования C. Синтаксис и семантика. URL: <http://softcraft.ru/ppp/ppc/> (дата обращения: 31.01.2025).
11. Мейерс С. Наиболее эффективное использование C++. 35 новых рекомендаций по улучшению ваших программ и проектов. М.: ДМК Пресс, 2000. 304 с.
12. Легалов А.И. ООП, мультиметоды и пирамидальная эволюция // Открытые системы. 2002. № 3. С. 41–45.
13. POSIX Threads. URL: https://ru.wikipedia.org/wiki/POSIX_Threads (дата обращения: 31.01.2025).
14. Гримм Р. Параллельное программирование на современном языке C++. М.: ДМК Пресс, 2022. 616 с.
15. C. Concurrency support library. URL: <https://en.cppreference.com/w/c/thread> (дата обращения: 31.01.2025).
16. Размещение проекта с процедурно-параметрической версией языка программирования C на Гитхаб. URL: <https://github.com/kpdev/llvm-project/tree/pp-extension-v2> (дата обращения: 31.01.2025).
17. Примеры на Гитхаб, написанные с использованием процедурно-параметрической версии языка C: URL: <https://github.com/kreofil/evo-situations> (дата обращения: 31.01.2025).

Численный анализ неопределенности алгоритмов идентификации графовых моделей*

И.Д. Костылев, В.А. Калягин

Лаборатория алгоритмов и технологий анализа сетевых структур,
Национальный исследовательский университет
Высшая школа экономики в Нижнем Новгороде

Рассматривается известная в интеллектуальном анализе данных задача идентификации графа концентраций по наблюдениям в общей постановке. Исследуется неопределенность алгоритмов идентификации графа концентраций. Основное внимание уделяется анализу влияния плотности графа концентраций и выбора меры зависимости на неопределенность различных алгоритмов идентификации. В качестве алгоритмов идентификации используются алгоритмы множественной проверки гипотез. Реализован генератор случайных графовых моделей. Проведены массивные вычислительные эксперименты по оценке неопределенности идентификации графа концентраций в различных сетях случайных величин в зависимости от плотности графа. Сделаны выводы.

Ключевые слова: графовые вероятностные модели, Марковские случайные поля, граф концентраций, плотность графа, алгоритмы идентификации графа концентраций, неопределенность алгоритмов, множественная проверка гипотез.

1. Введение

Графовые модели стали популярным инструментом интеллектуального анализа данных в последние десятилетия. В работе рассматривается один из аспектов анализа вероятностных графовых моделей (Probabilistic Graphical Models or Random Markov Fields): идентификация графа концентраций по наблюдениям. Задача идентификации графа концентраций (Graphical Model Selection Problem) хорошо известна и активно исследуется в литературе (см. обзоры [1, 2]). Вместе с тем, в литературе пока еще недостаточно изучено влияние параметров графовой модели на качество алгоритмов идентификации графа концентраций. Ввиду высокой сложности теоретического анализа такого влияния, наиболее перспективным представляется подход, основанный на специально построенных вычислительных экспериментах. Для проведения вычислительных экспериментов требуется генератор случайных неотрицательно определенных матриц с заданной плотностью ненулевых элементов. В данной работе мы используем общую постановку задачи идентификации графа концентраций в сети случайных величин и методологию анализа влияния параметров графовой модели на качество алгоритмов идентификации, предложенную в нашей работе [3]. Основное отличие данной работы в использовании другого генератора случайных графовых моделей. В работе [3] использовался генератор случайных графовых моделей с заданной плотностью графа концентраций, основанный на разложении Холецкого. Целью настоящей работы является численный анализ неопределенности идентификации графовой модели в зависимости от плотности графа. В настоящей работе мы представляем результаты массивных вычислительных экспериментов по оценке влияния плотности графа на качество (неопределенность) алгоритмов идентификации графа концентраций с использованием генератора случайных графовых моделей, построенного на основе принципа доминирующей диагонали. Результаты проведенных экспериментов показывают близкое поведение неопределенности идентификации графа концентраций в зависимости от плотности графа для различных генераторов случайных графовых моделей. Это подтверждает адекватность предложенной методологии анализа и открывает новые перспективные задачи для исследо-

* Работа подготовлена в рамках Программы фундаментальных исследований Национального исследовательского университета «Высшая школа экономики» (НИУ ВШЭ).

вания. В частности, для обоих генераторов заметно резкое ухудшение качества алгоритмов идентификации с ростом плотности графа. Этот феномен требует дополнительного исследования и модификации алгоритмов для повышения их качества.

Работа построена следующим образом. В разделе 2, следуя [3], мы кратко напоминаем постановку задачи, методологию оценки неопределенности алгоритмов идентификации графа концентраций и описываем рассматриваемые алгоритмы идентификации. В разделе 3 приведено описание генератора случайных графовых моделей, построенного на основе принципа доминирующей диагонали. В разделе 4 дано краткое описание результатов экспериментов и приведена ссылка на ресурс с полным представлением результатов. В последнем разделе приведены выводы по представленным материалам и описаны дальнейшие направления исследований.

2. Задача идентификации графа концентраций

Сетью случайных величин мы называем пару (X, γ) , где $X = (X_1, X_2, \dots, X_N)$ случайный вектор размерности N , γ – мера зависимости между парой случайных величин. Предполагается, что значение меры зависимости $\gamma(X_i, X_j) = 0$ является признаком независимости случайных величин X_i, X_j . Сеть случайных величин порождает полный взвешенный граф Γ с N вершинами, в котором вес ребра (i, j) определяется величиной $\gamma_{i,j} = \gamma(X_i, X_j)$. Графом концентраций $Conc(\Gamma)$ графа Γ назовем граф с N вершинами, в котором ребро $(i, j) \in Conc(\Gamma)$ тогда и только тогда, когда $\gamma_{i,j} \neq 0$. Граф концентраций отражает графовую модель зависимостей компонент случайного вектора X , связанную с мерой зависимости γ . Классическая и наиболее популярная графовая модель связана с условными зависимостями компонент случайного вектора, что соответствует выбору меры частных корреляций, как меры зависимости. В настоящей работе рассматриваем граф концентраций в Гауссовских сетях случайных величин (вектор X имеет нормальное распределение) для корреляционных мер зависимости: классическая для этой задачи сеть частных корреляций (порождает Гауссовскую графическую модель, Gaussian Graphical Models), сеть корреляций Пирсона, сеть корреляций Фехнера и сеть корреляций Кендалла. Важно отметить, что для заданного случайного вектора X с многомерным нормальным распределением графы концентраций в сетях корреляций Пирсона, Фехнера и Кендалла совпадают. Этот верно и для более широкого класса эллиптических распределений [4].

2.1 Общая постановка задачи идентификации

Для заданной сети случайных величин (X, γ) и графа Γ Задача идентификации графа концентраций состоит в восстановлении графа $Conc(\Gamma)$ по наблюдениям. Под наблюдениями мы понимаем выборку размера n из распределения X . Моделью наблюдений является набор независимых одинаково распределенных случайных величин $X(1), X(2), \dots, X(n)$. Любой алгоритм идентификации графа концентраций можно рассматривать как отображение из выборочного пространства (sample space) в пространство неориентированных, невзвешенных простых графов (graph space). Неопределенность алгоритма идентификации связана с различием между графом $Conc(\Gamma)$ и графом, полученным в результате применения алгоритма идентификации. Качество алгоритма идентификации определяется численной характеристикой различия графов т.е. большим или малым значением неопределенности.

2.2 Методология оценки неопределенности

Мы рассматриваем задачу идентификации графа концентраций как задачу бинарной классификации ребер полного, неориентированного простого графа с N вершинами. Ребро (i, j) относим к классу “Yes”, если оно присутствует в графе $Conc(\Gamma)$, и относим его к классу “No”, если его нет в графе $Conc(\Gamma)$. Любой алгоритм идентификации графа концентраций в этом случае можно рассматривать как алгоритм классификации. Для оценки качества алгоритмов бинарной классификации традиционно используется стандартная матрица ошибок со следующими элементами: TP (True Positive, число верно классифицированных ребер класса “Yes”), TN (True Negative, число верно классифицированных ребер класса “No”), FP (False Positive, число ребер

класса “No”, ложно отнесенных к классу “Yes”), FN (False negative, число ребер класса “Yes”, ложно отнесенных к классу “No”). Для оценки качества алгоритмов идентификации графа концентраций по наблюдениям (по выборке размера n из распределения случайного вектора X) мы используем математические ожидания (по распределению X) следующих характеристик качества: True Negative Rate $TNR = TN / (TN + FP)$, True Positive Rate $TPR = TP / (TP + FN)$, False Discovery Rate $FDR = FP / (TP + FP)$, False Omission Rate $FOR = FN / (TN + FN)$. Эти четыре характеристики отражают разные аспекты качества алгоритмов идентификации. Для оценки математических ожиданий характеристик качества мы используем среднее значение по заданному числу повторений экспериментов. В качестве обобщенных характеристик качества алгоритмов идентификации графа концентраций мы рассматриваем три характеристики: классические Balanced Accuracy BA, F1 score, и сравнительно недавно ставший популярным Matthew Correlation Coefficient MCC.

2.3 Алгоритмы идентификации

В работе исследуются алгоритмы идентификации графа концентраций, основанные на множественной проверке гипотез. Задача идентификации графа концентраций в постановке множественной проверки гипотез может быть выражена следующим образом:

$$H_{ij}: \gamma_{ij} = 0 \text{ против } K_{ij}: \gamma_{ij} \neq 0, \quad i, j = 1, \dots, N.$$

Статистические тесты используются для определения нулевых значений попарной меры связи между X_i и X_j на заданном уровне значимости α . Если гипотеза была отвергнута, ребро (i, j) включается в множество ребер $Conc(\Gamma)$. В противном случае соответствующее ребро не включается в множество ребер. Для полной идентификации графа концентраций необходимо проверить C_N^2 гипотез, т.е. проверить каждую пару компонент случайного вектора X .

Для каждого конкретного вида сети случайных величин необходимы разные тестовые статистики. Детальное описание каждой из используемых тестовых статистик может быть найдено в [4].

Один тест контролирует вероятность ошибки первого рода на уровне значимости α . При этом при совместном применении всех индивидуальных тестов не контролирует FWER (вероятность допустить хотя бы одну ошибку первого рода на всем множестве из $M = C_N^2$ тестов). Для решения данной проблемы используются различные поправки на множественность гипотез, контролируемые FWER или FDR на заданном уровне значимости. В работе рассматриваются следующие алгоритмы множественной проверки гипотез:

Simultaneous Inference (SI): Алгоритм множественной проверки гипотез без контроля FWER и FDR.

1. Вычислить тестовые статистики T_{ij} и вычислить соответствующие им р-значения p_{ij} .
2. Сравнить полученные р-значения с выбранным уровнем значимости. Если $p_{ij} < \alpha$, ребро $(i, j) \in E$, иначе ребро не включается в множество ребер.

Bonferroni correction (B): Алгоритм множественной проверки гипотез с контролем FWER.

1. Вычислить тестовые статистики T_{ij} и вычислить соответствующие им р-значения p_{ij} .
2. Вычислить количество тестов $M = C_N^2$.
3. Сравнить полученные р-значения с $\frac{\alpha}{M}$. Если $p_{ij} < \frac{\alpha}{M}$, ребро $(i, j) \in E$, иначе ребро не включается в множество ребер.

Holm step down procedure (H): Алгоритм множественной проверки гипотез с контролем FWER.

1. Вычислить тестовые статистики T_{ij} и вычислить соответствующие им р-значения p_{ij} .
2. Упорядочить полученные р-значения по возрастанию $p_1 \leq \dots \leq p_M, M = C_N^2$.
3. Найти $R = \min \left\{ k: p_k > \frac{\alpha}{M+1-k} \right\}$ и отклонить все гипотезы, соответствующие $k = 1, \dots, R - 1$.
4. Для каждой отклоненной гипотезы добавить соответствующее ребро в множество ребер.

Benjamini-Hochberg procedure (BH): Алгоритм множественной проверки гипотез с контролем FDR для независимых тестовых статистик (не в случае задачи идентификации графа концентраций).

1. Вычислить тестовые статистики T_{ij} и вычислить соответствующие им р-значения p_{ij} .

2. Упорядочить полученные р-значения по возрастанию $p_1 \leq \dots \leq p_M, M = C_N^2$.
3. Найти $R = \max \left\{ k: p_k \leq \frac{\alpha k}{M} \right\}$ и отклонить все гипотезы, соответствующие $k = 1, \dots, R - 1$.
4. Для каждой отклоненной гипотезы добавить соответствующее ребро в множество ребер.

Benjamini-Yekutieli procedure (BY): Алгоритм множественной проверки гипотез с контролем FDR в общем случае (даже если тестовые статистики не независимы)

1. Вычислить тестовые статистики T_{ij} и вычислить соответствующие им р-значения p_{ij} .
2. Упорядочить полученные р-значения по возрастанию $p_1 \leq \dots \leq p_M, M = C_N^2$.
3. Найти $R = \max \left\{ k: p_k \leq \frac{\alpha k}{M c_M} \right\}, c_M = \sum_{l=1}^M \frac{1}{l}$ и отклонить все гипотезы, соответствующие $k = 1, \dots, R - 1$.
4. Для каждой отклоненной гипотезы добавить соответствующее ребро в множество ребер.

3. Генератор графовых моделей с заданной плотностью графа концентраций

Для оценки математического ожидания неопределенности алгоритмов идентификации графа концентраций необходим генератор случайных графовых моделей с заданной плотностью графа. Такой генератор должен создавать случайные положительно определенные матрицы с заданной плотностью ненулевых элементов.

В работе используется генератор матриц с доминирующей диагональю, предложенный в [5]. В начале работы генератор использует модель случайных графов Эрдеша-Реньи для создания случайной матрицы смежности с заданной плотностью. Затем ненулевым элементам матрицы присваиваются случайные веса из равномерного распределения на интервале $[-1, -0.5] \cup [0.5, 1]$, и каждый вес нормируется согласно правилу доминирующей диагонали. В конце происходит симметризация такой взвешенной матрицы смежности. Алгоритм создания случайной положительно определенной матрицы с заданной плотностью может быть описан следующим образом:

1. зафиксировать параметры N, p ;
2. получить матрицу смежности A , соответствующую случайному графу Эрдеша—Реньи $G(N, p)$;
3. вычислить матрицу B , где $B_{ij} = A_{ij} * Uniform([-1, 0.5] \cup [0.5, 1])$;
4. преобразовать матрицу C : $C_{ij} = \frac{B_{ij}}{1.5 * \sum_k B_{ik}}$;
5. получить результирующую матрицу симметризацией матрицы C : $\frac{C+C^T}{2}$.

Представленный генератор был реализован на языке Python с использованием библиотек Numpy и NetworkX.

```
def generateDominantDiagonal(dim: int, density: float) -> tuple:
    graph = nx.gnp_random_graph(dim, density)
    adj = nx.adjacency_matrix(graph).toarray()

    A = np.random.uniform(0.5, 1, size=(dim, dim))
    B = np.random.choice([-1, 1], size=(dim, dim))
    prec = adj * A * B
    rowsums = np.sum(np.abs(prec), axis=1)
    rowsums[rowsums == 0] = 0.0001

    prec = prec / (1.5 * rowsums[:, None])
    prec = (prec + prec.T) / 2 + np.eye(dim)

    precision = prec
    covariance = np.linalg.inv(precision)

    pD = np.diag(1 / np.sqrt(np.diag(precision)))
    pcorr = -(pD @ precision @ pD)
    np.fill_diagonal(pcorr, 1)

    return covariance, precision, pcorr
```

4. Численный анализ неопределенности идентификации графовых моделей

4.1 Описание численных экспериментов

Численный анализ неопределенности алгоритмов идентификации графа концентраций требует проведения массивных экспериментов на семействе случайных графовых моделей, созданных при помощи генератора матриц с доминирующей диагональю. Параметры экспериментов:

- размерность случайного вектора $N = 20$;
- размер выборки $n = 40, 100, 300$;
- распределение данных: нормальное распределение с нулевым средним;
- количество случайных графовых моделей $S_{sg} = 500$;
- количество повторений для выбранной графовой модели $S_{obs} = 100$;
- значения плотности графа: $d \in [0.1, 0.9]$ (20 равноудаленных точек на интервале);
- уровень значимости индивидуальных тестов $\alpha = 0.05$.

Один эксперимент проводится по следующей схеме:

1. зафиксировать плотность d ;
2. сгенерировать случайную графовую модель размерности N с заданной плотностью;
3. сгенерировать выборку размера n из распределения вектора X ;
4. применить алгоритмы идентификации графа концентраций к выборке и вычислить значения метрик качества;
5. повторить шаги 3–4 S_{obs} раз и усреднить полученные значения метрик;
6. повторить шаги 2–5 S_{sg} раз и усреднить полученные значения метрик качества для оценки математического ожидания неопределенности алгоритмов на семействе случайных графовых моделей.

Выполнение всех шагов является достаточно трудоемкой вычислительной задачей. Для решения данной проблемы мы используем возможности суперкомпьютера для распараллеливания вычислений, выполняемых для фиксированных графовых моделей. Таким образом, каждая из S_{sg} графовых моделей обрабатывается независимо от остальных. Затем, после того как все графовые модели будут обработаны, итоговые метрики качества собираются и усредняются. Такой параллелизм позволяет значительно уменьшить время выполнения экспериментов.

4.2 Результаты численных экспериментов

Результаты численных экспериментов представлены в виде графиков зависимости от плотности характеристик неопределенности алгоритмов множественной проверки гипотез для идентификации графа концентраций в различных сетях случайных величин для нормального распределения случайного вектора X . В экспериментах использован генератор случайных графовых моделей, основанный на принципе доминирующей диагонали. Характерные результаты, отражающие результаты в целом, представлена на рис. 1–4. Полные результаты всех проведенных экспериментов можно найти по ссылке: https://github.com/cofofprom/gms_uncertainty/blob/main/Dominant%20Diagonal%20density%20experiments.pdf.

На рис. 1–4 выбраны классическая для этой задачи сеть частных корреляций и сеть корреляций Кендалла. На рисунках используются следующие сокращения для алгоритмов: SI (Simultaneous Inference), B (Bonferroni), H (Holm), BH (Benjamini-Hochberg), BY (Benjamini-Yekutieli). Как можно видеть из представленных графиков, все алгоритмы обеспечивают высокое качество характеристики TNR идентификации класса “No” (отсутствие ребра). Это соответствует известным теоретическим свойствам алгоритмов, связанных с гарантированным контролем ошибки FWER (вероятность хотя бы одного ложного ребра). Все алгоритмы, кроме SI успешно контролируют FDR, что тоже соответствует известным теоретическим результатам. Вместе с тем, характеристики качества идентификации класса “Yes” (присутствие ребра в графе концентраций) являются весьма далекими от приемлемых. Кроме того, явно виден рост ошибки

FOR (ложное отсутствие ребра) с ростом плотности графа концентраций. Обобщенные характеристики BA, F1, MCC качества алгоритмов множественной проверки гипотез для идентификации графа концентраций показывают низкое качество алгоритмов, особенно при росте плотности графа. Отмеченные особенности характерны для идентификации графа концентраций во всех рассматриваемых сетях случайных величин.

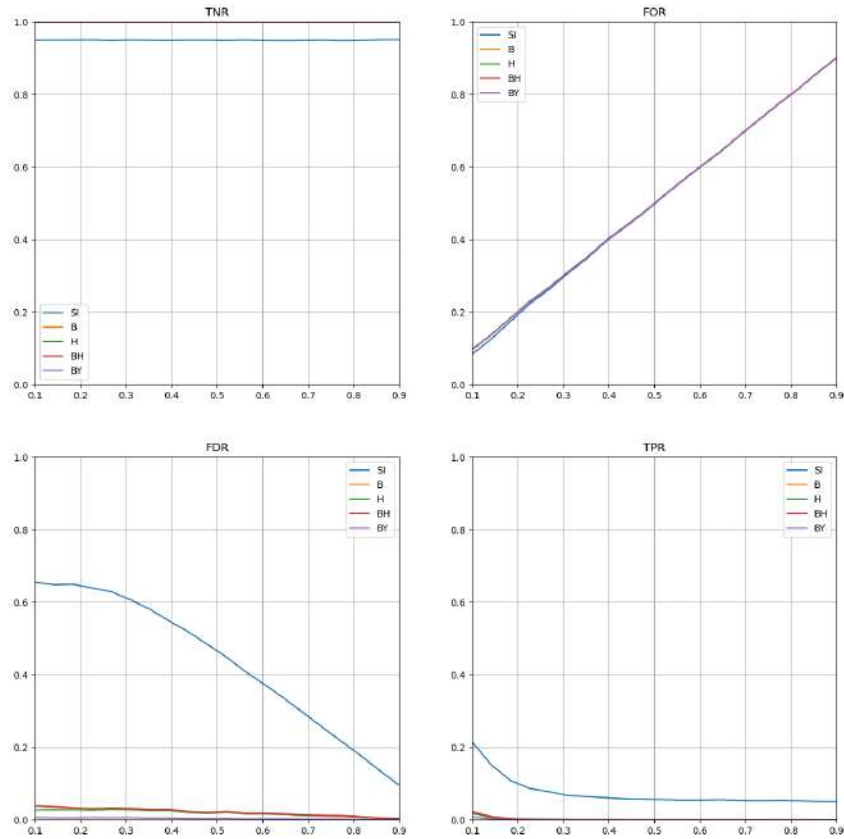


Рис. 1. Зависимость от плотности графа характеристик неопределенности TNR, FOR, FDR, TPR алгоритмов множественной проверки гипотез для идентификации графа концентраций в сети частных корреляций для нормального распределения вектора X и размера выборки $n=40$ (Gaussian Graphical Model Selection Problem)

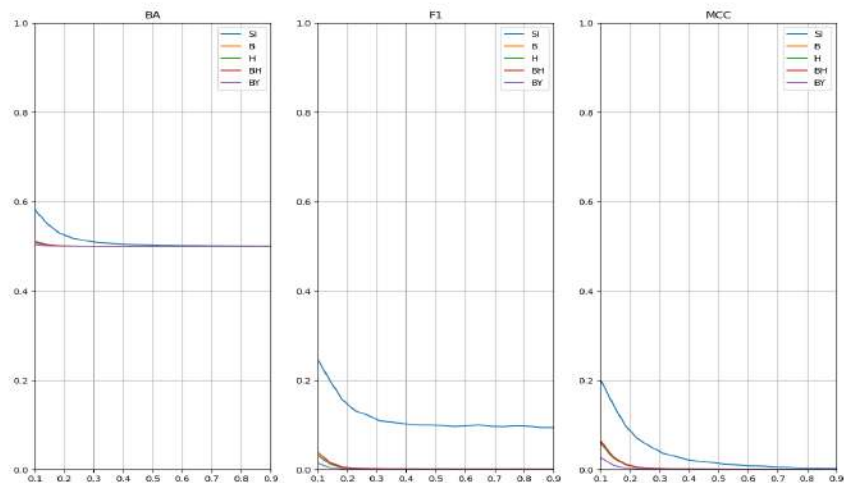


Рис. 2. Зависимость от плотности графа обобщенных характеристик неопределенности BA, F1, MCC алгоритмов множественной проверки гипотез для идентификации графа концентраций в сети частных корреляций для нормального распределения вектора X и размера выборки $n=40$ (Gaussian Graphical Model Selection Problem)

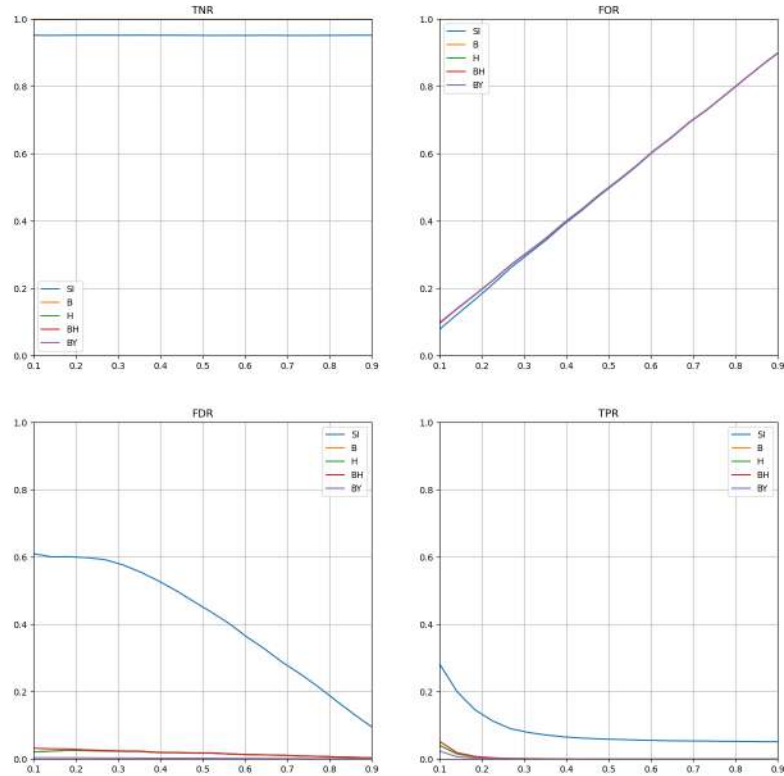


Рис. 3. Зависимость от плотности графа характеристик неопределенности TNR, FOR, FDR, TPR алгоритмов множественной проверки гипотез для идентификации графа концентраций в сети корреляций Кендалла для нормального распределения вектора X и размера выборки $n=40$ (Gaussian Graphical Model Selection Problem)

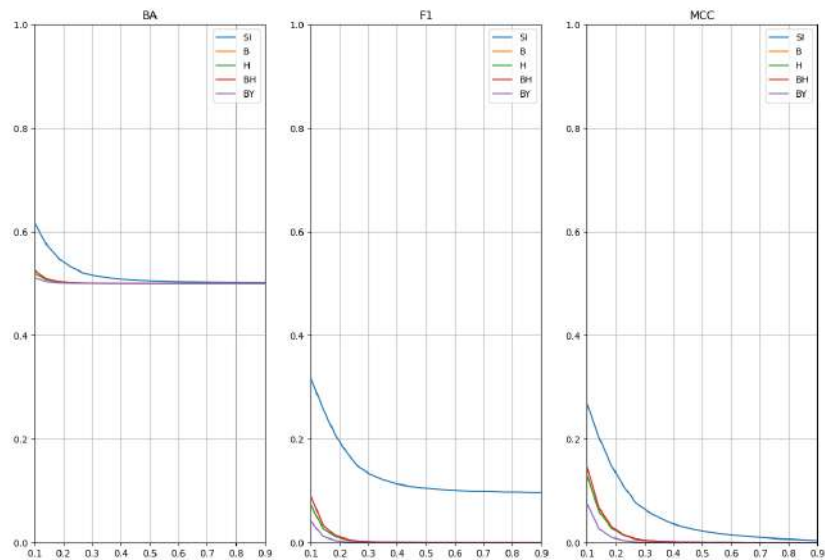


Рис. 4. Зависимость от плотности графа обобщенных характеристик неопределенности BA, F1, MCC алгоритмов множественной проверки гипотез для идентификации графа концентраций в сети корреляций Кендалла для нормального распределения вектора X и размера выборки $n=40$ (Gaussian Graphical Model Selection Problem)

5. Выводы

Настоящая работа является вкладом в исследование общей задачи анализа существующих и разработки новых алгоритмов идентификации графовых моделей в различных сетях случайных величин. В перспективе предполагается исследование устойчивости популярных алгоритмов и разработка новых устойчивых алгоритмов идентификации. Представленные вычислительные эксперименты направлены на изучение зависимости от плотности графа неопределенности алгоритмов множественной проверки гипотез для идентификации графа концентраций. Анализ результатов показывает, что независимо от выбранного генератора случайных графовых моделей общие тенденции зависимости от плотности качества алгоритмов множественной проверки гипотез совпадают и отражают специфику этих алгоритмов. В представленных результатах заметно резкое ухудшение качества алгоритмов идентификации с ростом плотности графа. Этот феномен требует дополнительного теоретического и экспериментального исследования для возможной адаптации алгоритмов множественной проверки гипотез при изменении плотности графа. Вычислительные эксперименты выполнены на суперкомпьютере CHARISMa НИУ ВШЭ в рамках проекта «графовые нейронные сети в задачах комбинаторной оптимизации».

Литература

1. Drton M., Maathuis M.H. Structure Learning in Graphical Modeling // *Annual Review of Statistics and Its Application*. 2017. Vol. 4. P. 365–393. DOI: 10.1146/annurev-statistics-060116-053803.
2. Cordoba I., Bielza C., Larranaga P. A review of Gaussian Markov models for conditional independence // *Journal of Statistical Planning and Inference*. 2020. Vol. 206. P. 127–144. DOI: 10.1016/j.jspi.2019.09.008.
3. Kalyagin V., Kostylev I. Graph Density and Uncertainty of Graphical Model Selection Algorithms // *Advances in Optimization and Applications. OPTIMA 2023*. Vol. 1913 / eds. by N. Olenov, Y. Evtushenko, M. Jaćimović, M. Khachay, V. Malkova. Springer, Cham, 2023. P. 188–201. Communications in Computer and Information Science. DOI: 10.1007/978-3-031-48751-4_14.
4. Kalyagin V.A., Koldanov A.P., Koldanov P., Pardalos P.M. Statistical Analysis of Graph Structures in Random Variable Networks. Springer, 2020. DOI: 10.1007/978-3-030-60293-2.
5. Peng J., Wang P., Zhou N., Zhu J. Partial Correlation Estimation by Joint Sparse Regression Models // *Journal of the American Statistical Association*. 2009. Vol. 104. P. 735–746 DOI: 10.1198/jasa.2009.0126.

Оценка эффективности различных форматов представления разреженных матриц для вычислений на графических ускорителях

Р.М. Куприй^{1,2}, Б.И. Краснополский¹, К.А. Жуков²

¹Институт механики МГУ имени М.В. Ломоносова,

²Факультет ВМК МГУ имени М.В. Ломоносова

Использование графических ускорителей является распространенным способом ускорения вычислений благодаря их более высокой производительности и пропускной способности памяти по сравнению с центральными процессорами. Они находят достаточно широкое применение для расчета векторных и матрично-векторных операций, являющихся составной частью многих вычислительных приложений, и в частности, итерационных методов решения систем линейных алгебраических уравнений. Эффективное использование графических ускорителей требует адаптации алгоритмов и форматов хранения матриц с учетом особенностей архитектуры ускорителей. В литературе за прошедшее время было предложено множество различных форматов хранения данных, но не все они одинаково эффективны при организации вычислений с матрицами общего вида на современных графических процессорах. В данной работе проводится сравнение ряда популярных форматов хранения разреженных матриц и соответствующих реализаций операции умножения матрицы на вектор. Выбранные форматы (Vector CSR, Adaptive CSR, ELL-T, ELLR-T, Sliced ELL) реализованы в рамках библиотеки численных методов XAMG на языке CUDA. Проведена серия численных экспериментов по оценке эффективности реализованных форматов для набора тестовых матриц из SuiteSparse Matrix Collection и ряда синтетических сгенерированных матриц. Эксперименты проводились как для обособленной операции умножения разреженной матрицы на вектор, так и в составе итерационного метода решения систем линейных алгебраических уравнений. В сравнении с библиотекой cuSPARSE реализованные алгоритмы умножения матрицы на вектор продемонстрировали сопоставимую производительность. Среди реализованных форматов предпочтительным оказался формат Adaptive CSR.

Ключевые слова: формат хранения данных, графические ускорители, умножение разреженной матрицы на вектор, системы линейных алгебраических уравнений.

1. Введение

Решение систем линейных алгебраических уравнений (СЛАУ) является часто встречающейся подзадачей при моделировании различных физических процессов. Этот этап вычислений может занимать значительную часть всего времени расчета. Поэтому важно уделять внимание эффективной реализации и оптимизации этой подзадачи вместе с развитием современных вычислительных систем.

Возникающие в задачах математической физики системы уравнений зачастую имеют большие разреженные матрицы. Характерное число ненулевых элементов в строке матрицы колеблется в диапазоне 10^1 – 10^2 , тогда как размер матрицы может составлять 10^5 – 10^{10} или более. Для разреженных матриц обычно хранятся только значения ненулевых элементов и информация об их расположении в матрице. Такой способ организации информации называется форматом хранения разреженных матриц.

Для решения систем уравнений широко используются различные итерационные численные методы [1]. Составным элементом этих методов является операция умножения раз-

реженной матрицы на вектор (SpMV):

$$b = Ax.$$

Здесь, $A \in \mathbb{R}^{n \times n}$ – матрица размера $n \cdot n$, $x, b \in \mathbb{R}^n$ – плотные вектора. Операция SpMV, а вследствие и сами итерационные методы, обладает низкой арифметической интенсивностью и относится к классу алгоритмов, чья производительность лимитируется в первую очередь пропускной способностью шины памяти вычислительной системы [2].

В настоящее время для ускорения вычислений приобрели широкое распространение графические ускорители (GPU). Популярность GPU обусловлена на порядки большей производительностью в сравнении с центральными процессорами. Графические ускорители обладают тысячами маломощных вычислительных ядер и высокоскоростной шиной памяти, что позволяет значительно ускорить выполнение алгоритмов, ограниченных не только вычислительными мощностями процессора, но и пропускной способностью шины памяти. Поскольку к последним алгоритмам относится и операция SpMV, то использование GPU может заметно ускорить решение СЛАУ.

Для достижения максимальной эффективности вычислений при реализации алгоритмов необходимо учитывать особенности архитектуры графических ускорителей. Особенно важна организация хранения данных и обеспечение свойств локальности при вычислениях на GPU. Для операции SpMV характер доступа к данным в памяти зависит от формата хранения разреженной матрицы, что оказывает прямое влияние на итоговую производительность вычислений. Существующие форматы различаются, прежде всего, способом представления дополнительной информации о расположении ненулевых элементов в матрице. Многообразие способов представления этой информации порождает большое количество разных форматов хранения, использующих специфические свойства матриц.

Одним из наиболее широко используемых и простых в реализации форматов хранения разреженных матриц является формат Compressed Sparse Row (CSR) [1, 3]. Этот формат применяется в таких известных библиотеках численных методов, как hypre [4] или PETSc [5]. Формат CSR универсален и прост в реализации, но не всегда демонстрирует оптимальную производительность. Поэтому были разработаны многочисленные модификации и вариации формата для повышения эффективности операций с разреженными матрицами. Для центральных процессоров был разработан ряд модификаций, например, Jagged Diagonal [6], Blocked CSR [7] или CSR5 [8]. Однако, эти форматы не учитывают особенности архитектуры ускорителей, поэтому для организации вычислений на GPU они мало применимы.

Другим популярным форматом является ELLPACK (ELL) [9]. Этот формат также прост в реализации, как и CSR, но более предпочтителен при использовании графических ускорителей, поскольку хорошо подходит для потоковой обработки. Он менее универсален в сравнении с CSR и применим только для матриц с ограниченным разбросом числа ненулевых элементов в строках матрицы. Значительное отклонение среднего числа ненулевых элементов в строках матрицы от максимального приводит к неэффективному использованию памяти и снижению производительности операций с матрицами. Для расширения области применимости этого формата и повышения производительности операций с разреженными матрицами при использовании графических ускорителей со временем были разработаны различные модификации, такие как ELLR-T [10], Sliced ELL [11] или SELL-C- σ [12].

Формат Adaptive CSR [13] является примером специализированного формата, разработанного для выполнения вычислений типа умножения матрицы на вектор на графических ускорителях. Этот формат является модификацией формата CSR и ориентирован на увеличение эффективности при работе с матрицами с нерегулярной топологией расположения ненулевых элементов.

Представленное в литературе множество форматов хранения разреженных матриц не позволяет однозначно ответить на вопрос о преимуществах и недостатках того или иного

формата на репрезентативном наборе матриц, и в частности для итерационных методов решения СЛАУ, включая многосеточные методы. В настоящей работе рассматривается реализация перечисленных форматов хранения матриц и операции умножения матрицы на вектор на современных графических ускорителях, и на наборе существенно различных тестовых матриц исследуется их эффективность при выполнении операций умножения матрицы на вектор и при решении СЛАУ. Оставшаяся часть текста организована следующим образом. Во втором разделе статьи представлены реализованные в ходе проведенной работы форматы хранения данных и соответствующие алгоритмы умножения матрицы на вектор на GPU. В третьем разделе приведены детали программной реализации алгоритмов и методология проведенных численных экспериментов. Соответствующие результаты оценки производительности описаны в четвертом разделе. В заключении приведено обобщение полученных результатов.

2. Форматы хранения

В следующих параграфах описаны представленные в данной работе форматы хранения разреженных матриц и соответствующие алгоритмы операции SpMV на GPU. Реализации форматов приводятся на примере матрицы, изображенной на рис. 1.

$a_{0,0}$			$a_{0,3}$		
	$a_{1,1}$			$a_{1,4}$	$a_{1,5}$
		$a_{2,2}$			
			$a_{3,3}$		
$a_{4,0}$		$a_{4,2}$		$a_{4,4}$	
		$a_{5,2}$			$a_{5,5}$

$n = 6, nnz = 12$

Рис. 1. Пример разреженной матрицы

2.1. CSR

В формате CSR для представления разреженной матрицы используется три массива, *Val*, *Col* и *Row*. Массив *Val* содержит значения всех ненулевых элементов матрицы, упорядоченных построчно, а в массиве *Col* хранятся номера столбцов соответствующих элементов. Размер этих массивов равен количеству ненулевых элементов, *nnz*. Массив *Row* размером $n + 1$ содержит смещения от начала матрицы для первых ненулевых элементов в каждой строке, а также общее число ненулевых элементов. Содержимое этих массивов при представлении тестовой матрицы приведено на рис. 2.

Простейший вариант реализации операции SpMV на центральном процессоре для матрицы в формате CSR показан на рис. 3. Базовый алгоритм SpMV для GPU заключается в обработке каждой строки матрицы одним потоком. Такой способ является не эффективным, поскольку не позволяет задействовать ресурсы ускорителя в достаточной мере. Более эффективным является векторный алгоритм SpMV, предложенный в [14]. Он заключается в выделении целого варпа (группы вычислительных нитей) на обработку каждой строки матрицы. Однако, в настоящее время существуют различные одноименные вариации векторного алгоритма. Так, например, для более эффективной работы на современных GPU

Row	0	2	5	6	7	10	12
Col	0 3	1 4 5	2	3	0 2 4	2 5	
Val	$a_{0,0}$ $a_{0,3}$	$a_{1,1}$ $a_{1,4}$ $a_{1,5}$	$a_{2,2}$	$a_{3,3}$	$a_{4,0}$ $a_{4,2}$ $a_{4,4}$	$a_{5,2}$ $a_{5,5}$	

Рис. 2. Пример представления тестовой матрицы в формате CSR

```
for (i = 0; i < n; i++) {
    for (j = Row[i]; j < Row[i+1]; j++)
        y[i] += x[Col[j]] * Val[j];
}
```

Рис. 3. Реализация операции SpMV для формата CSR на CPU

один варп может выделяться для обработки сразу нескольких строк разреженной матрицы. Каждый варп делится на несколько групп последовательно пронумерованных потоков – векторов, каждый из которых обрабатывает свою строку матрицы. Размер варпов фиксирован и составляет 32 нити, в то время как размер вектора является гиперпараметром и определяется в зависимости от степени разреженности матрицы – от среднего числа ненулевых элементов в строке. В такой вариации векторного алгоритма используется два дополнительных параметра: *lane* – номер нити в векторе и *TPV* – число потоков в одном векторе. Размер вектора является делителем 32 и подбирается таким образом, чтобы наиболее равномерно распределить вычислительную нагрузку между потоками. Такой подход позволяет наиболее полно задействовать вычислительные ресурсы графического ускорителя.

```
int thread_id = blockDim.x * blockIdx.x + threadIdx.x;
int lane = threadIdx.x & (TPV - 1);
int row = thread_id / TPV;
if (row < nrows) {
    double sum = 0.0;
    for (j = Rows[row] + lane; j < Rows[row+1]; j+=TPV)
        sum += Vals[j] * x[Cols[j]];

    #pragma unroll
    for (int i = TPV >> 1; i > 0; i >= 1)
        sum += __shfl_down_sync(0xffffffff, sum, i, TPV);

    if (lane == 0)
        y[row] += sum;
}
```

Рис. 4. Реализация векторного алгоритма SpMV для формата CSR на GPU

В общем виде код описанного алгоритма умножения матрицы на вектор представлен на рис. 4. Для накопления промежуточных сумм используется разделяемая переменная *sum*, а для получения конечного результата применяется схема редукции с накоплением суммы в одном потоке каждого вектора посредством примитива уровня варпа. У данного алгоритма есть существенный недостаток, который может привести к значительному снижению производительности вычислений: если в разреженной матрице имеется большой разброс числа ненулевых элементов в строках, то часть потоков выполнит обработку строк раньше потоков, обрабатывающих наиболее длинные строки. Такой дисбаланс приводит к простоям вычислительных ядер графического ускорителя и, следовательно, к замедлению вычислений.

2.2. Adaptive CSR

Формат Adaptive CSR является модификацией классического формата CSR, которая направлена на балансировку загрузки графического ускорителя при выполнении операций с разреженными матрицами. Для этого последовательные строки матрицы объединяются в группы. Каждая группа строк обрабатывается отдельным блоком потоков по определенному алгоритму. Таким образом, для обработки коротких и длинных строк выделяется разное число потоков, не синхронизирующихся друг с другом. При этом, количество строк в группе зависит от размера общей памяти, выделяемой для каждого блока. Размер выделяемой общей памяти и число потоков в блоке являются гиперпараметрами.

Помимо трех массивов, используемых в формате CSR, в адаптивном формате используется два дополнительных массива: *Row_blocks* и *Block_threads*. Размеры этих массивов равны $k + 1$ и k соответственно, где k – число групп строк. В первом массиве хранятся номера первых строк в каждой группе, а последний элемент соответствует числу строк матрицы. Во втором массиве хранится число потоков для обработки одной строки в каждой группе. Пример заполнения массивов данных формата Adaptive CSR для тестовой матрицы приведен на рис. 5.

<i>Row_blocks</i>	0					3					6				
<i>Block_threads</i>	4					4									
<i>Row</i>	0	2				5	6	7	10				12		
<i>Col</i>	0	3	1	4	5	2	3	0	2	4	2	5			
<i>Val</i>	$a_{0,0}$	$a_{0,3}$	$a_{1,1}$	$a_{1,4}$	$a_{1,5}$	$a_{2,2}$	$a_{3,3}$	$a_{4,0}$	$a_{4,2}$	$a_{4,4}$	$a_{5,2}$	$a_{5,5}$			

Рис. 5. Пример хранения матрицы в формате Adaptive CSR

В зависимости от количества выделенных потоков на одну строку, для обработки группы строк используется один из трех алгоритмов: потоковый алгоритм SpMV, векторный алгоритм SpMV или отдельный алгоритм для обработки сверхдлинных строк матрицы. Если в группе больше двух строк, то применяется потоковый алгоритм SpMV: каждый поток выполняет умножение элемента матрицы на элемент вектора и сохраняет результат в общую память, а затем выполняется редукция по строке из общей памяти. Если для обработки строки выделяется один поток, то используется линейная редукция, иначе используется комбинированная редукция.

Если в группе одна или две строки, то для их обработки применяется векторный алгоритм SpMV. Для формата Adaptive CSR используется следующая реализация векторного алгоритма: для обработки строки выделяется целый блок потоков, используется общая память для сохранения вычисленных значений и применяется древовидная редукция по блоку. Обработка строки целым блоком означает, что для этой задачи выделяются все потоки блока. Такой вид алгоритма обусловлен тем, что в такие маленькие группы объединяются достаточно длинные строки.

В том случае, когда число ненулевых элементов в строке превышает размер выделенной общей памяти для одного блока, используется алгоритм обработки строк с редукцией через глобальную память. Для краткости изложения обобщенный код алгоритма не представлен в настоящей работе, с ним можно ознакомиться в оригинальной статье [13].

2.3. ELL

В формате ELL используется всего два массива *Val* и *Col* для представления разреженной матрицы. Ненулевые элементы каждой строки хранятся в виде пар *<значение, номер столбца>*. Все строки дополняются нулями до одинаковой длины, равной максимальному количеству ненулевых элементов в строке, *width*. Размер используемых массивов равен *width · nrow*s, где *nrow*s – количество строк. Пример представления тестовой матрицы в формате ELL показан на рис. 6.

<i>Col</i>			<i>Val</i>			<i>Row_len</i>	<i>Slice_start</i>
0	3	0	$a_{0,0}$	$a_{0,3}$	0	2	0
1	4	5	$a_{1,1}$	$a_{1,4}$	$a_{1,5}$	3	
2	0	0	$a_{2,2}$	0	0	1	
3	0	0	$a_{3,3}$	0	0	1	9
0	2	4	$a_{4,0}$	$a_{4,2}$	$a_{4,4}$	3	
2	5	0	$a_{5,2}$	$a_{5,5}$	0	2	
							18

Рис. 6. Пример использования массивов *Col*, *Val*, *Row_len* и *Slice_start* для хранения матрицы в форматах ELL, ELLR и Sliced ELL

Как и в случае с базовым алгоритмом SpMV для формата CSR, обработка каждой строки матрицы одним потоком графического ускорителя не эффективна. Формат ELL, для которого реализуется операция SpMV с выделением нескольких потоков для обработки одной строки называется ELL-T. Число потоков *T* является гиперпараметром и подбирается в зависимости от характеристик матрицы и используемого графического ускорителя. В этом алгоритме используется объединение потоков в группы – вектора, как и в векторном алгоритме SpMV формата CSR. Обобщенный код алгоритма умножения матрицы на вектор в формате ELL-T представлен на рис. 7.

```

int thread_id = blockDim.x * blockIdx.x + threadIdx.x;
int lane = thread_id & (32 - 1);
int row = thread_id / TPV;
double temp_sum = 0.0;
if (row < nrow) {
    for (int j = lane; j < width; j += TPV) {
        int idx = row * width + j;
        temp_sum += Vals[idx] * x[Cols[idx]];
    }

    for (int i = TPV >> 1; i > 0; i >>= 1)
        temp_sum += __shfl_down_sync(0xffffffff, temp_sum, i, TPV);

    if (lane == 0)
        y[row] += temp_sum;
}

```

Рис. 7. Реализация операции SpMV для формата ELL-T на GPU

2.4. ELLR

Формат ELLR является модификацией базового формата ELL. В этой модификации используется вспомогательный массив *Row_len* размера *nrows*, в котором хранится число ненулевых элементов в каждой строке. Пример заполнения массива для тестовой матрицы показан на рис. 6. С использованием этого массива в операции SpMV обрабатываются только ненулевые элементы матрицы, благодаря чему задействованные потоки не совершают лишнюю работу. При этом, для сохранения выровненного и равномерного доступа к элементам массивов *Val* и *Col* дополнение строк нулевыми элементами остается. Такой подход позволяет в некоторых случаях уменьшить конкуренцию потоков за вычислительные ресурсы GPU.

Как и в случае с форматом ELL, формат ELLR и соответствующий алгоритм умножения матрицы на вектор с использованием нескольких потоков для обработки одной строки называется ELLR-T. Ввиду отсутствия принципиальных отличий в реализации алгоритма, обобщенный код опущен для краткости изложения.

2.5. Sliced ELL

Формат Sliced ELL также является модификацией формата ELL. Эта модификация направлена на уменьшение заполнения строк нулевыми элементами по максимальной длине строки ненулевых элементов матрицы. Для этого, матрица делится на полосы заданного одинакового размера, для каждой полосы вычисляется максимальное число ненулевых элементов и организуется заполнение строк в каждой полосе. Для адресации начала каждой полосы используется дополнительный массив *Slice_start*. Он содержит смещения первых ненулевых элементов в каждой полосе от начала матрицы, а также общее число элементов матрицы. Размер массива определяется в зависимости от числа строк матрицы и длины одной полосы. Пример хранения использования вспомогательного массива при хранении разреженной матрицы приведен на рис. 6. Также возможна дальнейшая модификация этого формата путем сортировки строк и выделения полос разного размера для дальнейшего сокращения потребления памяти при заполнении строк нулевыми элементами.

3. Методика тестирования

Результаты, представленные в настоящей работе, были получены на рабочей станции с 8-ядерным процессором Intel Core i7-11700, двухканальной оперативной памятью DDR4 2933 МГц и видеокартой NVIDIA GeForce RTX 3060ti. Фактическая пропускная способность шины памяти для центрального процессора на тестах типа *stream* составляет 43 GB/s, а для видеокарты – 417 GB/s. Для компиляции библиотеки использовалась библиотека OpenMPI версии 4.1.2, компилятор GCC версии 11.4.2 с набором флагов оптимизации `-O3 -ffast-math -funroll-loops -ftree-vectorize -march=rocketlake -mtune=rocketlake -mavx2 -m64` и CUDA SDK версии 12.5.

Все проведенные численные эксперименты на CPU используют все доступные физические ядра и учитывают привязку процессов MPI к ядрам процессора. При проведении расчетов на графическом ускорителе использовался один MPI процесс, взаимодействовавший с GPU.

3.1. Детали реализации

Исследуемые форматы реализованы в библиотеке численных методов XAMG [15, 16]. Эта библиотека предназначена для решения больших разреженных СЛАУ со многими правыми частями, в том числе возникающих в результате дискретизации эллиптических дифференциальных уравнений. В ней реализован набор численных методов, включающий в

себя некоторые итерационные методы подпространства Крылова, алгебраический многосеточный метод и базовые методы типа Якоби или гибридного симметричного Гаусса-Зейделя, которые, в том числе, используются вместе с многосеточным методом в качестве сглаживателей. Библиотека обеспечивает иерархическое трехуровневое распараллеливание с гибридной моделью параллельного программирования MPI + Posix Shared Memory с учетом аппаратных особенностей NUMA-архитектуры.

Библиотека XAMG написана на языке C++ (в соответствии со стандартом C++11) с использованием шаблонов, что позволяет реализовывать форматы хранения матриц с шаблонными типами данных. Часть библиотеки написана на языке CUDA для использования графических ускорителей для проведения вычислений.

3.2. Тестовые матрицы и сценарии тестирования

Для оценки эффективности обсуждаемых форматов хранения данных при выполнении операции SpMV на графическом ускорителе проведено несколько серий численных экспериментов. Проведенные эксперименты включают в себя как оценку времени выделенной операции, умножения матрицы на вектор, так и решение систем линейных алгебраических уравнений. Для получения репрезентативной картины сопоставления различных форматов использовался набор тестовых матриц из коллекции Suite Sparse Matrix Collection (SSMC) [18] и несколько матриц, сгенерированных с помощью генератора матриц библиотеки XAMG. SSMC содержит большой набор матриц из разных предметных областей. Для проведения тестов было взято некоторое подмножество матриц, для которых можно было получить устойчивое решение с помощью выбранного численного метода (стабилизированный метод би-сопряженных градиентов с классическим алгебраическим многосеточным предобуславливателем). При этом время выполнения операции SpMV на GPU с выбранными матрицами варьируется в диапазоне от нескольких микросекунд до нескольких миллисекунд. Более подробное описание набора используемых матриц приведено в [20].

Тестирование эффективности различных форматов было выполнено в три этапа. На первом этапе проведено сравнение времени выполнения операции умножения матрицы на вектор для алгоритма Vector CSR, реализованного в библиотеке XAMG, и времени расчета этой операции в библиотеке NVidia cuSPARSE. На втором этапе проведены замеры времени умножения матрицы на вектор для форматов CSR, Adaptive CSR, ELL-T, ELLR-T и Sliced ELL, реализованных в библиотеке XAMG. На финальном этапе тестирования проведены численные эксперименты по оценке производительности алгоритма решения СЛАУ на центральном процессоре и графическом ускорителе. В этом случае выполнялся расчет фиксированного числа итераций численного метода. Все тесты выполнялись для чисел с плавающей точкой, представленных в двойной точности.

4. Численные эксперименты

4.1. Сравнение производительности операции SpMV в библиотеках XAMG и cuSPARSE

Для подтверждения релевантности представленных далее результатов на первом этапе тестирования получены результаты сопоставления эффективности базовой реализации векторного варианта алгоритма SpMV для формата CSR в библиотеках XAMG и NVidia cuSPARSE. Времена расчета для библиотеки NVidia cuSPARSE, нормированные на времена для библиотеки XAMG, представлены на рис. 8. Значения выше 1 означают более высокую производительность библиотеки XAMG; значения меньше 1 указывают на преимущество библиотеки NVidia cuSPARSE.

За исключением матрицы #14, сравнение результатов показывает превосходство библиотеки XAMG над cuSPARSE в среднем на 13%. Анализ матрицы #14 показал, что она

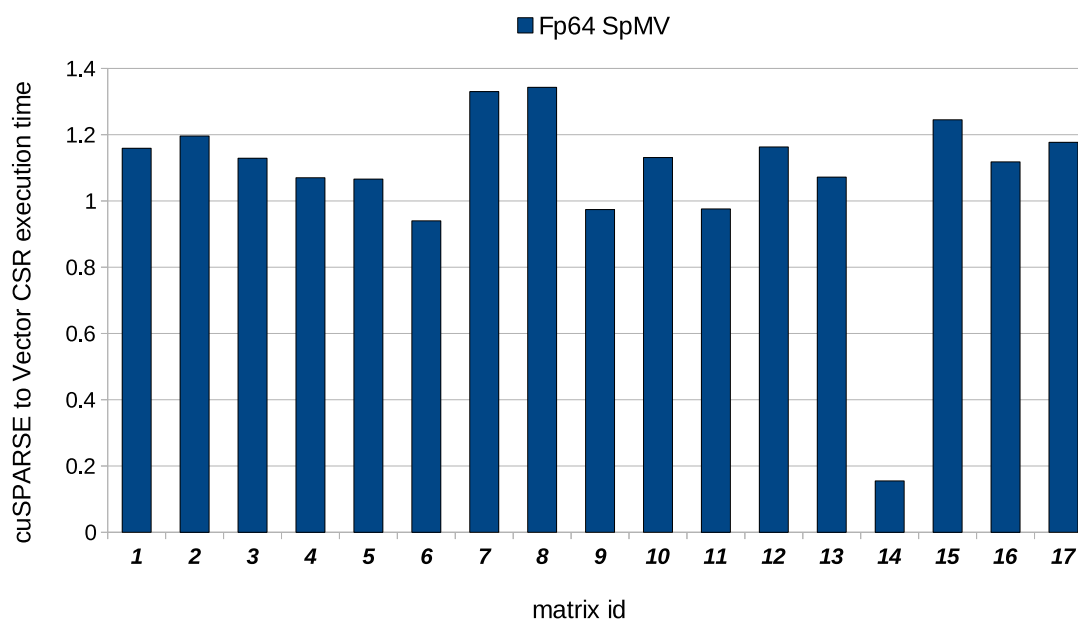


Рис. 8. Отношение времени выполнения операции умножения матрицы на вектор для формата CSR в библиотеках NVidia cuSPARSE и XAMG

обладает группой строк с кратно большим числом ненулевых элементов по сравнению со средним числом ненулевых элементов в матрице. В базовом варианте формата CSR с векторным алгоритмом SpMV нет балансировки загрузки ресурсов GPU, что и приводит к низкой производительности реализации операции SpMV в библиотеке XAMG для данной матрицы. Полученные результаты тестирования позволяют сделать вывод, что производительность векторного варианта алгоритма SpMV для формата CSR в библиотеке XAMG сопоставима с производительностью алгоритмов библиотеки NVidia cuSPARSE.

4.2. Сравнение производительности операции SpMV для различных форматов хранения разреженных матриц

Для реализованных в библиотеке XAMG форматов хранения матриц CSR, Adaptive CSR, ELL-T, ELLR-T и Sliced ELL проведено тестирование алгоритмов умножения матрицы на вектор. Полученные результаты тестирования представлены на двух графиках на рис. 9 и 10. На первом графике приведено относительное ускорение для формата Adaptive CSR в сравнении с Vector CSR. За исключением матрицы #14, адаптивный формат, в среднем, оказывается медленнее на 2%. При этом, разброс результатов составляет $\pm 10\%$. Это объясняется разницей в эффективности подобранных гиперпараметров используемых алгоритмов для каждой конкретной матрицы. Применение формата Adaptive CSR для матрицы #14 позволяет ускорить выполнение операции SpMV более чем в 6 раз. Такой результат достигается за счет рационального использования ресурсов GPU – длинные строки обрабатываются отдельно от коротких выделенным блоком потоков, что позволяет сбалансировать загрузку графического ускорителя.

На втором графике (рис. 10) приведены отношения времен выполнения операции SpMV в формате Vector CSR к ELL-T, ELLR-T и Sliced ELL. В среднем, использование формата ELL-T оказывается на 2% медленнее векторного алгоритма SpMV для формата CSR. Для матриц #10 и #15 использование дополнительной памяти для обеспечения выровненного доступа к элементам массива приводит к значительному замедлению вычислений, что обусловлено большим объемом лишних вычислений при обработке строк матрицы. Для остальных матриц результаты тестов сопоставимы с форматом CSR, поскольку отношение

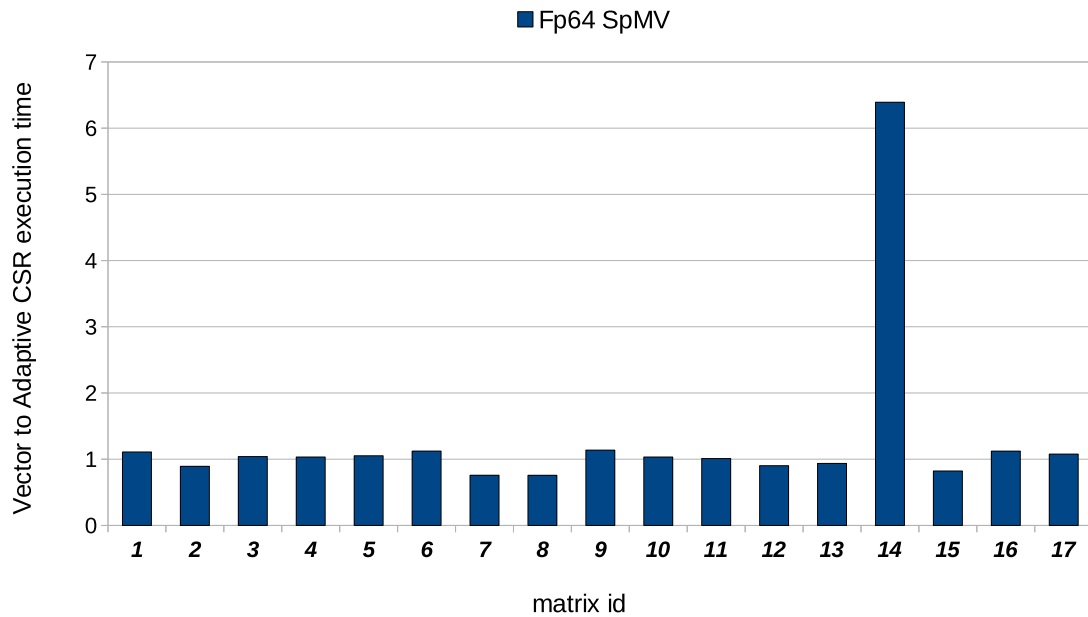


Рис. 9. Отношение времени выполнения операции умножения матрицы на вектор для форматов Vector CSR и Adaptive CSR в библиотеке XAMG

среднего числа ненулевых элементов в строке к максимальному числу ненулевых элементов составляет не менее 0.85.

Для модификации ELLR-T наблюдается замедление в среднем на 3% в сравнении с Vector CSR. При этом, для матриц #10 и #15 использование вспомогательного массива позволяет избавиться от лишних вычислений с нулевыми элементами при обработке строк, поэтому операция SpMV выполняется быстрее в сравнении с оригинальным форматом ELL-T. Однако, для остальных матриц, использование дополнительного массива для устранения лишней работы потоков не столь эффективно, и лишь замедляет вычисления из-за роста накладных расходов на чтение элементов массива *Row_len* и увеличения числа используемых регистров при выполнении алгоритма на GPU.

Использование формата Sliced ELL приводит к замедлению вычислений в среднем на 15% при выполнении операции SpMV. Для матриц #10 и #15 раздельное заполнение отдельных полос матрицы нулевыми элементами также помогает увеличить эффективность вычислений в сравнении с форматом ELL-T. Однако, в сравнении с форматом CSR все еще наблюдается значительное замедление. Для остальных матриц также наблюдается замедление выполнения операции SpMV в сравнении с CSR. Разделение матриц на полосы и использование вспомогательного массива приводит к усложнению алгоритма SpMV и возникновению накладных расходов на чтение его элементов. Экономия памяти в данном случае не в полной мере компенсирует потери производительности при вычислениях на GPU, что приводит к итоговому замедлению расчета.

Таким образом, на основе проведенной серии тестов можно сделать вывод о том, что использование форматов Adaptive CSR и ELL-T является предпочтительным для рассмотренного набора тестовых матриц. Форматы Adaptive CSR и ELL-T демонстрируют уровень производительности, сопоставимый с Vector CSR, однако при этом обладают большей универсальностью и менее чувствительны к наличию в матрице единичных строк с большим числом ненулевых элементов. Формат ELLR-T является более сложной модификацией формата ELL-T, но при этом на рассмотренном множестве тестовых матриц он не демонстрирует никакого прироста производительности. Наконец, формат Sliced ELL демонстрирует худшие результаты по времени выполнения операции умножения матрицы на вектор среди рассмотренных форматов.

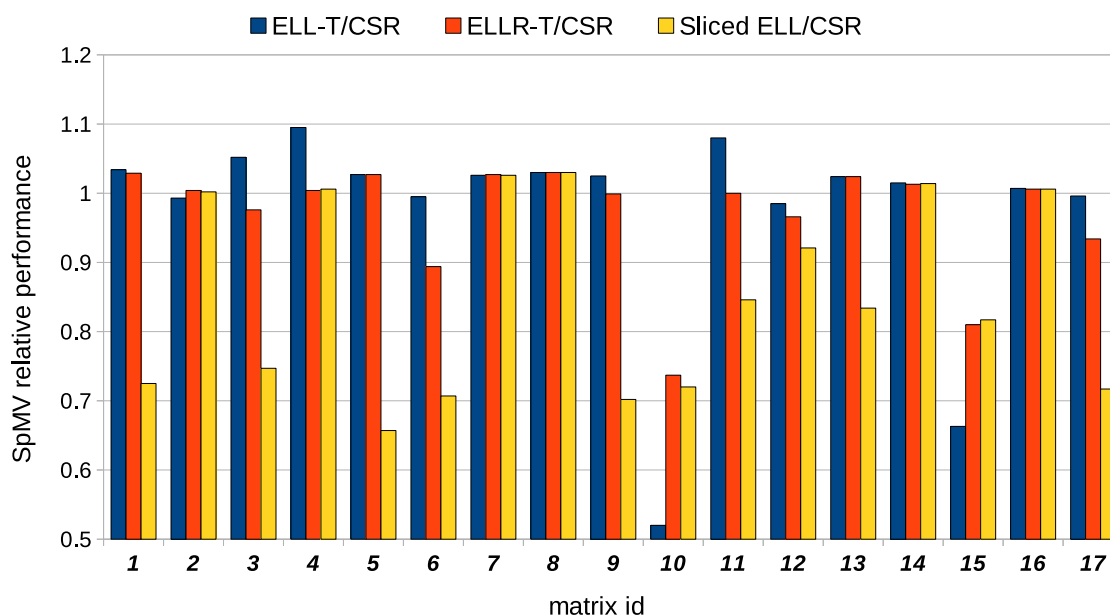


Рис. 10. Результаты сравнения производительности операции SpMV при использовании модификаций формата ELL

4.3. Решение систем линейных алгебраических уравнений

Третья серия экспериментов проведена для оценки эффективности использования формата Adaptive CSR на GPU в сравнении с форматом CSR на CPU при решении СЛАУ. Потенциальное ускорение при переносе вычислений можно оценить с помощью соотношения пропускной способности шины памяти графического ускорителя и центрального процессора. Для используемой вычислительной системы это соотношение составляет порядка 10. Однако, итерационные методы подпространства Крылова и многосеточный метод совмещают множество операций с векторами и операции умножения матрицы на вектор. Для каждой из этих операций реализуется вызов отдельного CUDA-ядра, что сопряжено с определенными накладными расходами. Это приводит к тому, что при проведении вычислений с матрицами нижних уровней многосеточного метода использование графических ускорителей может быть не целесообразным [17].

Полученные в ходе тестирования результаты для набора из 17 матриц приведены на рис. 11. Данные на графике представлены в виде отношения времен вычислений на центральных процессорах и графических ускорителях. Приведенные результаты демонстрируют среднее ускорение расчетов в 5.2 раза, что ниже оценки, основанной на отношении пропускной способности памяти. Это, с одной стороны, демонстрирует возможность существенного ускорения расчетов в библиотеке XAMG при использовании графических ускорителей, а с другой – иллюстрирует необходимость дальнейших оптимизаций кода в части миграции потока вычислений с графического ускорителя на центральный процессор, а также уменьшения числа вызовов CUDA-ядер за счет внедрения технологии слияния ядер (loop fusion) для достижения результатов, близких к теоретическим оценкам. Указанные вопросы будут являться предметом дальнейших исследований.

5. Заключение

В работе рассмотрено несколько форматов хранения разреженных матриц при вычислениях на графических ускорителях: Vector CSR, Adaptive CSR, ELL-T, ELLR-T и Sliced ELL. Рассмотренные форматы и соответствующие алгоритмы для операции умножения матри-

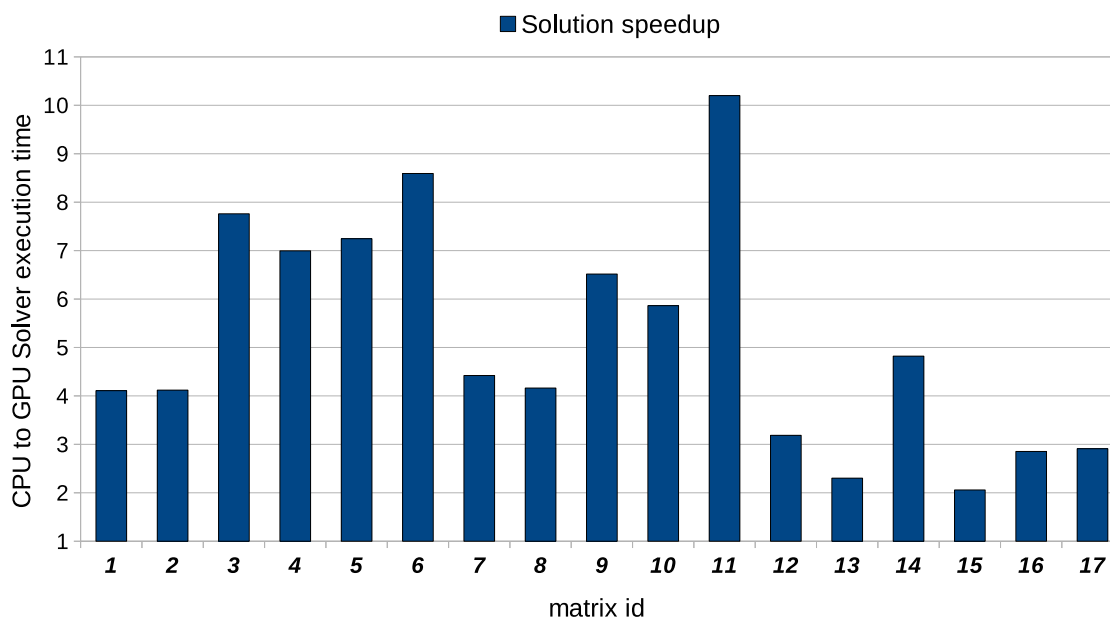


Рис. 11. Ускорение решения СЛАУ при использовании графического ускорителя вместо центрального процессора

цы на вектор реализованы в библиотеке XAMG на языке CUDA. Для набора из 17 матриц проведена оценка времени выполнения операции умножения матрицы на вектор при использовании разных форматов хранения данных и исследовано потенциальное ускорение решения СЛАУ при переходе в расчетах от центральных процессоров к графическим ускорителям.

Проведенные численные эксперименты показали, что среди рассмотренных форматов наиболее эффективным и универсальным является формат Adaptive CSR. Производительность операции SpMV с адаптивным алгоритмом сопоставима с векторным вариантом формата CSR. Главным его преимуществом является возможность эффективно обрабатывать матрицы с существенным дисбалансом числа ненулевых элементов в строках. Формат ELL также позволяет получать на большинстве матриц близкие времена для операции умножения матрицы на вектор, однако в отдельных случаях наблюдается заметное увеличение времени расчета. Рассмотренные модификации формата ELL несколько уступают по производительности исходному формату. Эффективность ELL-подобных форматов сильно зависит от характеристик разреженной матрицы, в частности от соотношения среднего и максимального числа ненулевых элементов в строке.

На примере стабилизированного метода би-сопряженных градиентов с классическим алгебраическим многосеточным методом в качестве предобуславливателя получена оценка потенциального ускорения времени решения систем линейных алгебраических уравнений при переносе вычислений с центральных процессоров на графические ускорители. Наблюдаемое ускорение вычислений при использовании формата Adaptive CSR для хранения разреженных матриц на GPU для текущей версии библиотеки XAMG составляет в среднем 5.2 раза.

Литература

1. Saad Y. Iterative methods for sparse linear systems. SIAM, 2nd edition. Philadelphia, PA. 2003. 520 p.
2. Williams S., Waterman A., Patterson D. Roofline: An insightful visual performance model

- for multicore architectures // *Communications of the ACM*. 2009. Vol. 52, no. 4. P. 65–76. DOI: 10.1145/1498765.1498785.
3. Barrett R. et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 1994.
 4. Falgout R.D., Yang U.M. hypre: A Library of High Performance Preconditioners // *Lecture Notes in Computer Science*. 2002. Vol. 2331. P. 632–641. DOI: 10.1007/3-540-47789-6_66.
 5. Balay S. et al. PETSc Web page. URL: <https://petsc.org> (дата обращения: 15.01.2025).
 6. Anderson E., Saad Y. Solving Sparse Triangular Linear Systems on Parallel Computers // *International Journal of High Speed Computing*. 1989. Vol. 1. P. 73–95. DOI: 10.1142/S0129053389000056.
 7. Vassiliadis S., Cotofana S., Stathis P. Block based compression storage expected performance // *Kluwer International Series in Engineering and Computer Science*. 2001. Vol. 657. P. 389–406. DOI: 10.1007/978-1-4615-0849-6_26.
 8. Liu W., Vinter B. CSR5: An Efficient Storage Format for Cross-Platform Sparse Matrix-Vector Multiplication // *Association for Computing Machinery*. 2005. P. 339–350. DOI: 10.1145/2751205.275120.
 9. Kincaid D., Oppe T., Young D. ITPACKV 2D user's guide // *Technical report*. 1989. URL: <https://www.osti.gov/servlets/purl/7093021> (дата обращения: 15.02.2023).
 10. Vázquez F., Ortega G., Fernández J.J., Garzón E.M. Improving the Performance of the Sparse Matrix Vector Product with GPUs // *2010 10th IEEE International Conference on Computer and Information Technology*. 2010. P. 1146–1151. DOI: 10.1109/CIT.2010.208.
 11. Monakov A., Lokhmotov A., Avetisyan A. Automatically Tuning Sparse Matrix-Vector Multiplication for GPU Architectures // *High Performance Embedded Architectures and Compilers. HiPEAC 2010. Lecture Notes in Computer Science*. Vol. 5952. P. 111–125. URL: 10.1007/978-3-642-11515-8_10
 12. Kreutzer M. et al. A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide SIMD units // *SIAM Journal on Scientific Computing*. 2014. Vol. 36, no. 5. P. 401–423. DOI: 10.1137/130930352.
 13. Greathouse J.L., Daga M. Efficient Sparse Matrix-Vector Multiplication on GPUs Using the CSR Storage Format // *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2014. P. 769–780. DOI: 10.1109/SC.2014.68.
 14. Bell N., Garland M. Efficient Sparse Matrix-Vector Multiplication on CUDA // *NVIDIA Corporation, technical report*. 2008.
 15. Krasnopolsky B., Medvedev A. XAMG: A library for solving linear systems with multiple right-hand side vectors // *SoftwareX*. 2021. Vol. 14. Article 100695. DOI: 10.1016/j.softx.2021.100695.
 16. Krasnopolsky B., Medvedev A. XAMG: Source code repository. URL: <https://gitlab.com/xamg/xamg> (дата обращения: 21.01.2025).
 17. Krasnopolsky B., Medvedev A. Acceleration of large scale OpenFOAM simulations on distributed systems with multicore CPUs and GPUs // *Parallel Computing: On the Road to Exascale (conf. proc.)*. Vol. 27 of *Advances in Parallel Computing*. 2016. P. 93–102. DOI: 10.3233/978-1-61499-621-7-93.

18. Davis T.A., Hu Y. The university of Florida sparse matrix collection // ACM Transactions on Mathematical Software. 2011. Vol. 38, no. 1. P. 1–25. DOI: 10.1145/2049662.2049663.
19. NVIDIA cuSPARSE Library. 2025. URL: <https://docs.nvidia.com/cuda/cusparse> (дата обращения: 21.01.2025).
20. Kuprii R.M., Krasnopolsky B.I., Zhukov K.A. Estimating the Effect of Indices Compression in the CSR-like Data Storage Formats for Matrix-Vector Multiplications and Solving Linear Systems // Lobachevskii J Math. 2023. Vol. 44. P. 3100–3111. DOI: 10.1134/S1995080223080346.

Подход к параллельной реализации решения псевдогеометрической версии задачи коммивояжёра геометрическим подходом

Б.Ф. Мельников¹, Бовень Лю¹, М.Э. Абрамян^{1,2}

¹Совместный университет МГУ–ППИ в Шэньчжэне,

²Южный федеральный университет

Объекты исследования статьи таковы. Во-первых – псевдогеометрическая версия задачи коммивояжёра, получающаяся из геометрической версии умножением каждого элемента соответствующей матрицы расстояний на свою случайную величину. Во-вторых – эвристический подход к решению всей задачи, заключающийся в псевдооптимальном расположении точек в единичном квадрате и применении к этому расположению одной из версий т. н. алгоритма «луковой шелухи». В-третьих – варианты вычисления невязки, включая случаи, когда еще не все точки расположены окончательно. В-четвертых – «применение с запаздыванием» окончательного решения о расположении точки: это уменьшает получающуюся невязку. Именно последний вариант удобен при параллельной реализации.

Ключевые слова: задача коммивояжёра, псевдогеометрическая версия, геометрический подход, эвристические алгоритмы.

1. Введение

Представляемая статья продолжает наши работы, связанные с алгоритмами решения псевдогеометрической версии задачи коммивояжёра (ЗКВ). Среди наших последних публикаций, связанных с этой проблемой, упомянем [1–9]. Некоторые из этих публикаций посвящены именно псевдогеометрической версии (одному из основных объектов исследования статьи) – но в той или иной степени с псевдогеометрической версией связаны все эти статьи.

Эту связь можно объяснить следующим образом (подробнее см. в первую очередь статьи [1, 5]). Имеющееся «в Интернете» мнение, что существуют эвристические алгоритмы, с минимальной погрешностью решающие ЗКВ для любой размерности, верно – но верно только для геометрической версии: генерация исходных данных для нее производится случайным расположением точек в единичном квадрате (либо расположением, заданным каким-то иным способом, [7] и др.) – и заполнение матрицы расстояний числами, являющимися обычными евклидовыми расстояниями между нужными точками. Псевдогеометрическая же версия получается из геометрической версии умножением каждого элемента соответствующей матрицы расстояний на свою случайную величину (с. в.); при этом все такие с. в. являются независимыми одинаково распределенными (н. о. р. с. в.) с нормальным законом распределения, математическим ожиданием, равным 1, и некоторой заданной дисперсией (обычно небольшой). В предельных для возможного значения дисперсии случаях псевдогеометрическая версия становится:

- либо геометрической (когда дисперсия равна 0),
- либо случайной (когда дисперсия фактически равна ∞);

вообще, случайной версией обычно называют ситуации, когда заполняются непосредственно элементы матрицы расстояний, каждый элемент является н. о. р. с. в. с равномерным законом распределения.

По-видимому, именно псевдогеометрическая версия ЗКВ наиболее интересна: она является моделью для задач, возникающих в самых разных предметных областях; в предыдущих публикациях мы среди таких предметных областей рассматривали:

- сети связи большой размерности;
- графы для представления химических элементов;
- квантовые графы;
- алгоритмы, вычисляющие расстояния между ДНК-цепочками
- и др.

В настоящей статье мы рассматриваем вариант решения псевдогеометрической версией ЗКВ «подгонкой под ответ»¹. То есть нельзя сказать, что мы здесь предлагаем *окончательный* вариант эвристического алгоритма решения задачи: мы пользуемся решением исходного геометрического варианта ЗКВ, но ведь при решении конкретного варианта проблемы мы его не знаем, нам известна только исходная матрица!

В связи с этим мы применяем т.н. геометрический подход – к матрице, которая, конечно, геометрическую ЗКВ собою не представляет. Для его понимания сначала рассмотрим тривиальный алгоритм восстановления матрицы, относящейся к геометрической версии ЗКВ. Мы в ней располагаем первую точку в начале координат, вторую – на оси (OX) на известном от нее расстоянии, третью – исходя из известных нам сторон образовавшегося треугольника. Для четвертой точки таких треугольников получается 3 (6, 10, 15, 21, ...) – и, соответственно, такое же число *пар* точек. но с помощью несложного вспомогательного алгоритма из каждой пары выбирается приемлемый вариант: 3 точки из получающихся 6, фактически представляющие собой одну и ту же точку. И так далее. После выбора размещения мы применяем известный алгоритм «луковой шелухи».

Для псевдогеометрической ЗКВ мы делаем практически то же самое. Отметим, что максимально получающееся число треугольников квадратично зависит от первоначального числа точек – такое значение приемлемо. Однако, конечно, выбрав очередное число точек (6, 10, 15, 21, ... пар точек, см. выше), мы не получим фактического совпадения между собой половины из них. Но *при малых значениях дисперсии* (примененной ранее при генерации варианта ЗКВ) мы с помощью несложного вспомогательного алгоритма находим точки «этой половины». Далее производится усреднение координат этих точек – и уже к полученному размещению мы применяем алгоритм «луковой шелухи».

В конце введения отметим следующее. Мы уже публиковали некоторые связанные с этой тематикой результаты вычислительных экспериментов, [10]. Однако, во-первых, в той работе использовалось малое число сгенерированных вариантов ЗКВ; в частности, для больших размерностей только 1 вариант. Во-вторых, алгоритмы были реализованы «на черном», т.е. мы не оптимизировали алгоритмы для вспомогательных эвристик. В-третьих, производилось сравнение всех результатов с версией генетического алгоритма – который мы сейчас считаем неудачным; вообще, применение генетических алгоритмов для решения ЗКВ вряд ли имеет смысл. И так далее: мы привели только некоторые улучшения описанных ранее алгоритмов и результатов счета. Еще отметим, что некоторые результаты вычислительных экспериментов были приведены в [5] – но эти результаты можно назвать только предварительными. То есть можно считать, что мы только приступаем к полному исследованию рассматриваемой проблемы.

2. Формализованная постановка задачи

Постановка задачи обычная для задачи коммивояжера: для заданной матрицы расстояний

$$\{a_{i,j} \mid i, j \in \{1, 2, \dots, n\}\}$$

¹Пользуемся «школьной терминологией».

(диагональные элементы a_{ii} обычно не используются, в программных реализациях их удобно полагать равными ∞) требуется найти перестановку $(k_1, k_2, \dots, k_n)$, минимизирующую значение суммы

$$a_{k_1 k_2} + a_{k_2 k_3} + \dots + a_{k_{n-1} k_n} + a_{k_n k_1}; \quad (1)$$

значение k_1 обычно полагается равным 1, что на 1 уменьшает размерность.

Поскольку даже быстрые эвристические алгоритмы (в частности, версии метода ветвей и границ (МВГ), [1–4] и др.) в худшем случае дают экспоненциальное время работы, для псевдогеометрической версии мы применяем другие эвристики – либо дополняющие МВГ, либо, гораздо чаще, с ним не связанные, [5, 8, 9] и др.; конкретные варианты применяемых эвристических алгоритмов рассматриваются в статье далее.

Для оценки качества конкретного эвристического алгоритма, конечно, желательно применять обычные характеристики работы алгоритма, а также их усреднение для различных вариантов входных данных. Однако даже для размерностей порядка 100 время вычисления на обычном «среднем» современном компьютере точного оптимального значения суммы (1) обычно находится в пределах 2–3 часов (см. конкретные значения времени работы наших программ в [1, 3]) – и, повторим, в худшем случае может значительно превосходить такие значения; поэтому подобные варианты оценки качества эвристического алгоритма в данной ситуации могут быть применены только разово, и их вряд ли удастся усреднять даже для не очень большого числа вычислительных экспериментов.

Как уже было отмечено во введении, мы для решения ЗКВ применяем псевдооптимальное размещение точек. При этом некоторые из точек в процессе вычислений размещены только *предварительно* – однако для вычисления требуемой нам невязки мы пользуемся их временными координатами. Эти временные координаты мы получаем не рассматривая все необходимые треугольники – а рассматривая только их последовательность, в которой для временного размещения следующей точки рассматриваются только 2 пары треугольников. Само значение невязки (и временное, и окончательное) вычисляется обычным способом: это

$$\sqrt{\sum_{1 \leq i < j \leq n} (a_{ij} - \overline{a_{ij}})^2},$$

где a_{ij} , как и ранее, суть значения исходной матрицы, а $\overline{a_{ij}}$ – соответствующие значения матрицы получаемой; это значение для наглядности можно разделить на общее количество наддиагональных элементов матрицы – однако при фиксированном значении размерности это необязательно. Еще отметим, что мы применяем и добавочные вспомогательные алгоритмы, также необходимые для вычисления текущего значения невязки; они кратко описаны в следующем разделе.

Итак, в этом разделе кратко описан эвристический подход к решению всей задачи, заключающийся в *псевдооптимальном* расположении точек в единичном квадрате – и применении к этому расположению одной из версий алгоритма «луковой шелухи».

3. Краткое описание алгоритмов

Основная «параллельная» часть разработанной¹ программы – это «применение с запаздыванием» окончательного решения о расположении точки, после предварительного выполнения следующего шага: этот вариант существенно уменьшает получающуюся невязку. Именно последний вариант удобен при параллельной реализации.

Подробнее. В предыдущих публикациях (а также в материале, описанном выше) порядок рассмотрения и размещения точек фактически предполагался заранее заданным; например, их можно рассматривать просто в порядке возрастания номеров, получаемых

¹И разрабатываемой далее.

при задании или случайной генерации исходных данных. Однако нет никакой гарантии, что при этом получается минимально возможная невязка: ведь число возможных порядков рассмотрения равно $(n - 1)!$.

При параллельной реализации мы в качестве очередной точки выбираем любой возможный вариант – размещая каждый из этих вариантов «в своем процессе». Более того, поскольку в наших вычислительных экспериментах число процессоров примерно равно исходной размерности задачи, мы каждый раз последовательно пытаемся провести еще 2–3 шага (точное их число задается параметром алгоритма, оптимизация этого параметра нами еще не рассматривалась) – и выбираем следующую точку, рассматривая полученные в разных процессах значения невязки и выбирая лучшее из этих значений. Понятно, что последовательная реализация подобного алгоритма будет требовать значительного объема памяти, а также очень усложнит работу программиста, требующуюся при возможных модификациях вспомогательных алгоритмов.

Как мы уже отмечали, рассматриваемый здесь алгоритм нельзя назвать окончательным – он является «подгонкой под ответ»: мы пользуемся последовательностью точек, полученной с помощью алгоритма «луковой шелухи», примененного к исходному геометрическому варианту ЗКВ (в реальности мы этот исходный вариант знать не должны). Но, с нашей точки зрения, отношение значений, полученных для таких двух вариантов – то есть значения псевдогеометрической ЗКВ к соответствующему значению геометрической ЗКВ, – представляет большой интерес; более того, разрабатывая новые версии алгоритмов желательно стремиться к минимизации такого отношения.

Эти значения отношения результатов двух алгоритмов мы усредняем для нескольких вычислительных экспериментов. Стоит отметить разные варианты этого усреднения:

1. обычное среднее арифметическое;
2. среднее арифметическое для набора значений, не включающих 20% лучших и 20% худших;
3. усреднение с помощью функции риска, применяющейся для увеличения весов «хороших» результатов;
4. усреднение с помощью функции риска, применяющейся для увеличения весов «плохих» результатов.

(Эти же номера будут использоваться и в дальнейшем. Подробное описание применяемых нами функций риска см. в [11], а также, кратко, в [12]. Некоторые пояснения приведены на следующих двух рисунках.)

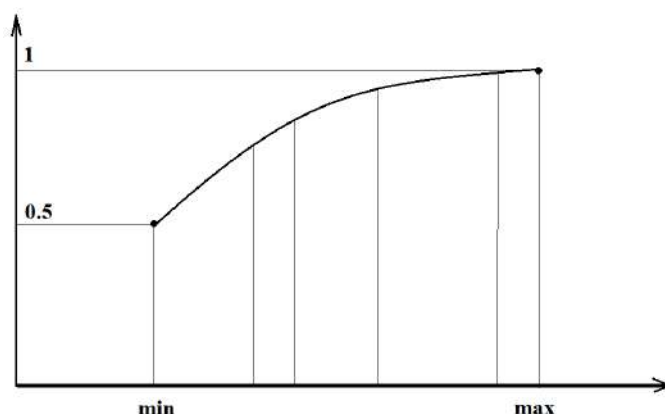


Рис. 1. Пример функции риска; бóльшие значения являются «плохими»; наихудший вариант наиболее важен

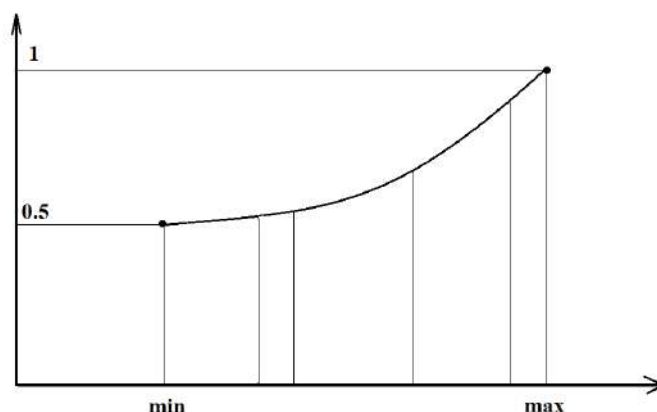


Рис. 2. Пример функции риска; большие значения являются «плохими»; средний вариант наиболее важен

4. Результаты вычислительных экспериментов

Сначала приведем вариант расположения точек, получаемый с помощью алгоритма «луковой шелухи» для геометрического варианта ЗКВ: см. рис. 3. Как и во всех версиях этого алгоритма, мы сначала получаем все возможные контуры, после чего с помощью вспомогательных алгоритмов соединяем контуры соседние.

Несколько менее удачный вариант работы приведен для псевдогеометрического варианта ЗКВ: см. рис. 4: в нем имеется пересечение, которое можно заметить в окрестностях точек с номерами 82 и 86. Однако необходимо отметить, что среди псевдогеометрических вариантов ЗКВ, даже для малых дисперсий, этот рисунок очень удачный: самопересечений очень мало.

Итоговые результаты усреднений приведены в следующей таблице. Как мы уже отмечали, мы четырьмя разными способами усредняем отношения полученных значений – для псевдогеометрического варианта и для геометрического.

Таблица 1. Результаты усреднения отношений значений

Дисперсия	№1	№2	№3	№4
≈ 0.02	2.76	2.75	2.76	2.71
≈ 0.0064	1.76	1.70	1.68	1.64
≈ 0.0004	1.27	1.27	1.27	1.24

Как мы видим, ухудшение результатов в зависимости от увеличения дисперсии есть (что, конечно, ожидалось), а существенного изменения значений в зависимости от способа усреднения не обнаружилось.

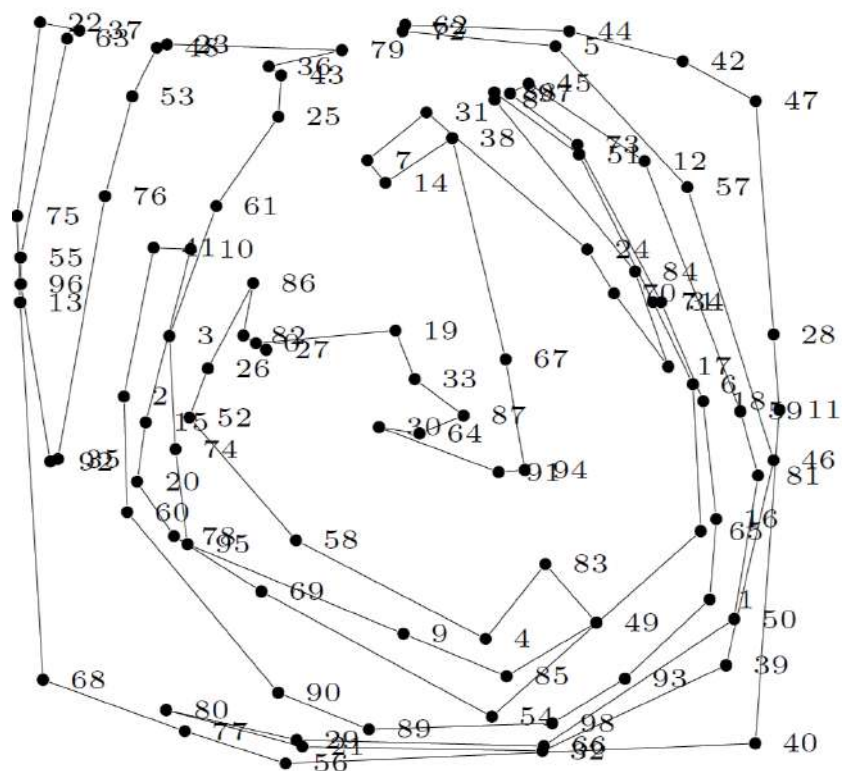


Рис. 3. Итоговое расположение точек для геометрического варианта ЗКВ

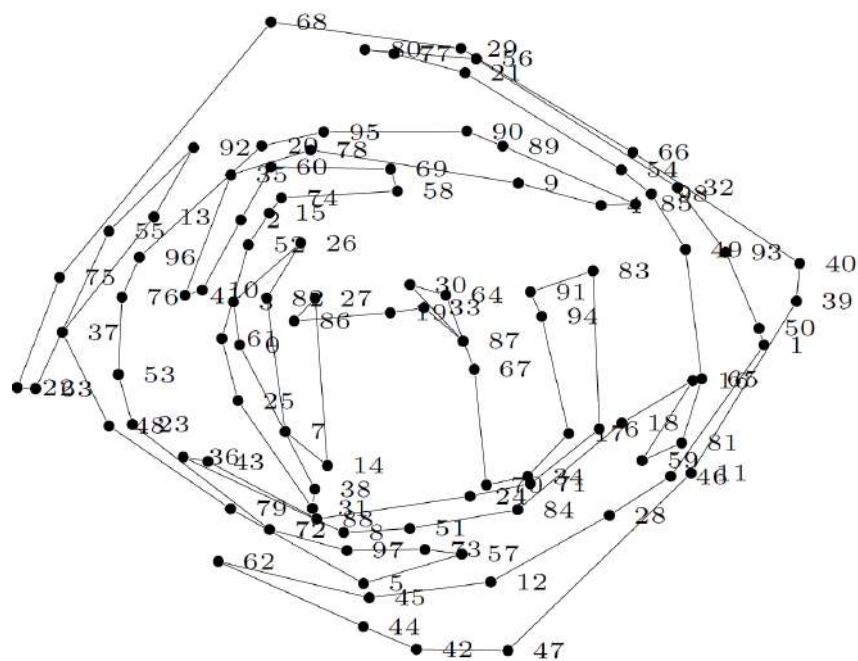


Рис. 4. Итоговое расположение точек для псевдогеометрического варианта ЗКВ

Литература

1. Мельников Б.Ф., Мельникова Е.А. О классической версии метода ветвей и границ // Компьютерные инструменты в образовании. 2021. № 1. С. 21–44. DOI: 10.32603/2071-2340-2021-1-21-45.
2. Мельников Б.Ф. Об объектно-ориентированной реализации метода ветвей и границ для задачи коммивояжёра. Часть I // Современные информационные технологии и ИТ-образование. 2022. Т. 18, № 2. С. 287–299. DOI: 10.25559/SITITO.18.202202.287-299.
3. Мельников Б.Ф. Об объектно-ориентированной реализации метода ветвей и границ для задачи коммивояжёра. Часть II // Современные информационные технологии и ИТ-образование. 2022. Т. 18, № 3. С. 644–654. DOI: 10.25559/SITITO.18.202203.644-654.
4. Melnikov B., Melnikova E. On the classical version of the branch and bound method // Computer Tools in Education. 2022. No. 2. P. 41–58. DOI: 10.32603/2071-2340-2022-2-41-58.
5. Melnikov B., Melnikova E. Versions of the “onion husk” algorithm in the pseudo-geometric traveling salesman problem with small variance // Computer Tools in Education. 2023. No. 4. P. 30–40.
6. Мельников Б.Ф., Терентьева Ю.Ю., Чайковский Д.А. О применении эвристических алгоритмов решения псевдогеометрической версии задачи коммивояжёра для проектирования сетей связи // Информатизация и связь. 2023. № 4. С. 7–16.
7. Melnikov B., Terentyeva Yu., Chaikovskii D. Pseudogeometric version of the traveling salesman problem: application in quantum physics models and a heuristic variant of point placement // Cybernetics and Physics. 2023. Vol. 12, no. 3. P. 194–200. DOI: 10.35470/2226-4116-2023-12-3-194-200.
8. Melnikov B., Melnikova E. Optimization of the solutions of the pseudogeometric version of the traveling salesman problem // Nonlinear Analysis and Extremal Problems (NLA-2024). Proceedings of the 8th International School-Seminar. Irkutsk, 2024. P. 178–180.
9. Мельников Б.Ф., Терентьева Ю.Ю. О применении нескольких предикторов в методе ветвей и границ (на примере случайного варианта задачи коммивояжёра) // International Journal of Open Information Technologies. 2024. Vol. 12, no. 3. P. 1–17.
10. Макаркин С.Б., Мельников Б.Ф. Геометрические методы решения псевдогеометрической версии задачи коммивояжёра // Стохастическая оптимизация в информатике. 2013. Т. 9, № 2. С. 54–72.
11. Melnikov B. Heuristics in programming of nondeterministic games // Programming and Computer Software. 2001. Vol. 27, no. 5. P. 277–288. DOI: 10.1023/A:1012345111076.
12. Melnikov B. New algorithms for restoring DNA matrix and their statistical study // Cybernetics and Physics. 2024. Vol. 13, no. 4. P. 288–295. DOI: 10.35470/2226-4116-2024-13-4-288-295.

Численная оптимизация проверочной матрицы LDPC-кода для применения в протоколе квантового распределения ключей с использованием высокопараллельных вычислений*

В.И. Морозов, В.О. Башара, М.В. Емельяненко

Национальный исследовательский университет «Высшая школа экономики»
(101000, г. Москва, ул. Мясницкая, д. 20)

Исправление ошибок в секретном ключе является обязательным этапом протоколов квантового распределения ключей (КРК). Для его реализации, как правило, используются современные помехоустойчивые коды. Несовершенство аппаратуры, используемой в системах КРК, приводит к появлению битовых ошибок в канале. Более того, для подобных систем характерно несимметричное распределение таких ошибок. Учет такой асимметрии в модели канала не только позволяет повысить эффективность используемого кода, но и оставляет открытым вопрос об оптимальности существующих проверочных матриц для данной задачи. В данной работе проводится оптимизация проверочной матрицы LDPC-кода для заданного квантового канала. Полученные результаты показывают применимость численной оптимизации стандартных проверочных матриц для повышения эффективности КРК.

Ключевые слова: квантовое распределение ключей, помехоустойчивое кодирование, LDPC, оптимизация.

1. Введение

Развитие современных квантовых технологий открывает возможности для практической реализации систем, существование которых несколько десятилетий назад было только теорией. Одним из видов таких систем являются системы квантового распределения ключей (КРК). Протоколы КРК позволяют двум сторонам информационного обмена выработать общую последовательность бит. Благодаря тому, что секретность такой последовательности обеспечивается фундаментальными законами квантовой механики [1], биты могут быть использованы сторонами в качестве секретного ключа для дальнейшей защиты канала коммуникации.

Большинство протоколов квантового распределения ключей содержат схожий набор этапов, которые можно разделить на две основные группы: непосредственная передача квантовых состояний и алгоритмическая постобработка полученной «сырой» последовательности (подробно этапы протоколов КРК будут описаны в разделе 3). Помимо прочих этапов, обработка полученной последовательности бит, как правило, включает исправление битовых ошибок-инверсий, которые всегда имеют место в силу несовершенств практических реализаций протоколов КРК. Для эффективного согласования результирующей ключевой последовательности между сторонами используются методы помехоустойчивого кодирования.

Эффективность многих современных кодов, исправляющих ошибки (например, LDPC-кодов), во многом определяется структурой проверочной матрицы — прямоугольной матрицы над полем $GF(2)$, задающей набор проверок на четность. При этом для различных

* Публикация подготовлена в ходе проведения исследования 25-00-061 «Алгоритмическое обеспечение систем квантового распределения ключей» в рамках Программы «Научный фонд Национального исследовательского университета «Высшая школа экономики» (НИУ ВШЭ)» с использованием суперкомпьютерного комплекса НИУ ВШЭ.

каналов связи наилучшую эффективность могут показывать разные проверочные матрицы. В силу того, что квантовая среда передачи, по которой производится отправка «сырого» ключа, представляет собой канал, принципиально отличный от наиболее распространенных (таких как, например, двоичный симметричный канал или двоичный канал с гауссовским шумом), выработка оптимальной проверочной матрицы для такого канала не является решенной задачей.

Таким образом, целью данного исследования является получение проверочной матрицы LDPC-кода, демонстрирующей высокую эффективность при использовании в протоколах КРК. Для решения подобных задач используются различные подходы. Одним из наиболее эффективных является численная оптимизация, которая и будет использована в данной работе. В ходе такой оптимизации проверочная матрица итерационно подвергается различным преобразованиям, не меняющим ее размерность. Те из преобразований, которые приводят к улучшению кода по заданному критерию качества, остаются в результирующей матрице. Удачное построение такой последовательности преобразований позволяет достигать значительного прироста корректирующей способности кода для заданных параметров канала.

Дальнейшее повествование организовано следующим образом. В разделе 2 приведен краткий обзор научных работ, посвященных построению и оптимизации проверочных матриц помехоустойчивых кодов, а также использованию таких кодов в протоколах КРК. Раздел 3 содержит описание работы протоколов КРК, применяемых в них методов исправления ошибок, а также используемого в данной работе подхода к оптимизации. Раздел 4 посвящен описанию программной реализации применяемых методов численной оптимизации. В разделе 5 представлены результаты вычислительных экспериментов по оптимизации проверочной матрицы. Обсуждение полученных результатов и выводы по проделанной работе приведены в разделе Заключение.

2. Обзор актуальных работ по тематике исследования

С момента появления квантовой криптографии в 1984 году [2] было предложено множество усовершенствований оригинального протокола BB84, а также немалое количество новых. Усовершенствования касались не только подходов к передаче квантовых состояний, но и алгоритмической постобработки ключа, в том числе и исправления ошибок. Первые протоколы, применяемые для восстановления ключевой информации в КРК [3, 4], были основаны на проверках четности и, согласно утверждению авторов [4], были довольно неэффективны.

Результатом последующих исследований в этом направлении стал протокол CASCADE [5], также основанный на проверках четности и двоичном поиске. Этот протокол получил дальнейшее развитие в более поздних работах (например, [6, 7]), однако большую популярность в современных исследованиях получили методы помехоустойчивого кодирования, нашедшие свое применение как в КРК, так и во многих других областях. Такими методами являются полярные и LDPC-коды.

Так, работы [8, 9] посвящены применению однопроходных схем исправления ошибок в КРК. Такие схемы отличаются отсутствием необходимости во взаимодействии с передающей стороной во время исправления ошибок. Это позволяет увеличить скорость генерации ключа за счет уменьшения количества передаваемой информации. Авторы используют полярные и LDPC-коды и достигают лучших результатов, чем другие однопроходные подходы.

Большое количество современных работ [10–14] посвящено «слепому» исправлению ошибок в ключе. Такое название метод получил из-за того, что в нем не производится предварительная оценка ошибки в канале — QBER (quantum bit error rate). Вместо этого авторы предлагают использовать методы «выкалывания» и «укорачивания», чтобы в про-

цессе исправления ошибок итеративно подстроить скорость кода — отношение количества информационных бит к общему количеству бит кодового слова — таким образом, чтобы ее было достаточно для исправления имеющегося уровня ошибок. Большинство подходов, предложенных авторами подобных исследований, используют LDPC-коды и достигают результатов, близких к пределу Шеннона.

Следует отметить, что полярные коды, введенные позднее, чем LDPC, также нашли свое применение в КРК. Так, в работе [15] предлагается использование полярных кодов в КРК. А исследование [16], в свою очередь, посвящено усовершенствованию подхода к кодированию таким образом, что это позволяет снизить необходимую длину блока до 4 раз. Это дает важное преимущество, в виду того, что возможность использования коротких кодов критически важна в задачах квантовой криптографии, т.к. на большой длине линии ключевая информация генерируется медленно.

Если же возвращаться к применению LDPC-кодов в КРК, то, помимо распространенного «слепого» декодирования, некоторые работы [17, 18] также посвящены многомерным схемам КРК, где в качестве носителей информации используются не двоичные кубиты, а d -мерные «кудиты», несущие в себе более одного бита информации. Для исправления ошибок в таких схемах используются недвоичные LDPC-коды. Как утверждают авторы, такие подходы позволяют достигать большей эффективности исправления ошибок, чем классические двоичные.

Таким образом, исходя из содержания рассмотренных работ в области согласования информации в протоколах квантового распределения ключей, можно заключить, что LDPC-коды являются одним из наиболее популярных инструментов для решения данной задачи, т.к. предоставляют высокую эффективность, а также возможность адаптации к особенностям КРК. По этой причине именно такой тип кодов был взят в качестве объекта оптимизации в данной работе. Далее в обзорной главе пойдет речь об исследованиях, посвященных оптимизации LDPC-кодов.

В число базовых работ, направленных на построение и оптимизацию LDPC-кодов, которые нашли свое отражение во многих более новых исследованиях, входят статьи [19, 20].

В [19] описывается алгоритм Progressive Edge Growth (PEG), позволяющий выстраивать проверочную матрицу LDPC-кода псевдослучайным образом с соблюдением заданных ограничений. Основной метрикой качества кода, на которой и основан данный алгоритм, является охват графа Таннера (см., например, [21]) для данного кода — длина наименьшего цикла в таком графе. Данная метрика используется в качестве целевой функции во многих более современных исследованиях, т.к. может быть эффективно вычислена в ходе построения кода и хорошо коррелирует с его результирующей корректирующей способностью.

В свою очередь, авторы [20] описывают еще одну метрику качества кода — Extrinsic Message Degree (EMD), которая показывает степень связности цикла в графе с остальным графом через узлы переменных. Основываясь на этой метрике, авторы представляют подход ACE (Approximated Cycle EMD), позволяющий строить эффективные по EMD матрицы LDPC-кодов. Этот подход, а также его доработки и расширения (например, [22]), используется в качестве базового во многих дальнейших исследованиях.

Базируясь на рассмотренных выше работах, исследователи представляют различные подходы к оптимизации проверочных матриц. Так, ряд исследований [23–25] посвящен использованию генетических алгоритмов для решения данной задачи. В качестве целевой функции в данных работах тем или иным образом используются результаты симуляции исправления ошибок с использованием полученной на том или ином этапе оптимизации проверочной матрицы. В статье [23] авторы ищут точку пересечения графика зависимости выходного уровня ошибок (BER) от уровня шума в канале (SNR) с некоторым заданным значением. Удаленность этой точки от нуля берется в качестве значения целевой функции. В исследовании [24] в таких же целях используется мера способности полученного LDPC-кода исправлять все единичные ошибки в кодовом слове. А авторы [25], в свою очередь,

рассчитывают отношения отказов в декодировании к общему количеству попыток декодирования, произведенному во время симуляции (FER). При этом, в отличие от [23], расчет производится не на некотором диапазоне, а на единственном заранее заданном значении SNR.

Статьи других исследователей посвящены применению иных метаэвристических подходов к оптимизации LDPC-кодов. Так, в работе [26] предлагается использование метода роя частиц [27] для построения квазициклических LDPC-кодов. В качестве целевой функции авторы берут комплексный критерий, включающий проверку отсутствия циклов длины 4 в графе Таннера, а также значение выходного BER при заданном значении SNR. В более поздних работах авторы используют алгоритм имитации отжига [28] и метод дифференциальной эволюции [29] для построения более качественных кодов. Примечательно, что, т.к. последняя из перечисленных статей [29] направлена на оптимизацию энергопотребления декодера, в качестве целевой функции в ней используется не производительность кода, а временная и пространственная сложности декодирования.

Подходы к вычислению целевой функции, отличные от классических, встречаются и в других исследованиях. К примеру, авторы [30] используют комплексный многоступенчатый алгоритм в качестве критерия качества. Его вычисление включает как получение зависимости FER от SNR посредством симуляции, так и классические метрики, такие как ACE и охват графа. В более современной работе [31] используется сверточная нейронная сеть для оценки производительности получаемых в ходе оптимизации проверочных матриц. Авторы предлагают подход, включающий специальную предварительную обработку полученных кодов, а также дообучение сети в процессе оптимизации.

Наконец, ряд работ посвящен совершенствованию классических критериев и подходов к построению проверочных матриц. Так, в одной из работ последних лет [32] предлагается алгоритм GCE — Girth-Cycle-Embedding, позволяющий получать коды с любым заданным значением охвата. Суть алгоритма состоит в том, что узлы графа сначала собираются в циклы заданной длины, и только потом встраиваются в результирующий граф. Другое современное исследование [33] направлено на модификацию алгоритма CC-PEG (Cycle-concentrating progressive edge growth) [34], разработанного специально для эффективного расширения квазициклических кодов. Также, говоря о развитии классических подходов, следует упомянуть более раннюю работу [35], где авторы вводят метрику под названием «профиль охвата», на основе которой впоследствии осуществляется построение квазициклических LDPC-кодов с использованием жадного алгоритма.

Подводя итог, по результатам обзора исследований в данной сфере можно отметить, что, несмотря на популярность оптимизации LDPC-кодов среди исследователей, довольно малое внимание уделяется оптимизации протографов квазициклических кодов. Также зачастую авторы берут коды из современных телекоммуникационных стандартов исключительно для сравнения их по производительности со своими, построенными по заданным критериям, кодами. Что касается целевых функций, в виду ресурсозатратности вычислений, лишь в малой части рассмотренных работ в качестве метрики качества получаемой матрицы используются результаты симуляции декодирования на некотором диапазоне SNR (или входных BER). Таким образом, в данной работе были выбраны следующие подходы к решению задачи оптимизации проверочной матрицы LDPC-кода для использования в задачах квантового распределения ключей.

1. В качестве стартовой точки для оптимизации взята проверочная матрица из стандарта 5G [36].
2. В качестве целевой функции используется значение удаленности точки пересечения графика зависимости выходного FER от входного BER с пороговым значением, полученное в результате симуляции.

3. Теоретическая часть

3.1. Протоколы квантового распределения ключей

После того, как Беннет и Брассард в 1984 году представили первый протокол квантового распределения ключей [2], квантовая криптография ушла далеко вперед. Были разработаны как множество модификаций изначального протокола, так и другие схемы КРК, отличающиеся, например, другой передающей средой, другими носителями информации или использованием других свойств квантовых состояний для кодирования ключевой информации. Однако, несмотря на множество различий в физической реализации, с точки зрения алгоритмической постобработки информации, полученной из квантового канала, большинство протоколов имеют сходства. Во-первых, в большей части таких схем помимо квантового канала используется также классический открытый канал связи (например, Ethernet). Во-вторых, большая часть протоколов КРК включает следующие основные этапы.

1. *Передача квантовой информации.* Алиса (передающая сторона информационного взаимодействия) подготавливает квантовые носители информации и кодирует в каждом из них по одному или более информационных бит, сгенерированных случайно, выбирая базис кодирования также случайным образом. Недоступность базиса на первом этапе не позволяет потенциальному злоумышленнику провести качественное измерение информации, закодированной в квантовых носителях. Подготовленные носители информации отправляются Бобу (принимающей стороне информационного взаимодействия). Последовательность информационных бит, полученная Бобом, называется сырым ключом. Все дальнейшие взаимодействия Алисы и Боба в рамках данной сессии протокола происходят только по открытому каналу.
2. *Просеивание ключа.* В процессе передачи по квантовому каналу большая часть носителей информации утрачивается в силу влияния внешних факторов. Боб определяет, какие именно из переданных Алисой квантовых носителей информации были приняты, а также (возможно, посредством дополнительного взаимодействия с Алисой) принимает решение о том, какие из них были измерены верно (например, в верном базисе). Сформированный список номеров верно принятых информационных единиц принимающая сторона отправляет передающей, после чего обе стороны отбрасывают все непринятые или неверно принятые приемником биты ключа. Полученная последовательность бит называется просеянным ключом.
3. *Оценка уровня ошибки в канале.* Помимо ошибок-стираний, когда передаваемый бит не доходит до получателя, квантовым каналам (в силу неидеальности приемо-передающей аппаратуры) также присуще наличие ошибок-инверсий, когда информационный бит принимается получателем, но при этом меняет свое значение на противоположное. На данном этапе Алиса и Боб производят оценку уровня таких ошибок в канале (QBER). Существует ряд различных подходов к оценке QBER, однако все они, так или иначе, требуют раскрытия части информации о секретном ключе. Раскрытые информационные биты отбрасываются на последнем этапе.
4. *Исправление ошибок.* В силу того, что полученные в результате работы протокола биты предназначены для использования в качестве секретного ключа шифрования, даже при малом уровне QBER принимающей стороне необходимо исправить все ошибки-инверсии. Для этого, как правило, используются методы помехоустойчивого кодирования, позволяющие произвести исправление ошибок с раскрытием как можно меньшего (но не нулевого) количества информации о ключе. Более подробно этот этап будет рассмотрен далее в данной главе.

5. *Усиление секретности.* Чтобы результирующую последовательность можно было использовать в качестве секретного ключа шифрования, стороны проводят процедуру усиления секретности. Она состоит в хешировании полученной информации и отбрасывании того количества информации, которое было раскрыто в ходе предыдущих этапов. Результатом этого этапа является секретный ключ.

3.2. LDPC-кодирование

Как уже было отмечено ранее в данной главе, для исправления ошибок в секретных ключах в КРК используются методы помехоустойчивого кодирования [37]. Многие из таких методов для кодирования информации используют уравнения проверки на четность — линейные уравнения, включающие некоторые биты кодируемой информации, а также результат их сложения по модулю 2. Коэффициенты таких уравнений представляются в двоичном виде и собираются в проверочную матрицу H .

LDPC-коды [38] — семейство кодов, в котором используются проверочные матрицы с малым количеством единичных бит. В классическом кодировании, когда информация не является секретной, из проверочной матрицы H получают порождающую матрицу G . Умножение матрицы G на вектор информационных бит дает кодовое слово $c = mG$ — вектор бит, содержащий информационные биты, а также добавочные биты, содержащие дополнительную информацию, компенсирующую информационные потери при передаче слова по каналу связи. Матрица G подбирается таким образом, чтобы выполнялось $s = cH^T = 0$, где 0 обозначает вектор нулевых бит. Вектор s называют синдромом. В случаях же, когда кодовое слово искажено, такое перемножение с большой вероятностью дает ненулевой синдром, из чего стороны обмена могут сделать заключение о наличии искажений.

Чтобы восстановить искаженное кодовое слово $\hat{c} \neq c$, используются алгоритмы декодирования — итерационные алгоритмы, принимающие на вход оценки вероятностей значений каждого бита кодового слова (как правило, это логарифм отношения вероятности того, что бит равен 0, к вероятности того, что бит равен 1, — LLR — log likelihood ratio, $llr_i = \log(p(c_i = 0)/p(c_i = 1))$), а также синдром, вычисленный для этого слова. По окончании работы, каждый из таких алгоритмов либо сообщает об успехе декодирования и возвращает битовый вектор $\tilde{c}$, такой, что $\tilde{c}H^T = 0$, либо сообщает об отказе в декодировании. Отношение количества отказов декодера к общему числу попыток декодирования различных искаженных кодовых слов называется FER (frame error rate) и является метрикой качества кода.

Таким образом, принцип работы классического LDPC-кодирования можно схематично представить так, как показано на рис. 1. Для восстановления исходного информационного вектора в классическом подходе используются такие декодеры как Sum-Product [39], Min-Sum [40, 41], а также различные их модификации.

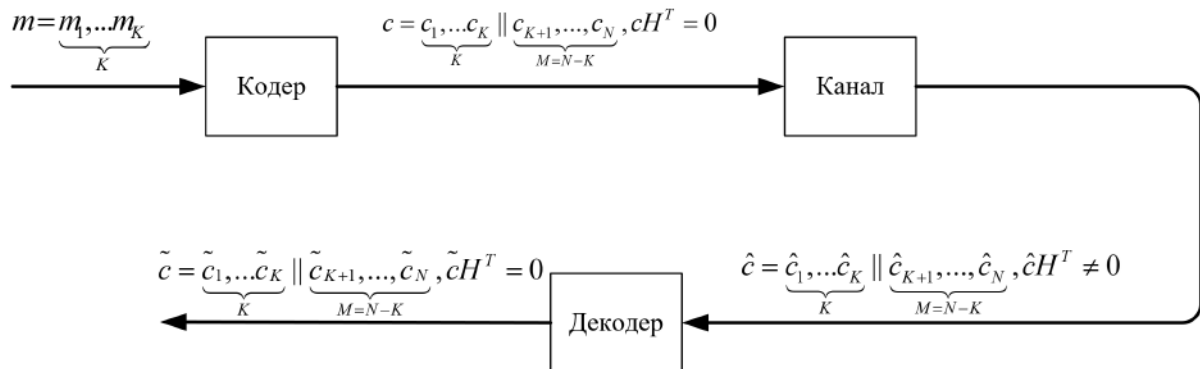


Рис. 1. Классическое помехоустойчивое кодирование

Однако классическая схема не подходит для квантового распределения ключей, т.к. требует передачи информационного вектора вместе с добавочными битами, что, в случае наличия злоумышленника, приведет к полному раскрытию секретного ключа. По этой причине в КРК используется альтернативный подход к кодированию [42]. В таком подходе Алиса передает Бобу вектор бит ключа m (по квантовому каналу), а также «синдром» $s = mH^T$. Для декодирования в таком случае используются модифицированные [43] версии классических декодеров, инвертирующие знак LLR на одном из этапов вычисления. Схема такого подхода к исправлению ошибок представлена на рис. 2.

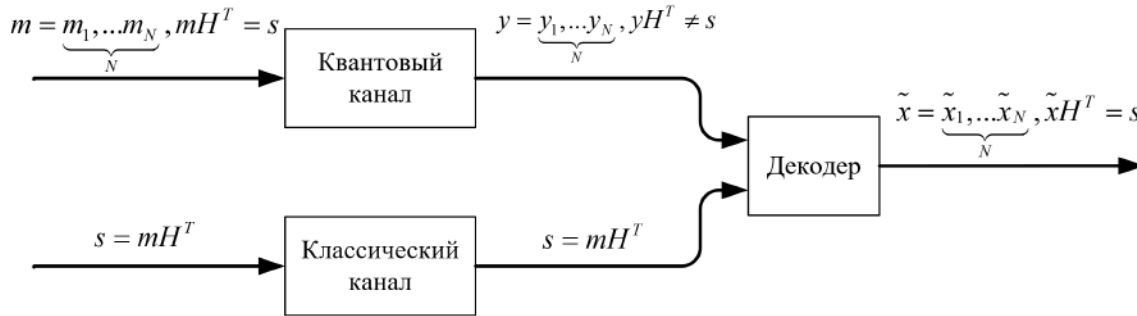


Рис. 2. Помехоустойчивое кодирование в КРК

3.3. Оптимизация проверочной матрицы — жадный алгоритм

Т.к. качество работы различных декодеров LDPC-кодов изменяется в зависимости от выбранной проверочной матрицы [21], можно утверждать, что эффективность того или иного LDPC-кода определяется его проверочной матрицей. Таким образом, в целях повышения эффективности исправления ошибок в секретном ключе можно рассматривать проверочную матрицу как объект оптимизации.

В современных телекоммуникационных стандартах (например, [36]) используются квазициклические коды. Проверочные матрицы таких кодов получаются из некоторой базовой матрицы BG , рекомендуемые варианты которых представлены в стандарте. Переход от базовой матрицы BG к проверочной матрице H производится заменой каждого единичного элемента в BG на квадратную единичную матрицу размера $Z_c \times Z_c$ с последующим циклическим сдвигом всех ее строк вправо на заданное количество позиций. Коэффициент расширения Z_c и длины циклических сдвигов для каждого из бит BG приводятся в стандарте.

Таким образом, несмотря на то, что большинство распространенных подходов к оптимизации LDPC-кодов направлены на построение или преобразование непосредственно самой матрицы H , в данной работе предлагается оптимизация базовой матрицы. Такой способ позволяет сократить количество итераций оптимизации за счет малого размера базовой матрицы, а также провести дополнительную оптимизацию сдвигов при необходимости. Значение целевой функции в данном случае вычисляется по следующему алгоритму.

На вход алгоритм принимает базовую матрицу BG , коэффициент расширения Z_c и количество итераций S , от которого зависит точность применяемого метода Монте-Карло.

1. Установить значение $i = 0$.
2. Расширить базовую матрицу BG на коэффициент Z_c , не производя циклические сдвиги единичных подматриц. Результат взять как матрицу H' .
3. Произвести циклический сдвиг тех из единичных подматриц размера $Z_c \times Z_c$, для которых размер сдвига не описан в стандарте, на случайное количество позиций. Результат взять как матрицу H_i .

4. Получить оценку качества матрицы $quality(H_i)$ (алгоритм будет описан далее).
5. Увеличить i на 1.
6. Если $i > S$, вернуть среднее значение метрики оценки качества по всем матрицам H_i , $i = 0, S - 1$, иначе перейти к шагу 2.

Функция $quality$ вычисляет оценку качества проверочной матрицы посредством симуляции по методу Монте-Карло. Функция принимает на вход следующие параметры.

- Двоичная матрица H размером $M \times N$.
- Величины симулируемой ошибки в канале e_{start} , e_{stop} и e_{step} .
- Величины, описывающие асимметричное распределение ошибок в канале E_i , каждая из которых соответствует одному из наблюдаемых в канале уровней ошибок.
- Величины, отвечающие за количество статистических итераций, определяющие точность метода Монте-Карло N_{total} , $N_{failures}$, I .

Оценка качества проверочной матрицы представляет собой вектор вещественных чисел, представляющих значения FER , соответствующие значениям $QBER$ в диапазоне от e_{start} до e_{stop} с шагом e_{step} . Вычисление этих значений производится по следующему алгоритму.

1. Установить $fers = ()$, где $()$ обозначает пустой вектор вещественных чисел.
2. Установить значение текущей ошибки $e = e_{start}$.
3. Установить счетчик $i = 0$.
4. Установить $fer_{sum} = 0$.
5. Установить счетчики $t = 0$, $f = 0$.
6. Сгенерировать битовый вектор m , $|m| = N$, где $||$ — оператор получения длины вектора.
7. Рассчитать синдром $s = mH^T$.
8. Сгенерировать искаженный битовый вектор y , $|y| = N$, путем инверсии бит вектора m с вероятностью $e + E_k$, где $k \in \{1, 2, 3, 4\}$ выбирается равновероятно для каждого бита сообщения m .
9. Передать векторы s и y на вход декодеру LDPC-кода.
10. Если декодирование прошло успешно, перейти к шагу 12, иначе перейти к шагу 11.
11. Увеличить счетчик отказов f на 1.
12. Увеличить счетчик попыток декодирования t на 1.
13. Если $t < N_{total}$ и $f < N_{failures}$, перейти к шагу 6, иначе перейти к шагу 14.
14. Увеличить значение fer_{sum} на f/t .
15. Увеличить i на 1. Если $i < I$, перейти к шагу 5, иначе перейти к шагу 16.
16. Добавить значение fer_{sum}/I в вектор $fers$.
17. Увеличить e на e_{step} . Если $e \leq e_{stop}$, перейти к шагу 3, иначе вернуть вектор $fers$.

Под средним качеством матриц $H_0, H_1, \dots, H_S$, качество каждой из которых определяется векторами $q_0, q_1, \dots, q_S$ понимается вектор

$$q_{av} = (\sum_{i=0}^S q_i^0/S, \sum_{i=0}^S q_i^1/S, \dots, \sum_{i=0}^S q_i^l/S),$$

где q_i^j — j -ый элемент i -ого вектора качества, а l — длина векторов качества ($l = \frac{e_{stop} - e_{start}}{e_{step}}$).

Алгоритм оптимизации принимает на вход базовую матрицу BG , коэффициент расширения Z_c , а также количество итераций оптимизации I_{opt} . Сам алгоритм представляет следующую последовательность шагов.

1. Расширить базовую матрицу BG размера $M_{BG} \times N_{BG}$ на коэффициент Z_c , произведя циклические сдвиги единичных подматриц согласно стандарту 5G [36]. Результат обозначить как матрицу H_{init} .
2. Получить вектор качества $q_{curr} = quality(H_{init})$.
3. Установить значение счетчика $i = 0$.
4. Сгенерировать два случайных числа $r, c \in \mathbb{N} : 0 < r < M_{BG}, 0 < c < N_{shift}$, где N_{shift} — индекс самого правого столбца базовой матрицы, для битов которого, согласно стандарту, производится ненулевой циклический сдвиг после развертывания в единичную подматрицу.
5. Инвертировать бит базовой матрицы BG в строке с номером r и столбце с номером c .
6. Рассчитать значение целевой функции для BG . Полученный вектор качества обозначить как q_i .
7. Если $q_i > q_{curr}$ (правила сравнения векторов качества будут описаны далее), установить $q_{curr} = q_i$ и перейти к шагу 9, иначе перейти к шагу 8.
8. Отменить изменения матрицы BG , сделанные на текущей итерации.
9. Увеличить счетчик i на 1. Если $i > I_{opt}$, вернуть матрицу BG , иначе перейти к шагу 4.

Неотъемлемым этапом многих алгоритмов оптимизации является сравнение значений целевых функций. Т.к. в нашем случае значение целевой функции представляет собой вектор, необходимо дополнительно привести правила сравнения таких значений. Для этого прежде всего фиксируется некоторое малое пороговое значение $fer_{threshold}$ представляющее собой целевое значение для всех элементов вектора качества. Далее сравнение двух векторов качества q_1 и q_2 длины l производится по следующим правилам.

1. Если $\forall 0 \leq i < l : q_1^i < fer_{threshold}, q_2^i < fer_{threshold}$, (т.е. если оба вектора качества изобразить на координатной плоскости, взяв их элементы как значения по оси ординат, а значения по оси абсцисс установить в соответствующие значения QBER, то они будут лежать под прямой, параллельной оси абсцисс, проведенной на высоте $fer_{threshold}$), то качество вектора q_i рассчитывается как $\sum_{j=0}^{l-1} fer_{threshold} - q_i^j$.
2. Если же условие $\forall 0 \leq i < l : q^i < fer_{threshold}$ выполняется только для одного из векторов качества (т.е. только один вектор лежит под линией $fer_{threshold}$), то этот вектор считается более качественным.
3. Если же ни один из векторов не лежит под линией $fer_{threshold}$, то более качественным признается тот, который пересекает линию $fer_{threshold}$ правее по оси абсцисс.

Выполнение ситуаций 2 и 3 продемонстрировано на рис. 3. Графики 1 и 2 демонстрируют ситуацию 3, когда ни один из векторов качества не лежит полностью под линией $fer_{threshold}$. В данном случае качество вектора на графике 2 выше, т.к. его значения пересекают пороговую линию правее по оси абсцисс. График 3 демонстрирует ситуацию 2, когда один из векторов полностью расположен под линией $fer_{threshold}$. Согласно пункту 2 критериев сравнения, вектор качества, изображенный на графике 3, демонстрирует лучшее качество, чем векторы на графиках 1 и 2.

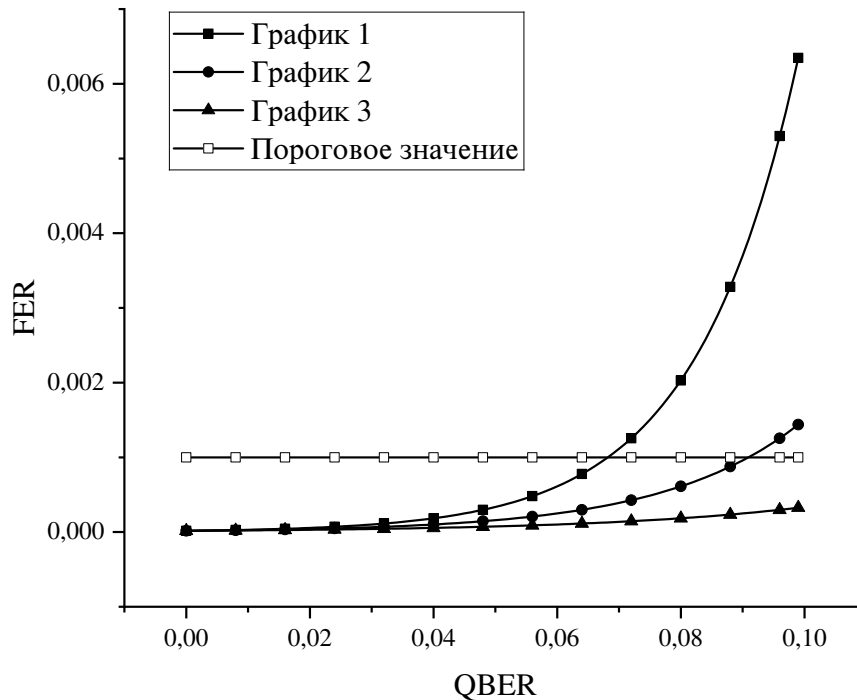


Рис. 3. Пример векторов качества

4. Программная реализация

Для исполнения алгоритмов, описанных в разделе 3, была разработана их программная реализация. С целью ускорения вычислений параллелизация производилась с использованием интерфейса MPI (Message Passing Interface), что позволило наиболее эффективно задействовать вычислительные узлы суперкомпьютерного комплекса НИУ ВШЭ [44]. При каждом запуске из всех параллельно работающих процессов выделялся один процесс-менеджер, остальные процессы назначались исполнителями. В ходе работы алгоритм оптимизации выполнялся процессом-менеджером. Однако расчет значений целевой функции делегировался процессам-исполнителям. Значения целевой функции для матриц с различными случайными сдвигами единичных подматриц независимы друг от друга. Благодаря этому, их вычисление производилось параллельно несколькими процессами-исполнителями, что привело к значительному приросту производительности оптимизации по сравнению с последовательным вариантом. Процессы-исполнители, в свою очередь, использовали многопоточность для вычисления качества проверочной матрицы на конкретных значениях QBER. Таким образом, ролевая структура MPI-процессов в программном исполнении оптимизатора была организована, как показано на рис. 4. N на рисунке означает количество одновременно запускаемых процессов-исполнителей. Содержимое сообщений «Задача» и «Результат» описано далее в данной главе.

Программная реализация оптимизатора исполнена на языке C++ с использованием следующего набора библиотек.

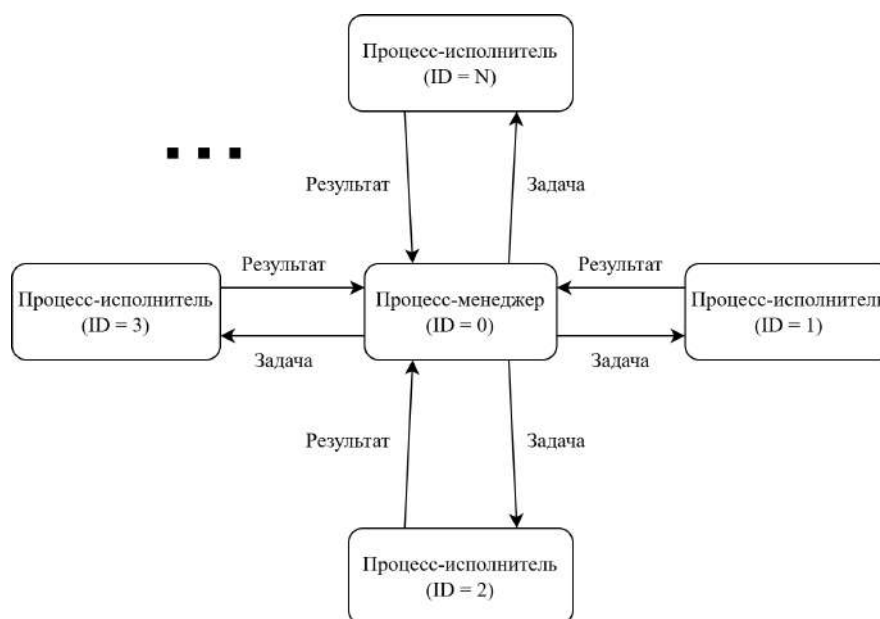


Рис. 4. Ролевая структура MPI-процессов

- OpenMPI с доступом через интерфейс Boost::MPI — для управления параллельными вычислениями и передачей сообщений между процессами по протоколу MPI.
- Eigen — для работы с матрицами, векторами и другими средствами линейной алгебры.
- Taskflow — для эффективной организации параллельных вычислений в пределах одного процесса.

Диаграмма классов представлена на рис. 5. Подробности, связанные с синхронизацией потоков, опущены для упрощения. Как видно из диаграммы, в программе выделены 4 основные сущности.

1. Класс Manager представляет менеджер задач, который получает задачи на вычисление целевой функции с теми или иными параметрами, распределяет их по процессам-исполнителям и собирает результаты.
2. Класс Worker представляет исполнителя, который получает параметры вычисления целевой функции, производит расчет и отправляет результат менеджеру.
3. Класс Task представляет входные параметры целевой функции (сообщение «Задача» на рис. 4).
4. Класс Result представляет результат вычисления целевой функции — вектор качества матрицы (сообщение «Результат» на рис. 4).

Сериализация объектов классов Task и Result для передачи между процессами производится средствами библиотеки Boost. При этом для экономии сетевых ресурсов проверочная матрица представляется в виде вектора позиций единичных вхождений, что, благодаря малому количеству единиц в ней, позволяет значительно сэкономить объем пересылаемого трафика. Более того, разреженность матрицы позволила отводить 1 байт информации для хранения каждого ее бита, а также осуществить совместное использование потоками элементов ее компактного представления с целью локализации данных на процессах-исполнителях.

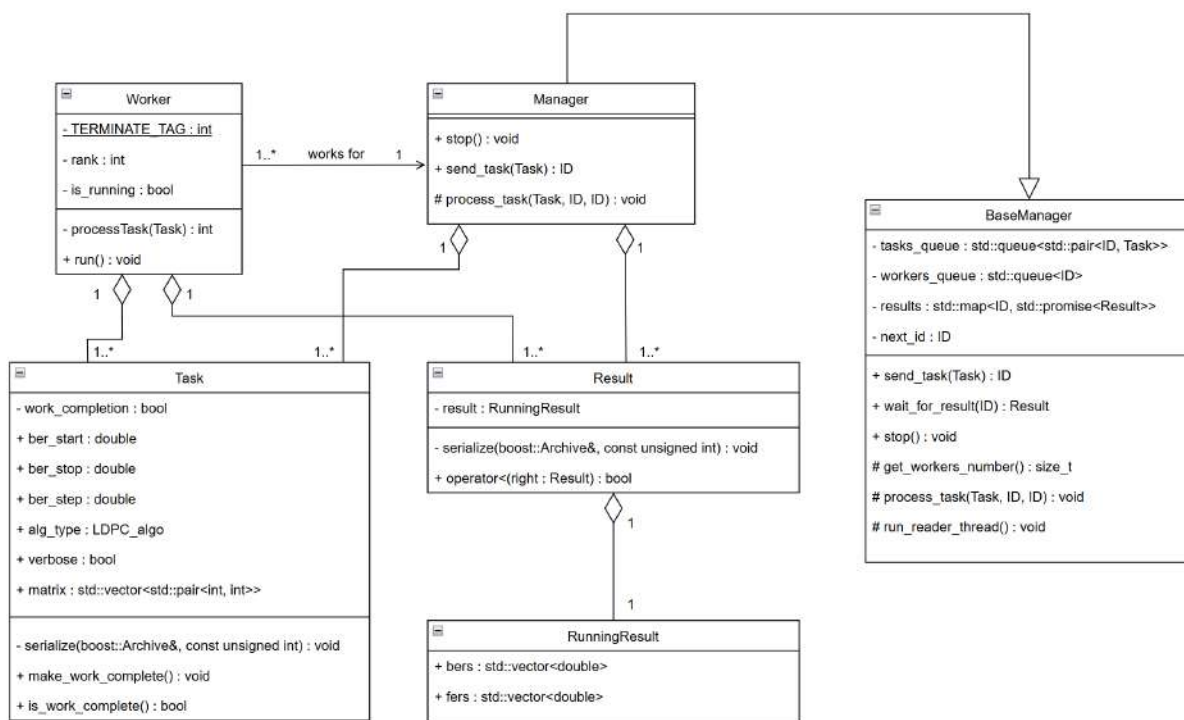


Рис. 5. Диаграмма классов

Оператор сравнения класса Result позволяет использовать обычную синтаксическую конструкцию для определения лучшего (по описанным выше критериям) из двух векторов качества проверочной матрицы.

Этап вычисления целевой функции для набора матриц, отличающихся только сдвигами единичных подматриц в их составе, происходит по следующему простому алгоритму.

1. Процессы-исполнители запускаются при старте программы и вызывают метод run класса Worker, который ожидает поступления задач и приступает к вычислению результатов.
2. Процесс-менеджер осуществляет сдвиги единичных подматриц и передает задачи вычисления целевой функции для них через метод send_task класса Manager.
3. Процесс-менеджер ожидает результатов вычислений через метод wait_for_result класса Manager.
4. Процесс-менеджер рассчитывает среднее значение метрики оценки качества.

Объектно-ориентированная реализация позволяет с незначительными доработками использовать представленное ПО для организации произвольных параллельных вычислений с использованием как MPI, так и других библиотек. Наследуя основные классы программы, можно контролировать выполнение вычислений на необходимом уровне от изменения целевой функции и ее параметров до смены механизма многозадачности.

5. Эксперимент

Оптимизация проверочной матрицы была проведена согласно алгоритму, описанному в подразделе 3.3. Для этого были установлены следующие значения для параметров алгоритмов.

1. В качестве базовой матрицы BG была выбрана матрица «Base graph 1» из стандарта 5G [36]. При этом она была усечена до $M_{BG} = 22$ строк и $N_{BG} = 44$ столбцов для получения скорости кода $R = 1 - M/N = 1/2$.
2. Коэффициент расширения Z_c был взят равным 4, что в итоге дает длину кодируемого блока $N = 44 \times 4 = 176$ бит. Такая длина была выбрана по двум причинам. Во-первых, выбранное значение позволило уменьшить сложность декодирования и, как следствие, выполнить больше итераций оптимизации. Во-вторых, что более важно, в системах квантового распределения ключей при большой длине линии скорость генерации просеянного ключа достаточно мала, из-за чего исправление ошибок на малом количестве бит является типичной задачей.
3. Количество сдвигов S было выбрано равным 10.
4. Величины e_{start} , e_{stop} и e_{step} были выбраны равными 0, 0.03 и 0.001 соответственно, что дает длину вектора качества $l = 30$.
5. Величины ошибок в канале E_i были выбраны следующими: $E_1 = 0.005$, $E_2 = 0.01$, $E_3 = 0.2$, $E_4 = 0.4$, что дает средний QBER 0.01875. Такое количество уровней ошибок и значение среднего QBER соответствуют, например, величинам, наблюдаемым при работе протокола КРК с фазово-временным кодированием [37].
6. Значения величин, отвечающих за точность расчетов методом Монте-Карло были выбраны следующими: $N_{total} = 10000$, $N_{failures} = 50$, $I = 30$.
7. Количество итераций оптимизации I_{opt} было выбрано равным 100.
8. Пороговое значение для сравнения векторов качества $fer_{threshold}$ было установлено равным 0.001.

В результате проведенной оптимизации значения целевой функции изменились, как показано на рисунке 6. На оси абсцисс в приведенном графике указано значение среднего QBER в декодируемом блоке, рассчитанное по формуле $(\sum_{i=1}^4 E_i)/4 + e$, где e — текущее значение добавляемой ошибки на каждой итерации вычисления целевой функции. На оси ординат показан средний уровень FER, полученный при декодировании входных векторов с заданным уровнем ошибки.

Как видно на графике, в результате оптимизации было достигнуто заметное улучшение FER на всем диапазоне вычисления целевой функции, что свидетельствует о большей эффективности применения оптимизированной матрицы для исправления ошибок в рассмотренном диапазоне QBER.

Чтобы продемонстрировать ход улучшения качества проверочной матрицы в процессе оптимизации, была использована метрика эффективности исправления ошибок в КРК, рассчитываемая по следующей формуле.

$$f_{EC} = \frac{M}{Nh(e)},$$

где M — длина синдрома, равная количеству строк в проверочной матрице, N — длина кодируемого блока, равная количеству столбцов в проверочной матрице, e — QBER, исправляемый кодом с заданной вероятностью (в нашем случае это $fer_{threshold} = 0.001$), а h — функция Шенноновской двоичной энтропии.

$$h(x) = -x \log_2(x) - (1-x) \log_2(1-x)$$

Данная метрика показывает отношение количества информации, фактически раскрываемого при исправлении ошибок, к теоретическому минимуму количества информации, которое

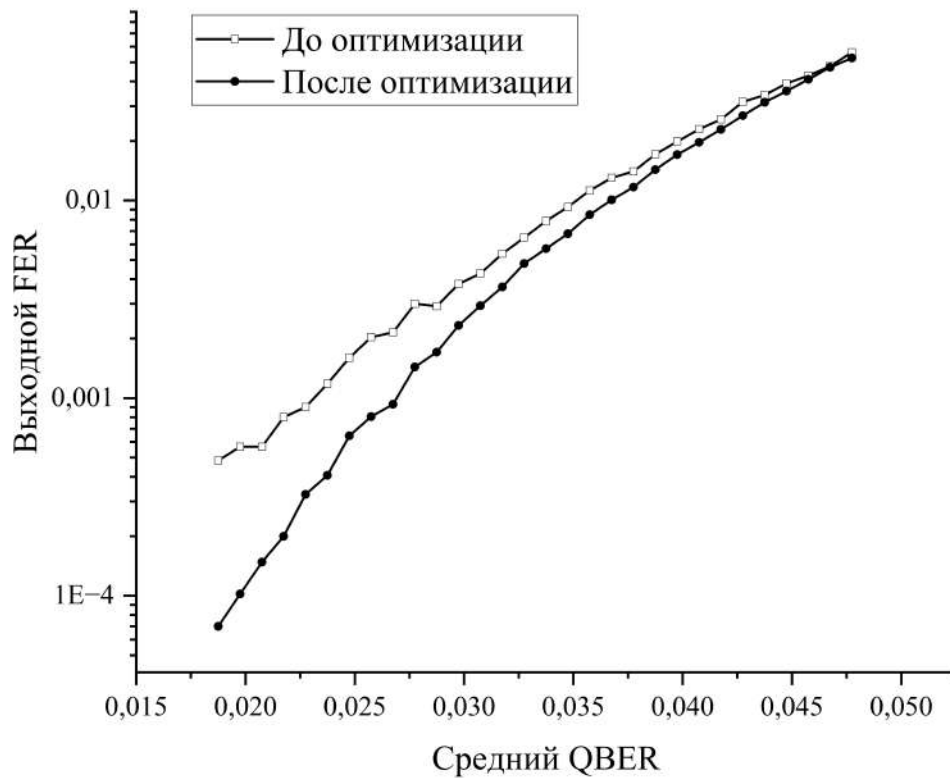


Рис. 6. Значения целевой функции до и после оптимизации

требуется раскрыть для исправления заданного уровня ошибок. Очевидно, $f_{EC} \geq 1$, где равенство достижимо лишь теоретически. Следовательно, чем меньше достигнутое значение f_{EC} , тем больше эффективность кода.

На рис. 7 представлена зависимость значения f_{EC} от номера итерации оптимизатора. При этом крайнее левое значение соответствует эффективности исправления ошибок на стартовой точке до оптимизации. Как видно, в ходе 100 итераций оптимизации f_{EC} было уменьшено примерно на 0.3.

Таким образом, по результатам применения представленного алгоритма оптимизации можно судить о его применимости для повышения корректирующей способности LDPC-кода в протоколах квантового распределения ключей. Также следует отметить, что наибольшие улучшения были достигнуты на первых 50 итерациях оптимизации, что свидетельствует о возможности остановки алгоритма или смены оптимизирующего преобразования после данного количества итераций.

6. Заключение

Таким образом, в ходе проведенного исследования был предложен алгоритм оптимизации проверочной матрицы LDPC-кода для использования в системах квантового распределения ключей. Согласно результатам вычислительных экспериментов, предложенный подход позволил улучшить эффективность декодирования на 0.3, а также достигнуть видимого улучшения на графике зависимости FER от QBER.

Более того, для реализации алгоритма было разработано ПО, модульная структура ко-

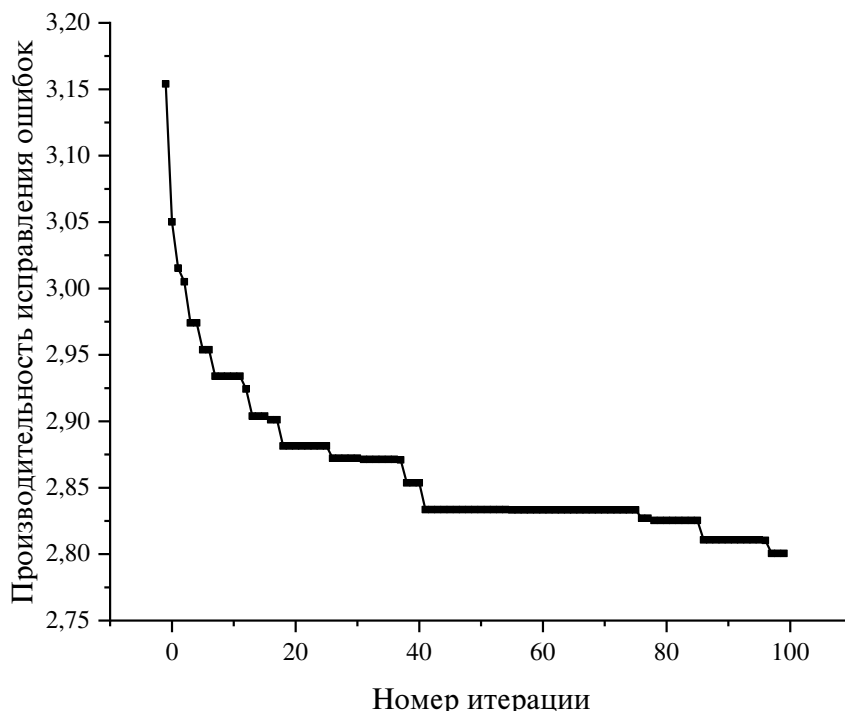


Рис. 7. Улучшение производительности декодирования в ходе оптимизации

торого предоставляет возможность реализации широкого круга параллельных вычислений.

Подводя итог по результатам проделанной работы, можно выделить следующие потенциальные направления дальнейших исследований.

1. Доработка примененного алгоритма оптимизации с использованием нескольких эпох, на каждой из которых инвертируется различное количество бит базовой матрицы.
2. Применение других подходов к оптимизации, к примеру, метаэвристических алгоритмов.
3. Применение более сложного подхода к вычислению целевой функции с использованием дополнительной оптимизации сдвигов единичных подматриц вместо их случайного выбора.

Литература

1. Wootters W.K., Zurek W.H. A single quantum cannot be cloned // Nature. 1982. Oct. Vol. 299, no. 5886. P. 802–803. DOI: 10.1038/299802a0.
2. Bennett C.H., Brassard G. Quantum cryptography: Public key distribution and coin tossing // Theoretical Computer Science. 2014. Vol. 560. P. 7–11. Theoretical Aspects of Quantum Cryptography – celebrating 30 years of BB84. DOI: 10.1016/j.tcs.2014.05.025.
3. Bennett C.H., Bessette F., Brassard G. et al. Experimental Quantum Cryptography // J. Cryptol. 1992. Vol. 5, no. 1. P. 3–28. DOI: 10.1007/BF00191318.
4. Bechmann-Pasquinucci H., Gisin N. Incoherent and coherent eavesdropping in the six-state protocol of quantum cryptography // Physical Review A. 1999. June. Vol. 59, no. 6. P. 4238–4248. DOI: 10.1103/physreva.59.4238.

5. Brassard G., Salvail L. Secret-Key Reconciliation by Public Discussion // Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. Vol. 765 / ed. by T. Hellesest. Springer, 1993. P. 410–423. Lecture Notes in Computer Science. DOI: 10.1007/3-540-48285-735.
6. Toyran M. More efficient implementations of CASCADE information reconciliation protocol // 24th Signal Processing and Communication Application Conference, SIU 2016, Zonguldak, Turkey, May 16-19, 2016. IEEE, 2016. P. 161–164. DOI: 10.1109/SIU.2016.7495702.
7. Mao H.-K., Li Q., Hao P.-L. et al. RETRACTED ARTICLE: High performance reconciliation for practical quantum key distribution systems // Optical and Quantum Electronics. 2022. Feb. Vol. 54, no. 3. DOI: 10.1007/s11082-021-03489-4.
8. Yang L., Dong H., Li Z. One-way information reconciliation schemes of quantum key distribution // Cybersecur. 2019. Vol. 2, no. 1. P. 16. DOI: 10.1186/S42400-019-0033-Z.
9. Tomamichel M., Martínez-Mateo J., Pacher C., Elkouss D. Fundamental finite key limits for one-way information reconciliation in quantum key distribution // Quantum Inf. Process. 2017. Vol. 16, no. 11. P. 280. DOI: 10.1007/S11128-017-1709-5.
10. Liu Z., Wu Z., Huang A. Blind information reconciliation with variable step sizes for quantum key distribution // Scientific Reports. 2020. Jan. Vol. 10, no. 1. DOI: 10.1038/s41598-019-56637-y.
11. Limei G., Qi R., Di J., Duan H. QKD Iterative Information Reconciliation Based on LDPC Codes // International Journal of Theoretical Physics. 2020. Mar. Vol. 59, no. 6. P. 1717–1729. DOI: 10.1007/s10773-020-04438-9.
12. Kiktenko E.O., Malyshev A.O., Fedorov A.K. Blind Information Reconciliation With Polar Codes for Quantum Key Distribution // IEEE Commun. Lett. 2021. Vol. 25, no. 1. P. 79–83. DOI: 10.1109/LCOMM.2020.3021142.
13. Zhang J., Jin H., Xu H., Feng J. A Rate Compatible LDPC Scheme in the Quantum Key Distribution System // 2023 3rd International Conference on Information Communication and Software Engineering (ICICSE). IEEE, 04/2023. P. 51–55. DOI: 10.1109/icicse58435.2023.10211941.
14. Borisov N., Petrov I., Tayduganov A. Asymmetric Adaptive LDPC-Based Information Reconciliation for Industrial Quantum Key Distribution // Entropy. 2023. Vol. 25, no. 1. P. 31. DOI: 10.3390/E25010031.
15. Jouguet P., Kunz-Jacques S. High Performance Error Correction for Quantum Key Distribution using Polar Codes // Quantum Info. Comput. 2014. Vol. 14, no. 3–4. P. 329–338. DOI: 10.5555/2600508.2600516.
16. Zhou H., Tang B.-Y., Chen H. et al. Appending Information Reconciliation for Quantum Key Distribution // Phys. Rev. Applied. 2022. Vol. 18, no. 4. DOI: 10.1103/PhysRevApplied.18.044022.
17. Mueller R., Ribezzo D., Zahidy M. et al. Efficient Information Reconciliation for HighDimensional Quantum Key Distribution // Quantum Information Processing. 2024. Vol. 23. Article 195. DOI: 10.1007/s11128-024-04395-w.

18. Müller R., Bacco D., Oxenløwe L.K., Forchhammer S. Information Reconciliation for HighDimensional Quantum Key Distribution using Nonbinary LDPC codes // 12th International Symposium on Topics in Coding, ISTC 2023, Brest, France, September 4-8, 2023. IEEE, 2023. P. 1–5. DOI: 10.1109/ISTC57237.2023.10273570.
19. Hu X., Eleftheriou E., Arnold D. Regular and irregular progressive edge-growth tanner graphs // IEEE Trans. Inf. Theory. 2005. Vol. 51, no. 1. P. 386–398. DOI: 10.1109/TIT.2004.839541.
20. Tian T., Jones C.R., Villasenor J.D., Wesel R.D. Selective avoidance of cycles in irregular LDPC code construction // IEEE Trans. Commun. 2004. Vol. 52, no. 8. P. 1242–1247. DOI: 10.1109/TCOMM.2004.833048.
21. Johnson S. Iterative Error Correction: Turbo, Low-Density Parity-Check and RepeatAccumulate Codes. Cambridge University Press, 2010. Iterative error correction: turbo, low-density parity-check and repeat-accumulate codes. URL: <https://books.google.ru/books?id=Ni71AAAAMAAJ>.
22. Vukobratovic D., Djurendic A., Senk V. ACE Spectrum of LDPC Codes and Generalized ACE Design // Proceedings of IEEE International Conference on Communications, ICC 2007, Glasgow, Scotland, UK, 24-28 June 2007. IEEE, 2007. P. 665–670. DOI: 10.1109/ICC.2007.114.
23. Broulim J., Georgiev V., Moldaschl J., Palocko L. LDPC code optimization based on Tanner graph mutations // 2013 21st Telecommunications Forum Telfor (TELFOR). IEEE, 11/2013. P. 389–392. DOI: 10.1109/telfor.2013.6716251.
24. Broulim J., Davarzani S., Georgiev V., Zich J. Genetic optimization of a short block length LDPC code accelerated by distributed algorithms // 2016 24th Telecommunications Forum(TELFOR). IEEE, 11/2016. P. 1–4. DOI: 10.1109/telfor.2016.7818770.
25. Elkelesh A., Ebada M., Cammerer S. et al. Decoder-in-the-Loop: Genetic OptimizationBased LDPC Code Design // IEEE Access. 2019. Vol. 7. P. 141161–141170. DOI: 10.1109/ACCESS.2019.2942999.
26. Yang Y., Xiao Y. Construction of QC-LDPC codes based on PSO algorithm // 4th IET International Conference on Wireless, Mobile amp; Multimedia Networks (ICWMMN 2011). IEEE, 2011. P. 118–122. DOI: 10.1049/cp.2011.0971.
27. Kennedy J., Eberhart R. Particle swarm optimization // Proceedings of International Conference on Neural Networks (ICNN'95), Perth, WA, Australia, November 27 - December 1, 1995. IEEE, 1995. P. 1942–1948. DOI: 10.1109/ICNN.1995.488968.
28. Usatyuk V., Vorobyev I. Simulated Annealing Method for Construction of High-Girth QCLDPC Codes // 41st International Conference on Telecommunications and Signal Processing, TSP 2018, Athens, Greece, July 4-6, 2018. IEEE, 2018. P. 1–5. DOI: 10.1109/TSP.2018.8441303.
29. Yaoumi M., Dupraz E., Leduc-Primeau F., Guilloud F. Energy optimization of quantized min-sum decoders for protograph-based LDPC codes // Ann. des Télécommunications. 2020. Vol. 75, no. 11. P. 615–621. DOI: 10.1007/S12243-020-00804-0.
30. Bocharova I.E., Johannesson R., Kudryashov B.D. A unified approach to optimization of LDPC codes for various communication scenarios // 8th International Symposium on Turbo Codes and Iterative Information Processing, ISTC 2014, 18-22 August 2014, Bremen, Germany. IEEE, 2014. P. 243–248. DOI: 10.1109/ISTC.2014.6955122.

31. Xiao Z., Li L., Xu J., Sha J. Construction of protograph LDPC codes based on the convolution neural network // *China Communications*. 2023. May. Vol. 20, no. 5. P. 84–92. DOI: 10.23919/jcc.2023.00.012.
32. Gao C., Liu S., Jiang D., Chen L. Constructing LDPC Codes with Any Desired Girth // *Sensors*. 2021. Vol. 21, no. 6. P. 1012. DOI: 10.3390/S21062012.
33. Lee H., Kim J. Modified CC-PEG Algorithm for Protograph-Based QC-LDPC Codes Over Non-Uniform Channel // *IEEE Access*. 2024. Vol. 12. P. 173660–173669. DOI: 10.1109/ACCESS.2024.3502712.
34. Yun D.-Y., Kim J.-W., Kwak H.-Y., No J.-S. The Cycle-Concentrating PEG Algorithm for Protograph Generalized LDPC Codes // *IEEE Access*. 2023. Vol. 11. P. 57285–57294. DOI: 10.1109/ACCESS.2023.3284314.
35. Bocharova I.E., Hug F., Johannesson R., Kudryashov B.D. A greedy search for improved QC LDPC codes with good girth profile and degree distribution // 2012 IEEE International Symposium on Information Theory Proceedings. IEEE, 07/2012. P. 3083–3087. DOI: 10.1109/isit.2012.6284129.
36. ISO/IEC/IEEE International Standard - Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications // ISO/IEC/IEEE 8802-11:2012(E) (Revision of ISO/IEC/IEEE 8802-11-2005 and Amendments). 2012. P. 1–2798. DOI: 10.1109/IEEESTD.2012.6361248.
37. Sinilshchikov I.V., Molotkov S. Decoy States and Low-Density Parity-Check ErrorCorrecting Codes in Quantum Cryptography with Phase-Time Coding // *Journal of Experimental and Theoretical Physics*. 2019. Aug. Vol. 129, no. 2. P. 168–196. DOI: 10.1134/s1063776119070124.
38. Gallager R. Low-density parity-check codes // *IRE Transactions on Information Theory*. 1962. Vol. 8, no. 1. P. 21–28. DOI: 10.1109/TIT.1962.1057683.
39. MacKay D. Good error-correcting codes based on very sparse matrices // *IEEE Transactions on Information Theory*. 1999. Vol. 45, no. 2. P. 399–431. DOI: 10.1109/18.748992.
40. Fossorier M., Mihaljevic M., Imai H. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation // *IEEE Transactions on Communications*. 1999. Vol. 47, no. 5. P. 673–680. DOI: 10.1109/26.768759.
41. Emran A.A., Elsabrouty M. Simplified variable-scaled min sum LDPC decoder for irregular LDPC codes // 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC). 2014. P. 518–523. DOI: 10.1109/CCNC.2014.6940497.
42. Slepian D., Wolf J. Noiseless coding of correlated information sources // *IEEE Transactions on Information Theory*. 1973. Vol. 19, no. 4. P. 471–480. DOI: 10.1109/TIT.1973.1055037.
43. Liveris A., Xiong Z., Georgiades C. Compression of binary sources with side information at the decoder using LDPC codes // *IEEE Communications Letters*. 2002. Vol. 6, no. 10. P. 440–442. DOI: 10.1109/LCOMM.2002.804244.
44. Kostenetskiy P.S., Chulkevich R.A., Kozyrev V.I. HPC Resources of the Higher School of Economics // *Journal of Physics: Conference Series*. 2021. Jan. Vol. 1740, no. 1. P. 012050. DOI: 10.1088/1742-6596/1740/1/012050.

Параллельный алгоритм для численного исследования дозвуковых реагирующих течений в цилиндрической системе координат

М.С. Мустайкин, Е.Е. Пескова

Национальный исследовательский Мордовский государственный университет

Статья посвящена разработке параллельного вычислительного алгоритма на основе технологии MPI для численного исследования дозвуковых газовых потоков в трубчатом реакторе. Математическая модель исследуемого процесса представляет собой уравнения газовой динамики в цилиндрической системе координат. Вычислительный алгоритм построен с учетом дозвукового характера течения. Разработанная программа показала рост ускорения и эффективности с увеличением количества расчетных ячеек.

Ключевые слова: математическое моделирование, дозвуковые течения, уравнения газовой динамики, технология MPI.

1. Введение

В настоящее время математическое моделирование является одним из основных методов исследования химически активных сред, наблюдение которых зачастую недоступно в лабораторных экспериментах. В связи с активным развитием вычислительных систем открывается возможность использовать в расчетах более сложные математические модели и большие сетки для получения результатов, наиболее приближенных к реальным течениям газа в химических реакторах.

Задачей, стоящей перед авторами статьи, является создание вычислительного алгоритма и программы для исследования дозвуковых химически активных газопылевых сред в присутствии лазерного излучения, поглощаемого газом и каталитическими наночастицами, в цилиндрической системе координат. Выбор системы координат обоснован необходимостью исследования газопылевых течений в круглых трубах. Трехмерная геометрия необходима для отражения вихревых процессов, наблюдаемых в исследуемых течениях, реализации различных углов ввода газовой смеси в трубу, несимметричного ввода лазерного излучения, что зачастую является необходимым условием интенсификации химических процессов.

В работах [1, 2] представлена математическая модель и разработан вычислительный алгоритм для исследования двухфазных газопылевых потоков с химическими реакциями и лазерным излучением для случая осесимметричных течений. Исходная математическая модель требует специализированного подхода при построении вычислительного алгоритма. Для обхода вычислительной сложности, возникающей при расчете изменений характеристик дозвуковых реагирующих сред, в алгоритме использован принцип расщепления по физическим процессам, ориентированный на выделение блоков для расчета каждого физического процесса с подбором адекватного численного метода. Сочетание быстрых химических реакций и медленных газовых потоков обуславливает необходимость выделения решения уравнений химической кинетики в отдельный блок. Отдельные трудности вызывает учет малых изменений давления в потоке. В условиях малых чисел Маха эти изменения незначительны, однако, они являются причиной изменения вектора скорости. В математической модели давление представлено в виде суммы давления, постоянного в области, и динамической поправки к давлению, которое меняется в каждой точке пространства на каждом шаге по времени. Также требует необходимости разработка отдельных блоков для учета лазерного излучения и характеристик каталитических наночастиц.

Целью настоящей работы является построение параллельного вычислительного алго-

ритма для уравнений газовой динамики в цилиндрической системе координат, поскольку уже на этапе учета только конвективных потоков и многокомпонентности смеси встала необходимость использования технологий параллельных вычислений.

2. Математическая модель и вычислительный алгоритм

Рассмотрим систему уравнений газовой динамики в цилиндрической системе координат. Данная система получена из уравнений Навье—Стокса в приближении малых чисел Маха с помощью отбрасывания диссипативных членов [3, 4].

$$\frac{\partial U}{\partial t} + \frac{\partial F^{(1)}(U)}{\partial z} + \frac{1}{r} \frac{\partial (rF^{(2)}(U))}{\partial r} + \frac{1}{r} \frac{\partial F^{(3)}(U)}{\partial \phi} = Q. \quad (1)$$

Здесь

$$U = \begin{pmatrix} \rho Y_m \\ \rho u_z \\ \rho u_r \\ \rho u_\phi \\ \rho h \end{pmatrix}, \quad F^{(1)}(U) = \begin{pmatrix} \rho u_z Y_m \\ \rho u_z^2 + \pi \\ \rho u_z u_r \\ \rho u_z u_\phi \\ \rho h u_z \end{pmatrix}, \quad F^{(2)}(U) = \begin{pmatrix} \rho u_r Y_m \\ \rho u_z u_r \\ \rho u_r^2 + \pi \\ \rho u_r u_\phi \\ \rho h u_r \end{pmatrix},$$

$$F^{(3)}(U) = \begin{pmatrix} \rho u_\phi Y_m \\ \rho u_z u_\phi \\ \rho u_r u_\phi \\ \rho u_\phi^2 + \pi \\ \rho h u_\phi \end{pmatrix}, \quad Q = \begin{pmatrix} R_m \\ 0 \\ (\rho u_\phi^2)/r \\ (-\rho u_r u_\phi)/r \\ 0 \end{pmatrix}.$$

Здесь ρ – плотность газа, Y_m – массовая доля компоненты газа, $m = \overline{1, M}$ – номер компоненты газа, M – количество компонент в газовой смеси, R_m – скорость образования или расхода m -ой компоненты смеси, u_z, u_r, u_ϕ – компоненты вектора скорости, $\pi = p - p_0$ – динамическое отклонение давления, p – давление, p_0 – давление, постоянное в области, R_m – скорость образования или расхода каждой компоненты смеси для случая наличия химических реакций.

Энтальпию газовой смеси можно определить из температуры смеси T и массовых долей ее компонент:

$$h(T, Y_m) = \sum_m Y_m h_m(T), \quad h_m(T) = \int_{T_{ref}}^T C_{pm}(T) dT + h_m^0. \quad (2)$$

Здесь $T_{ref} = 293.15 \text{ K}$, h_m^0 – энтальпия образования каждой компоненты газа, $C_{pm}(T) = a_0 + a_1 T + a_2 T^2 + a_3 T^3$ – удельная теплоемкость каждой компоненты смеси при постоянном давлении, заданная многочленом, аппроксимирующим известные термодинамические табличные данные в нужном интервале температур [5].

Для построения дискретной модели введём равномерную по каждому направлению сетку:

$$\Omega_h = \{z_i = i h_z, r_j = j h_r, \phi_k = k h_\phi; i = \overline{1, N_z}, j = \overline{1, N_r}, k = \overline{1, N_\phi}; N_z h_z = L_z, N_r h_r = L_r, N_\phi h_\phi = L_\phi\}$$

При построении вычислительного алгоритма используем схему расщепления по физическим процессам:

1. На первом шаге решается система уравнений химической кинетики:

$$\begin{cases} \frac{\partial \rho Y_1}{\partial t} = R_1, \\ \frac{\partial \rho Y_2}{\partial t} = R_2, \\ \dots \\ \frac{\partial \rho Y_M}{\partial t} = R_M. \end{cases} \quad (3)$$

Для решения данной системы используется подключаемый модуль RADAU5 [6]. В результате находятся промежуточные значения вектора $\widetilde{\rho Y}_m$, $m = \overline{1, M}$.

2. На втором шаге решается система уравнений (1) без учета вклада динамического отклонения давления и химических реакций:

$$\frac{U_{ijk}^{n+1} - U_{ijk}^n}{\Delta t} + \frac{\widetilde{F}_{i+1/2jk}^{(1)} - \widetilde{F}_{i-1/2jk}^{(1)}}{h_z} + \frac{r_{ij+1/2k} \cdot \widetilde{F}_{ij+1/2k}^{(2)} - r_{ij-1/2k} \cdot \widetilde{F}_{ij-1/2k}^{(2)}}{r_{ijk} \cdot h_r} + \frac{\widetilde{F}_{ijk+1/2}^{(3)} - \widetilde{F}_{ijk-1/2}^{(3)}}{r_{ijk} \cdot h_\phi} = Q_{ijk}. \quad (4)$$

Здесь $\widetilde{F}_{i+1/2jk}^{(1)}$, $\widetilde{F}_{ij+1/2k}^{(2)}$, $\widetilde{F}_{ijk+1/2}^{(3)}$ – дискретные потоки между i и $i+1$, j и $j+1$, k и $k+1$ ячейками соответственно, для расчета которых применяется поток Русанова [7]:

$$\begin{aligned} \widetilde{F}_{i+1/2jk}^{(1)} &= 0.5 \left(\widetilde{F}^{(1)} \left(U_{i+1/2jk}^+ \right) + \widetilde{F}^{(1)} \left(U_{i+1/2jk}^- \right) - \alpha_1 \left(U_{i+1/2jk}^+ - U_{i+1/2jk}^- \right) \right), \\ \widetilde{F}_{ij+1/2k}^{(2)} &= 0.5 \left(\widetilde{F}^{(2)} \left(U_{ij+1/2k}^+ \right) + \widetilde{F}^{(2)} \left(U_{ij+1/2k}^- \right) - \alpha_2 \left(U_{ij+1/2k}^+ - U_{ij+1/2k}^- \right) \right), \\ \widetilde{F}_{ijk+1/2}^{(3)} &= 0.5 \left(\widetilde{F}^{(2)} \left(U_{ijk+1/2}^+ \right) + \widetilde{F}^{(2)} \left(U_{ijk+1/2}^- \right) - \alpha_3 \left(U_{ijk+1/2}^+ - U_{ijk+1/2}^- \right) \right). \end{aligned}$$

$U_{i+1/2jk}^+$, $U_{i+1/2jk}^-$ – значения вектора переменных U слева и справа от грани между i и $i+1$ ячейками, для которой вычисляется поток $\widetilde{F}_{i+1/2jk}^{(1)}$, $U_{ij+1/2k}^+$, $U_{ij+1/2k}^-$ – значения вектора переменных U слева и справа от грани между j и $j+1$ ячейками, для которой вычисляется поток $\widetilde{F}_{ij+1/2k}^{(2)}$, $U_{ijk+1/2}^+$, $U_{ijk+1/2}^-$ – значения вектора переменных U слева и справа от грани между k и $k+1$ ячейками, для которой вычисляется поток $\widetilde{F}_{ijk+1/2}^{(2)}$.

Стабилизирующий член потока находится из выражений [8, 9]:

$$\begin{aligned} \alpha_1 &= \max \left(\sqrt{\left(u_{z,i+1/2jk}^+ \right)^2 + \left(u_{r,i+1/2jk}^+ \right)^2 + \left(u_{\phi,i+1/2jk}^+ \right)^2}, \right. \\ &\quad \left. \sqrt{\left(u_{z,i+1/2jk}^- \right)^2 + \left(u_{r,i+1/2jk}^- \right)^2 + \left(u_{\phi,i+1/2jk}^- \right)^2} \right), \\ \alpha_2 &= \max \left(\sqrt{\left(u_{z,ij+1/2k}^+ \right)^2 + \left(u_{r,ij+1/2k}^+ \right)^2 + \left(u_{\phi,ij+1/2k}^+ \right)^2}, \right. \\ &\quad \left. \sqrt{\left(u_{z,ij+1/2k}^- \right)^2 + \left(u_{r,ij+1/2k}^- \right)^2 + \left(u_{\phi,ij+1/2k}^- \right)^2} \right), \end{aligned}$$

$$\alpha_3 = \max \left(\sqrt{\left(u_{z,ijk+1/2}^+\right)^2 + \left(u_{r,ijk+1/2}^+\right)^2 + \left(u_{\phi,ijk+1/2}^+\right)^2}, \right. \\ \left. \sqrt{\left(u_{z,ijk+1/2}^-\right)^2 + \left(u_{r,ijk+1/2}^-\right)^2 + \left(u_{\phi,ijk+1/2}^-\right)^2} \right).$$

В результате находятся плотность ρ , концентрации компонент смеси Y_m , энтальпия h , предварительные компоненты вектора скорости u_r^* , u_z^* , u_ϕ^* , из решения уравнения (2) методом Ньютона находится температура T .

3. На третьем шаге рассчитывается динамическая поправка к давлению и проводится коррекция вектора скорости.

Для расчета динамической поправки к давлению используется выражение [3]:

$$\nabla \cdot \frac{1}{\rho^n} \nabla \pi^{n+1} = \frac{1}{\Delta t} (\nabla \cdot \vec{u}^* - \nabla \cdot \vec{u}^{n+1}). \quad (5)$$

Здесь $\vec{u}^{n+1} = \frac{1}{\rho_g} \sum_m \left(\frac{M_w}{M_{wm}} - \frac{h_m}{C_p T_g} \right) R_m$ – выражение для дивергенции вектора скорости [3], M_w , M_{wm} – молекулярная масса смеси и компоненты соответственно.

Для решения уравнения Пуассона (5) используется метод Якоби:

$$\frac{\pi_{i+1jk} - 2\pi_{ijk} + \pi_{i-1jk}}{h_z^2} + \frac{\pi_{ij+1k} - 2\pi_{ijk} + \pi_{ij-1k}}{h_r^2} + \frac{1}{r_{ijk}^2} \frac{\pi_{ijk+1} - 2\pi_{ijk} + \pi_{ijk-1}}{h_\phi^2} + \quad (6)$$

$$+ \frac{1}{r_{ijk}} \frac{\pi_{ij+1k} - \pi_{ij-1k}}{2h_r} = G_{ijk}, \quad G_{ijk} = \frac{\rho^n}{\Delta t} (\nabla \cdot \vec{u}^* - \nabla \cdot \vec{u}^{n+1}). \quad (7)$$

После нахождения динамической поправки к давлению корректируем вектор скорости:

$$\vec{u}_z^{n+1} = u_z^* - \frac{\Delta t}{\rho^n} \frac{\partial \pi^{n+1}}{\partial z}, \quad (8)$$

$$\vec{u}_r^{n+1} = u_r^* - \frac{\Delta t}{\rho^n} \frac{\partial \pi^{n+1}}{\partial r}, \quad (9)$$

$$\vec{u}_\phi^{n+1} = u_\phi^* - \frac{\Delta t}{\rho^n} \frac{1}{r} \frac{\partial \pi^{n+1}}{\partial \phi}. \quad (10)$$

$$(11)$$

В результате находятся динамическое отклонение от давления π , компоненты вектора скорости u_r , u_z , u_ϕ .

3. Параллельная реализация алгоритма

Расчетная область представляет собой цилиндр, покрытый равномерной сеткой с числом ячеек G_Nz , G_Nr , G_Np по соответствующему направлению (рис. 1).

За основу параллельного алгоритма принята трехмерная геометрическая декомпозиция области. Область разбивается на sz_z частей по оси OZ, sz_r частей по оси OR и sz_p по ОР с равным количеством ячеек в каждой из образовавшихся подобластей. Каждая подобласть будет обрабатываться параллельно, и решение будет производиться по одинаковой схеме на каждом вычислительном узле. Для сохранения однородности схемы вычисления, по всем границам подобласти вводятся по одному слою фиктивных ячеек. Значения газодинамических параметров в этих дополнительных ячейках определяются либо исходя из граничных условий, если область является приграничной, либо посредством межпроцессорных обменов с соседними областями. Общее количество параллельных процессов равно $sz\_z \cdot sz\_r \cdot sz\_p$.

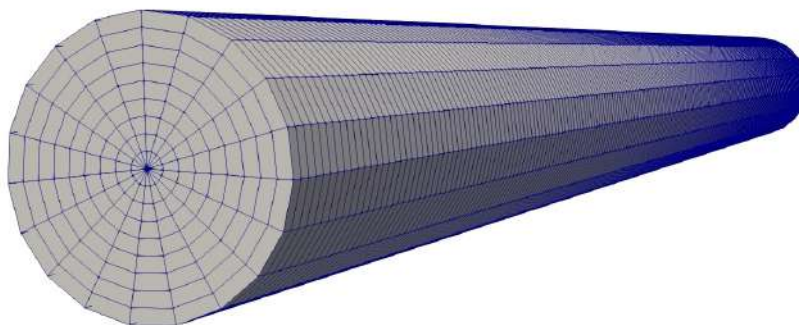


Рис. 1. Расчетная сетка

Параллельная программа разработана с использованием технологии для систем с распределенной памятью MPI, реализация механизмов межпроцессорного обмена осуществляется с помощью библиотеки MPICH. При запуске программы в первую очередь определяются соседи процесса по каждой границе и начальные координаты области на каждом параллельном процессе для возможности вычисления координаты любой ячейки. Начальных координат области достаточно, поскольку сетка равномерная по каждому направлению. Координаты ячейки необходимы для определения областей ввода смеси, лазерного излучения и вывода данных в файл. Далее, зная размерность сетки на каждом процессе $Nz \times Nr \times Nr$, выделяется память под массивы газодинамических параметров и происходит их заполнение начальными данными. Далее вызывается процедура выполнения одного шага по времени, которая имеет следующую структуру:

1. Организации граничных условий в фиктивных ячейках. На исходной границе условия определяются обычным способом (условия втекания, вытекания, прилипания, отражения), в случае внутренней границы фиктивные ячейки заполняются посредством межпроцессорных обменов между соседними областями. Для пересылки данных в программе используются функции `MPI_Send()` и `MPI_Recv()`.
2. Процедура расчета уравнений химической кинетики.
3. Процедура расчета уравнений газовой динамики.
4. Процедура решения уравнения Пуассона для динамической поправки к давлению. Поскольку используется метод Якоби, который требует небольшого количества итераций, внутри этой процедуры происходит межпроцессорный обмен до сходимости метода.

Вывод результатов производится каждым процессом в файлы с расширением *.vts, для визуализации используется программа ParaView.

4. Тестовый расчет

Параллельная версия предложенного алгоритма была реализована программно на языке C++. Для получения информации о производительности были выполнены расчеты для следующей задачи. Рассматривалась цилиндрическая труба диаметром 0.02 м и длиной 0.256 м. В начальный момент времени труба заполнена покоящимся метаном. Температура в области 600°C, давление 101325 Па. Слева в трубу поступает тоже метан, но с температурой 700°C и расходом 60 л/ч. На выходе из трубы справа задано условие вытекания, давление 101325 Па. Перепад давления на входе и выходе трубы задан 10^{-2} Па.

Поскольку нашей задачей является разработка основы параллельного алгоритма для дозвуковых двухфазных реагирующих течений и программа в дальнейшем будет расши-

ряться, температуры в начальных условиях приняты такие, что химические реакции разложения метана отсутствуют, смесь считается однокомпонентной. На рис. 2–3 представлены распределения плотности и температуры смеси, которые соответствуют физике процесса: с увеличением температуры в области посредством втекания горячего газа, плотность в этой области понижается.

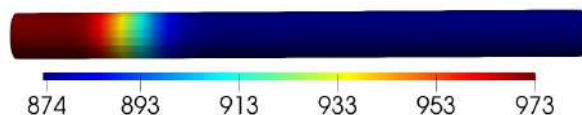


Рис. 2. Температура, К



Рис. 3. Плотность, кг/м³

Для оценки эффективности параллельного алгоритма измерялось время, затрачиваемое на выполнение определенного числа шагов по времени, с использованием различного количества параллельных процессов и расчетных ячеек. Расчеты проводились на рабочей станции ФГБОУ ВО «МГУ им. Н.П. Огарева» AMD Ryzen Threadripper 3990X 2900 МГц, 64 ядра.

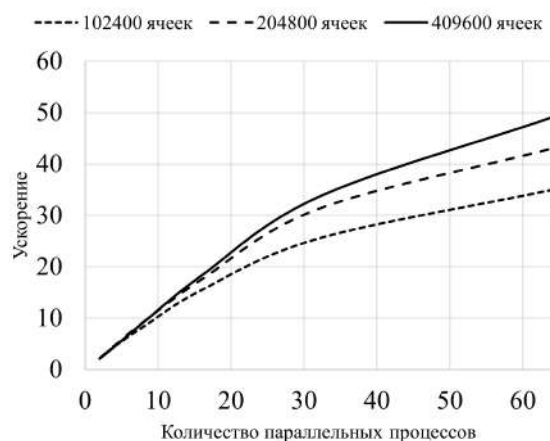


Рис. 4. Ускорение

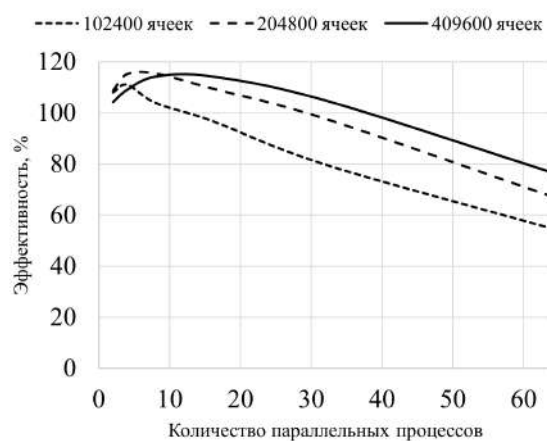


Рис. 5. Эффективность

При небольшом количестве параллельных процессов наблюдается суперлинейное ускорение и эффективность оказывается выше 100%. Однако, с увеличением количества процессов до 64 эффективность падает. Это связано с тем, что при увеличении количества декомпозиционных подобластей увеличивается и время, затраченное на межпроцессорные обмены. Из графиков видно, что с увеличением размерности расчетной сетки ускорение постепенно приближается к количеству вычислительных узлов и эффективность растет. Ожидается, что в случае включения громоздких схем химических реакций, которые будут независимо просчитываться на каждом процессе, эффективность на большом количестве процессов будет расти.

5. Заключение

Разработан параллельный вычислительный алгоритм с использованием технологии MPI для решения уравнений газовой динамики в цилиндрической системе координат с блочной структурой. Для разработанного параллельного алгоритма исследованы эффективность и ускорение для различного числа используемых процессоров. Сделаны выводы, что результирующий алгоритм позволит проводить серийные расчеты для различных углов ввода газовой смеси, а его блочная структура позволит расширить модель на учет различных эф-

фектов, возникающих в задачах моделирования химически активных двухфазных течений с лазерным излучением.

Литература

1. Snytnikov V.N., Peskova E.E., Stoyanovskaya O.P. Mathematical Model of a Two-Temperature Medium of Gas-Solid Nanoparticles with Laser Methane Pyrolysis // *Mathematical Models and Computer Simulations*. 2023. Vol. 15. P. 877–893. DOI: 10.1134/S2070048223050095.
2. Пескова Е.Е., Снытников В.Н., Жалнин Р.В. Вычислительный алгоритм для изучения внутренних ламинарных потоков многокомпонентного газа с разномасштабными химическими процессами // *Компьютерные исследования и моделирование*. 2023. Т. 15, № 5. С. 1169–1187. DOI: 10.20537/2076-7633-2023-15-5-1169-1187.
3. Day M.S., Bell J.B. Numerical simulation of laminar reacting flows with complex chemistry // *Combustion Theory and Modelling*. 2000. Vol. 4, no. 4. P. 535–556. DOI: 10.1088/1364-7830/4/4/309.
4. Борисов В.Е., Якуш С.Е. Применение адаптивных иерархических сеток для расчета течений реагирующих газов // *Физико-химическая кинетика в газовой динамике*. 2015. Т. 16, № 2. С. 1–13.
5. Stadnichenko O.A., Snytnikov V.N., Snytnikov V.I., Masyuk N.S. Mathematical modeling of ethane pyrolysis in a flow reactor with allowance for laser radiation effects // *Chemical Engineering Research and Design*. 2016. Vol. 109, P. 405–413. DOI: 10.1016/j.cherd.2016.02.008.
6. Hairer E., Wanner G. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer-Verlag, Berlin, 1996.
7. Русанов В.В. Расчет взаимодействия нестационарных ударных волн с препятствиями // *Журнал вычислительной математики и математической физики*. 1961. Т. 1, № 2. С. 267–279.
8. Klein B., Muller B., Kummer F., Oberlack M. A high-order discontinuous Galerkin solver for low Mach number flows // *International Journal for Numerical Methods in Fluids*. 2015. Vol. 81, no. 8. P. 489–520. DOI: 10.1002/fld.4193.
9. Peskova E.E. Mathematical Modeling of Nonstationary Problems Related to Laser Thermochemistry of Methane in the Presence of Catalytic Nanoparticles // *Doklady Mathematics*. 2024. Vol. 109, no. 3. P. 256–261. DOI: 10.1134/S1064562424702107.

Механические свойства композита Al_2Ni_3 , армированного углеродной нанотрубкой, при одноосной деформации растяжения*

А.С. Нарсеев, П.В. Захаров, У.И. Янковская

ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого»

В работе произведено моделирование металлматричного композита Al_2Ni_3 с армированием углеродными нанотрубками (УНТ). Интерес к таким материалам обусловлен, зачастую, уникальными свойствами композитов по сравнению с обычными металлами и сплавами. В данной работе моделирование произведено методом молекулярной динамики с последующими вычислениями на суперкомпьютере СПбПУ. Качественное описание взаимодействия обеспечено применением гибридной функции к межатомным потенциалам, полученным методом погруженного атома. Получены результаты для чистого интерметаллида Al_2Ni_3 и для Al_2Ni_3 -УНТ с различным количеством трубок в единице объема металла. Основными параметрами, которые оценивались в работе являются: предел упругости, предел прочности, деформация разрушения и нагрузка разрушения. Все параметры изучались в широком диапазоне температур от 300 до 1500 К. Это позволило отследить в динамике роль температурного фактора на параметры композита. Деформация прикладывалась к направлению кристалла, соответствующего [100] в индексах Миллера. Ось нанотрубки также располагалась вдоль этого направления. Полученные результаты показали ухудшение механических свойств композита по сравнению с чистым сплавом Al_2Ni_3 . «Нетипичная» ситуация обусловлена рядом факторов. В первую очередь это слоистая структура сплава вдоль этого направления. Внедрение нанотрубки привело к ослаблению связей между слоями разных типов атомов. Важную роль внесли и условия эксперимента в виде периодических граничных условий, накладываемых на модель. Данные результаты говорят о том, что вклад металлической матрицы и формирование дефектов в ней при внедрении нанотрубки в результирующие свойства композита являются решающими.

Ключевые слова: металлматричный композит, УНТ, метод молекулярной динамики, механические свойства композита, углеродные наноструктуры.

1. Введение

Композиты в наше время пользуются огромным интересом в качестве конструкционных материалов в различных сферах промышленности и науки. Их свойства, в отличие от чистых материалов, бывают более сбалансированы и эффективны [1]. Отдельный интерес представляют металлматричные композиты на основе интерметаллидов и низкоразмерных углеродных наноструктур. Такие материалы позволяют снизить плотность и повысить функциональные свойства. В данной статье мы рассмотрим характеристики композита Al_2Ni_3 -УНТ, его механические и структурные характеристики при одноосной деформации при различных температурах, ниже температуры плавления, и различной объемной доле нанотрубок. Металлматричные композиты являются активно развивающейся и перспективной отраслью благодаря своему набору свойств. Нанотрубки, а в частности углеродные нанотрубки, являются наиболее распространенным армирующим материалом, применяющимся для упрочнения. Сплавы на основе соединения Al_2Ni_3 нашли промышленное применение. Прежде всего они известны благодаря своим высокотемпературным свойствам (высокое сопротивление

* Исследование выполнено за счет гранта Российского научного фонда № 24-22-20038 (<https://rscf.ru/project/24-22-20038/>) и гранта Санкт-Петербургского научного фонда № 24-22-20038. Часть результатов работы была получена с использованием вычислительных ресурсов суперкомпьютерного центра Санкт-Петербургского политехнического университета Петра Великого (www.scc.spbstu.ru).

ползучести, сохранение прочностных характеристик при повышенных температурах, хорошая стойкость к окислению) Сплавы на основе алюминидов никеля применяют для изготовления литейных форм (в производстве стекла), штампов, валков для прокатки стальных заготовок, направляющих роликов при реализации процесса непрерывного литья, элементов муфельных печей и др.

Интерес к соединениям с Al очевиден, поскольку алюминиевые сплавы широко используются в различных технических устройствах: от бытовых до космических. Так, например, твердость композита Al/УНТ значительно возрастает в зависимости от массовой доли УНТ в нем. В работе [2] определили процентное содержание УНТ для максимальной твердости материала – 1,5 %. Полученный показатель в три раза выше, чем твердость чистого алюминия. Такие результаты обусловлены, помимо прочих факторов, хорошей адгезией Al-УНТ. По результатам исследований [3] видно, что чем больше плотность УНТ в композите, тем выше его жесткость и ниже масса.

Не менее важна такая характеристика как теплопроводность, она измеряется до и после испытаний на растяжение, позволяя выявить зависимость деформации от теплопроводности. Механизм разрушения при растяжении заключается в образовании трещин в металлической матрице и затем их последующее расширение до пустот. Этот вопрос разобран в работе [4].

Существует ряд проблем, связанных с получением композитов металл-углерод [1–5]. В частности трудности, связанные с контролем механических свойств, низкой адгезией металлов и нанотрубок, слабой дисперсностью УНТ в металлической матрице. Часть вопросов рассмотрена в нашей работе.

Распределение УНТ не менее важно, чем объемная доля УНТ. Так например, качественное диспергирование УНТ в слоистом композите Cu/УНТ позволило ему достичь высокой прочности. Слоистая структура обеспечила пластичность композита [5]. Также наличие УНТ позволяет контролировать размер зерен металлической матрицы, позволяет увеличить стабильность, закрепляя границы зерен.

Механические свойства никель-алюминиевого сплава рассмотрены нами при различных объемных долях нанотрубок (от 1,01 % до 5,06 %, радиус трубок равен 3,31 Å, количество варьируется от 1 до 4) при одноосном растяжении (вдоль оси трубок) методами молекулярной динамики. Моделирование деформации проводилось при различных температурах (300, 500, 700, 900, 1100 и 1500 K). В сравнении с чистым металлом прочность армированного композита снижается. При сравнении результатов видно, что прочность образцов уменьшается и при повышении температуры. Это связано со слоистой структурой композита и невысокой силой взаимодействия атомов Al-Ni, что не дает осевой прочности УНТ укрепить композит.

Экспериментальные методы исследования часто требуют много времени и усилий, и они не раскрывают кинетику процессов на атомном уровне. С появлением суперкомпьютеров и высокопроизводительных численных методов изучение многих явлений перешло в виртуальный мир. Методы молекулярно динамического моделирования в настоящее время являются общепринятым дополнением к экспериментам. Численные методы широко используются в качестве вспомогательных инструментов для прогнозирования и оценки лабораторных измерений.

В данной работе мы впервые рассчитали эти характеристики для данного материала. При этом следует сказать, что УНТ не всегда может быть эффективным армирующим материалом для металлматричных композитов при растяжении. Применение суперкомпьютерной техники позволило произвести достаточно число итераций для получения достоверного результата.

2. Методы и средства

Метод молекулярной динамики способен оперировать сотнями тысяч частиц, что удобно для понимания механизмов поведения материалов в наномасштабе и оценки механических свойств нанокompозитов.

Молекулярная динамика хорошо подходит для расчета взаимодействия металл-углерод на уровне частиц. Этой теме посвящено большое количество публикаций, в которых основное внимание уделяется компьютерному моделированию, например [6]. Благодаря атомному разрешению МД также используется для разработки и тестирования новых композитных

наноструктур с улучшенной межфазной адгезией и/или дисперсией нанонаполнителей. Например, авторы работы [7] с помощью МД протестировали новые конструкции матриц в композитах УНТ-Al и УНТ-Ni. Кроме того, в работе [8] метод МД был использован для углубленного изучения морфологии взаимодействия алюминия и графена. В работе [9] графен был использован в качестве нанонаполнителя для композитов на основе Cu с целью изучения влияния пористости на межфазную адгезию. Кроме того, в работе [10] провели МД-моделирование для изучения влияния никелевого покрытия на механическое поведение одностенных УНТ и композитов со алюминиевой матрицей. В результате оказалось, что в композите УНТ-Al с никелевым покрытием модуль Юнга значительно выше, чем в чистом композите УНТ-Al, несмотря на то, что модуль Юнга для УНТ с никелевым покрытием значительно ниже, чем для УНТ без покрытия. Они отметили, что передача возрастающей нагрузки между УНТ и алюминиевой матрицей в композите через никелевое покрытие может быть достаточно эффективной. Результаты также показывают, что Ni-покрытие нанотрубок резко увеличивает их межфазное соединение с алюминиевой матрицей. В работе [11] с помощью метода МД-моделирования оценивается механическое поведение композитного материала УНТ-Al при сжатии.

Данное исследование будет полезно при разработке композитов Al_2Ni_3 -УНТ, поскольку в нем представлена модель, способная прогнозировать жесткость.

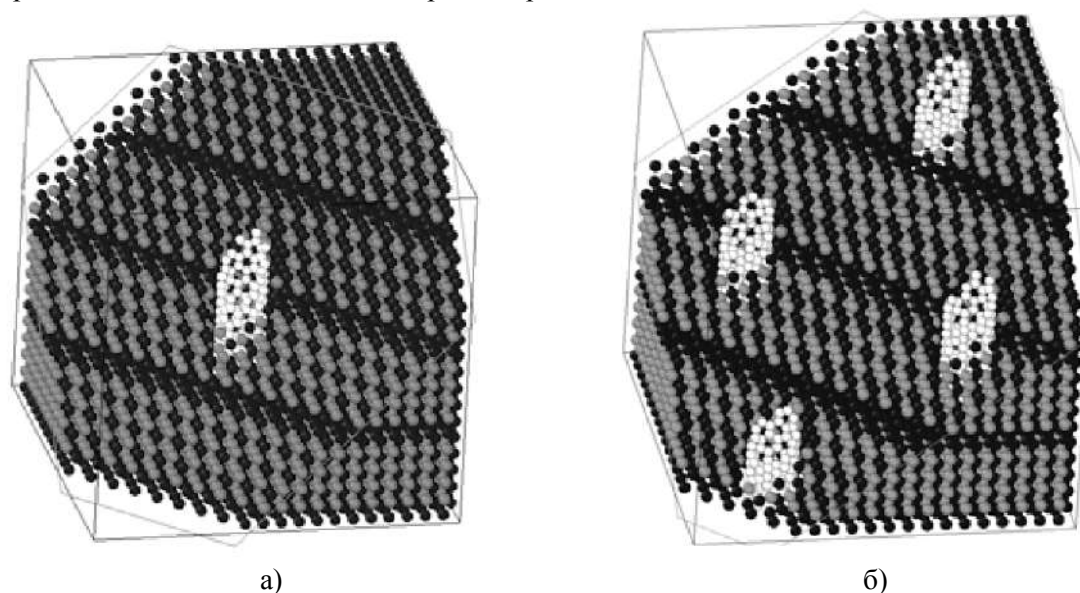


Рис. 1. Модели композита, армированного одной (а) и четырьмя (б) УНТ

В основе модели лежит гранецентрированный кубический кристалл интерметаллида Al_2Ni_3 постоянной решетки $a = 2,87 \text{ \AA}$. Линейные размеры составляют $58.3216 \text{ \AA} \times 58.3216 \text{ \AA} \times 58.3216 \text{ \AA}$. Оси X, Y и Z расположены вдоль кристаллографических направлений [100], [010], [001]. Периодические граничные условия накладывались на все трехмерные направления. Общее количество частиц варьировалось от 15984 до 16000 в зависимости от размера УНТ и их количества. Для изучения влияния температуры в монокристаллический металлический блок была интегрирована однослойная углеродная нанотрубка типа «зигзаг» вдоль оси Z (рис. 1а) с хиральными индексами (0,8), длиной $L = 58,32 \text{ \AA}$ и диаметром $D = 6,62$, количество трубок изменялось от 1 до 4, цилиндрическое отверстие было ориентировано по оси Z и по всей высоте Al_2Ni_3 , и все атомы металла внутри заданного цилиндрического пространства были удалены. Затем отверстие заполнялось УНТ, таким образом, обеспечивалось расстояние $3 \text{ \AA}$ между атомами C и Al_2Ni_3 вдоль границы раздела поликристалла. Эти параметры взаимной конфигурации показали минимум энергии.

Рассматривалась система с одной, двумя или четырьмя УНТ. Это было сделано для оценки вклада массовой и объемной доли в механические свойства кристаллов. В модели нанокompозита Al_2Ni_3 -УНТ с разным количеством нанотрубок использовались УНТ одинакового диаметра и хиральности (0,16). В таблице 1 приведены объемные и массовые доли углеродных нанотрубок

для всех моделей, использованных в данном исследовании. Периодические граничные условия применялись во всех направлениях, чтобы получить объемные свойства без эффектов краевой поверхности композита Al_2Ni_3 -УНТ. Основные структурные конструкции для моделирования были выполнены по программе AtomsK [12]. В качестве программного пакета для расчетов методом молекулярной динамики использовался «LAMMPS» [13]. Он обладает необходимым функционалом для данной работы и дальнейшего анализа деформации композитов. Ранее он показал себя как эффективный инструмент для анализа различных аспектов трансформации структуры кристаллических решеток в результате внешних воздействий, включая исследование дефектов динамических и топологических, устойчивости к нагреву кристаллов, армированных углеродными нанотрубками [14], анализ деформации. Для описания взаимодействия Al-Ni, Al-УНТ, Ni-УНТ был выбран потенциал EAM (модель погруженного атома). На начальном этапе была проведена минимизация энергии модели с последующей релаксацией 1 нс при заданной температуре и нулевом давлении. Это помогло избавиться от напряжения и достичь более стабильного состояния кристаллической структуры. Для визуализации результатов расчетов и их графического представления, последующего анализа использовалась программа OVITO [15]. На рис. 2 приведен пример модели композита Al_2Ni_3 -УНТ после первичной релаксации.

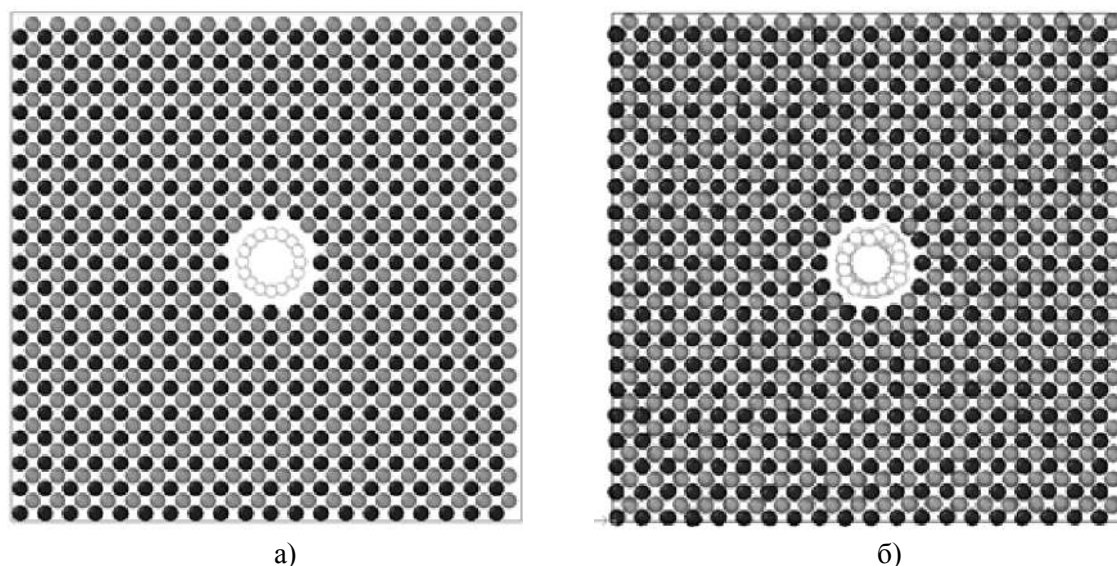


Рис. 2. Модель композита Al_2Ni_3 -УНТ до (а) и после (б) первичной релаксации

Механические напряжения рассчитываются на основе уравнения (1):

$$\sigma(r) = \frac{1}{\Omega} \sum_i \left(-m_i \dot{u}_i \otimes \dot{u}_i + \frac{1}{2} \sum_{i \neq j} r_{ij} \otimes f_{ij} \right), \quad (1)$$

где Ω – общий объем; m_i – масса атома i ; u_i – скорость i -го атома; r_{ij} – расстояние между векторами положения r_i и r_j атомов i и j , соответственно; f_{ij} – межатомная сила, действующая на атом i со стороны атома j [16]. Ω рассчитывается в равновесном состоянии, а общий объем композита Al_2Ni_3 -УНТ рассчитывается как сумма объемов интерметаллида с отверстием и УНТ. Моделирование растяжения проводилось до тех пор, пока деформация ε не достигнет предельных значений (около 70 %). Расчеты проводились в рамках канонического ансамбля NVT в диапазоне температур 300–1500 К, с шагом по времени 0,5 фс. Также было проведено моделирование растяжения интерметаллида, армированного в другой плоскости, вдоль слоев Al, с теми же параметрами.

Для проведения описанной методики расчета применялись ресурсы суперкомпьютерного центра «Политехнический». Расчет таких задач является достаточно трудоемким процессом в силу большого количества частиц в модели и необходимости моделирования процесса растяжения/сжатия с учетом реальных скоростей деформации материалов. Для реализации

поставленных задач по расчету кривых деформации использовалось 3 вычислительных узла гетерогенного вычислительного кластера «Торнадо», каждый по 56 ядер. Данные вычислительные ресурсы позволили рассчитать каждую кривую деформации за 10 часов, что является приемлемым при решении такого масштаба задач.

3. Результаты и обсуждение

Количество УНТ и их объемная доля в металломатричных композитах оказывают большое влияние на их механические свойства. Большинство исследований показывает, что УНТ оказывают упрочняющее действие на металломатричные композиты. Считается, что хороший армирующий эффект углеродных нанотрубок в металломатричных композитах связан с высокой несущей способностью УНТ, обусловленной их высокой прочностью. Но в данном исследовании УНТ наоборот, сделали материал более хрупким. Мы можем предположить, что это связано со слоистой структурой композита и меньшими силами взаимодействия Al-Ni по сравнению с Al-C и Ni-C.

Объемная доля УНТ является одним из факторов, влияющих на механические свойства композитных материалов. На рис. 3 представлена зависимость напряжения от деформации композита Al_2Ni_3 -УНТ для моделей с различными объемными долями УНТ при различных температурах, а также для чистого композита. Все эти кривые имеют линейный участок в начале, представляющий собой упругую область. Переход в пластическую область почти незаметен, но вывод о его существовании можно сделать из характера изменения наклона прямой, она плавно становится более «пологой». Отмечено, что наклоны кривых напряжение-деформация для моделей с различными объемными долями УНТ незначительно уменьшаются с ростом температуры. Это связано с тепловыми флуктуациями, которые увеличивают скорости атомов при высоких температурах и делают нанокompозиты гораздо мягче [17]. При более высоких температурах, близких к температурам плавления, эти изменения практически исчезают.

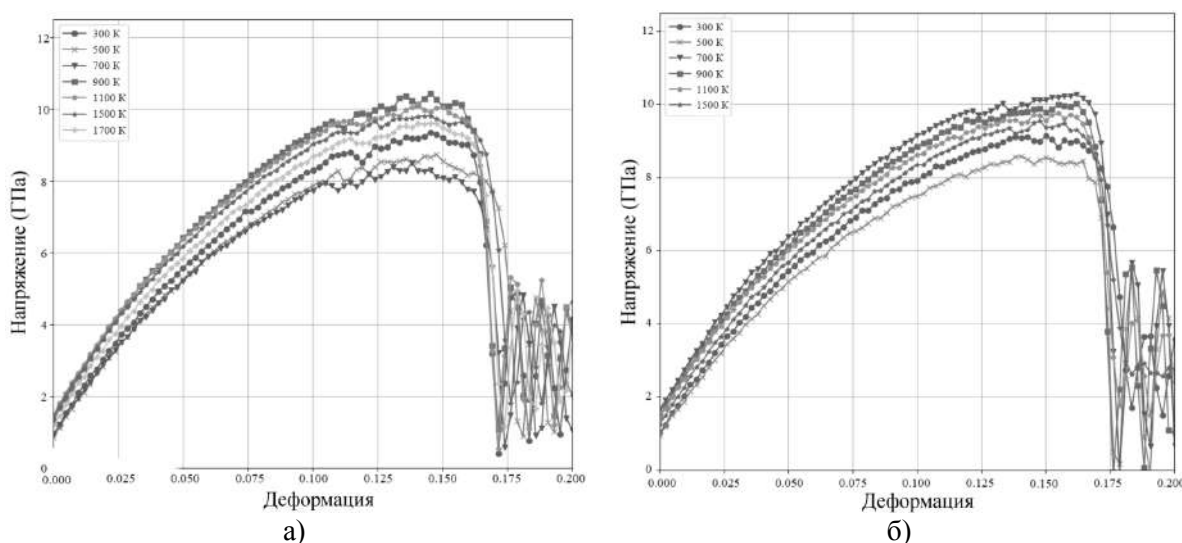


Рис. 3. Зависимости напряжения от деформации для моделей с 1 (а) и 2 (б) УНТ в качестве армирующих элементов при разных температурах

Тенденция на всех графиках имеет общий характер. Оценка результатов моделирования происходила согласно ГОСТу 1497-84.

Для наглядного понимания влияния температуры и объемной доли углеродных нанотрубок на механические свойства композита Al_2Ni_3 -УНТ были измерены значения модуля Юнга, предела прочности, деформации разрушения и предела текучести для исследуемой модели композита при растяжении. Они представлены в таблице 1. Результаты измерения модуля Юнга, предела прочности, деформации разрушения и предела текучести композита Al_2Ni_3 -УНТ при различных температурах представлены на рис. 4. Авторы работы [7] установили, что существует оптимальная объемная доля УНТ для композита. Если диаметр УНТ слишком мал, то большого

эффекта от армирования не будет. Если диаметр слишком велик, стабильность композитной системы снижается, и композит становится нестабильным при деформации. Наибольшее значение модуля Юнга при растяжении, равное 57,8 ГПа, измерено при 1 нанотрубке (с массовой долей 6,96 %) и температуре 300 К. От количества нанотрубок модуль Юнга не зависит напрямую исходя из графиков, но зависит от температуры. При ее увеличении модуль Юнга уменьшается.

Таблица 1. Механические характеристики металлматричных композитов Al_2Ni_3 -УНТ

Кол-во трубок	T, К	Предел упругости	Предел прочности	Деформация разрушения	Нагрузка разрушения
0	300	41,5	120,5	0,165	117,9
	500	44,2	116,4	0,160	115,2
	700	40,4	112,7	0,158	111,3
	900	44,7	108,0	0,162	108,0
	1100	44,8	104,6	0,155	104,3
	1500	44,0	98,5	0,154	97,2
1	300	57,8	105,8	0,168	100,3
	500	52,9	103,3	0,167	99,7
	700	52,4	99,2	0,166	86,9
	900	48,2	96,0	0,165	85,4
	1100	45,6	93,1	0,164	83,1
	1500	37,6	87,7	0,160	76,6
2	300	54,5	102,3	0,177	97,4
	500	47,8	99,7	0,169	95,3
	700	47,1	9,67	0,167	94,1
	900	41,5	9,38	0,170	89,5
	1100	37,1	90,5	0,174	86,2
	1500	35,7	84,4	0,171	78,6
4	300	40,3	88,4	0,178	83,1
	500	36,1	86,8	0,179	71,5
	700	39,4	84,5	0,171	77,1
	900	34,5	81,8	0,173	72,8
	1100	32,2	80,3	0,172	70,4
	1500	28,1	72,0	0,174	60,8

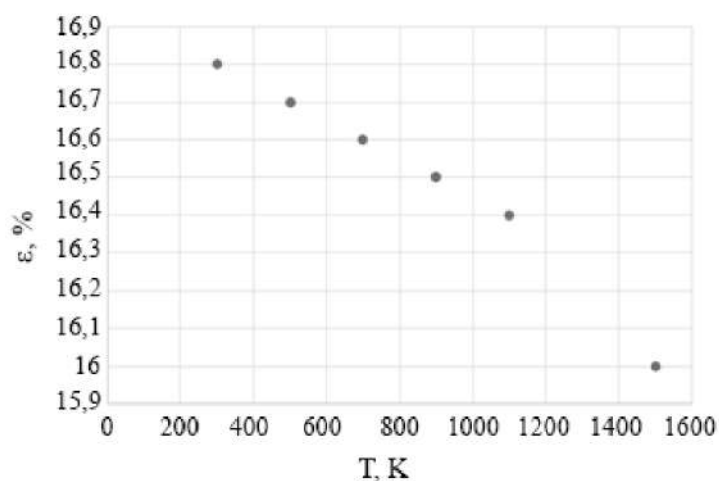


Рис. 4. Зависимость относительного удлинения от температуры, при которой происходит разрушение УНТ

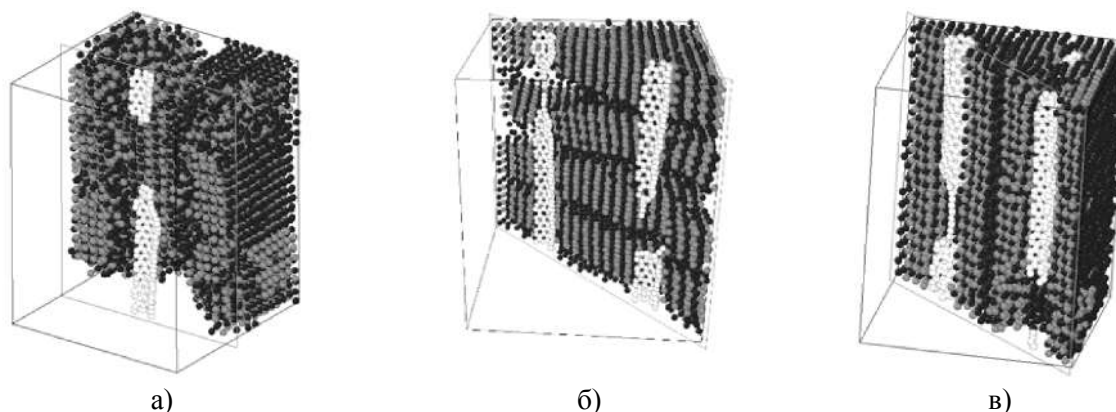


Рис. 5. Визуализация разрушения композита, содержащего 1 (а), 2 (б) и 4 (в) УНТ

Разрушение композита, армированного вдоль рассматриваемого направления, сопровождается формированием дислокационных петель. При этом само разрушение металлической матрицы происходит путем ее расслоения на стыке слоев Al и Ni.

4. Выводы

Таким образом, в данной работе методом молекулярной динамики исследованы механические свойства композита Al_2Ni_3 , армированного углеродными нанотрубками, при одноосном растяжении. Результаты показывают, что добавление углеродных нанотрубок незначительно влияют на модуль Юнга, изменяя его приблизительно на 11 % и уменьшает прочностные характеристики материала на величину порядка 40 %. Можно предположить, что это связано со структурой композита. Слои Al и Ni связаны друг с другом меньшей силой, чем атомы Al или Ni с УНТ, из-за этого происходит расслоение и образование дефектов, данное явление можно увидеть на рисунках выше. Предел текучести и деформация текучести композита Al_2Ni_3 -УНТ также не увеличиваются по сравнению с чистым композитом. Плотность нанотрубок играет важную роль, поскольку более высокая плотность УНТ приводит к еще большему снижению прочности. Температура при растяжении оказывает влияние на относительное удлинение и может вызвать разрыв УНТ, что приводит к уменьшению относительного удлинения почти на 12 %.

В целом, результаты данного исследования позволяют получить ценные сведения о механических свойствах композиционных материалов, армированных углеродными нанотрубками, что может помочь в разработке новых высокоэффективных материалов для различных областей применения. Значимость исследования заключается в возможности оптимизации свойств материала путем подбора соответствующей плотности и объемной доли углеродных нанотрубок, ориентации армирования, а также контроля температуры при растяжении, что приведет к созданию новых композитных материалов с заданными механическими свойствами.

Литература

1. Brigante D. New Composite Materials Selection, Design, and Application. Springer International Publishing Cham, 2014. 179 p. DOI: 10.1007/978-3-319-01637-5.
2. Abdullahi U., Maleque M.A., Ali M.Y. Hardness behaviour of carbon nanotube-aluminium nanocomposite using nanoindentation technique // Mater. Today Proc. 2019. Vol. 46. P. 6097–6101. DOI: 10.1016/j.matpr.2020.03.333.
3. Rao P.K.V., RaghuKumar B., Veera Sai Chandh Y., Teja A. Finite element analysis of CNT reinforced aluminium composite subjected to mechanical loading // Mater. Today Proc. 2019. Vol. 16. P. 308–313. DOI: 10.1016/j.matpr.2019.05.095.

4. Mansoor M., Shahid M. Fractographic evaluation of crack initiation and growth in Al-CNTs nanocomposite fabricated by induction melting // *Acta Phys. Pol. A*. 2015. Vol. 128. P. 276–278. DOI: 10.12693/APhysPolA.128.B-276.
5. Wang M., Yang X., Tao J. et al. Achieving high ductility in layered carbon nanotube/copper composite prepared by composite electrodeposition // *Diam. Relat. Mater.* 2020. Vol. 108. P. 107992. DOI: 10.1016/j.diamond.2020.107992.
6. Lee W., Jang S., Kim M.J., Myoung J.M. Interfacial interactions and dispersion relations in carbon-aluminum nanocomposite systems // *Nanotechnology*. 2008. Vol. 19. P. 285701. DOI: 10.1088/0957-4484/19/28/285701.
7. Yazdandoost F., Boroujeni A.Y., Mirzaeifa R. Nanocrystalline nickel-graphene nanoplatelets composite: Superior mechanical properties and mechanics of properties enhancement at the atomistic level // *Phys. Rev. Mat.* 2017. Vol. 1. P. 076001. DOI: 10.1103/PhysRevMaterials.1.076001.
8. Roman R., Cranford S. Strength and Toughness of Graphdiyne/Copper Nanocomposites // *Adv. Eng. Mater.* 2014. Vol. 16. P. 862–871. DOI: 10.1002/adem.201400160.
9. Song H.Y., Zha X.W. Influence of nickel coating on the interfacial bonding characteristics of carbon nanotube-aluminum composites // *Comput. Mater Sci.* 2010. Vol. 49. P. 899–903. DOI: 10.1016/j.commatsci.2010.06.044.
10. Silvestre N., Faria B., Canongia Lopes J.N. Compressive behavior of CNT-reinforced aluminum composites using molecular dynamics // *Compos. Sci. Technol.* 2014. Vol. 90. P. 16–24. DOI: 10.1016/j.compscitech.2013.09.027.
11. Hirel P. Atomsk: A tool for manipulating and converting atomic data files Pierre Hirel // *Comput. Phys. Comm.* 2015. Vol. 197. P. 212–219. DOI: 10.1016/j.cpc.2015.07.012.
12. LAMMPS: Molecular Dynamics Simulator. URL: <http://lammps.sandia.gov/> (дата обращения: 17.08.2022).
13. Shelepev I.A., Bayazitov A.M., Korznikova E.A. Modeling of supersonic crowdion clusters in FCC lattice: Effect of the interatomic potential // *J. Micromechanics Mol. Phys.* 2021. Vol. 6. P. 2050019. DOI: 10.1142/S2424913020500198.
14. Янковская У.И., Захаров П.В. Устойчивость к нагреву кристалла платины, армированного углеродными нанотрубками // *Materials. Technologies. Design*. 2021. Т. 3, № 6. С. 64–67. DOI: 10.54708/26587572_2021_34664.
15. Stukowski A. Computational analysis methods in atomistic modeling of crystals // *J. Min. Met. Mat. Soc.* 2013. Vol. 66. P. 399–407. DOI: 10.1007/s11837-013-0827-5.
16. Yoon E.J., Stelson A.C., Orloff N.D. et al. The Effect of Annealing Thin Film Parylene C-Platinum Interfaces Characterized by Broadband Dielectric Spectroscopy // *Proceedings of the 21st International Conference on Solid-State Sensors, Actuators and Microsystems, Transducers 2021, Orlando, FL, USA, 20-24 June 2021*. P. 884–887. DOI: 10.1109/Transducers50396.2021.9495440.
17. Tiwari S.K., Singh H., Midathada A. et al. Study of Fabrication Processes and Properties of Al-CNT Composites Reinforced by Carbon Nano tubes-A Review // *Mater. Today Proc.* 2018. Vol. 5. P. 28262–28270. DOI: 10.1016/j.matpr.2018.10.109.

Особенности выбора параметров для нейронной сети Бехлера—Парринелло на примере кристалла лития

А.Г. Потапов, А.В. Романов

Воронежский государственный университет

С момента публикации в 2011 году статьи Й. Бехлера о симметричных атомно-центрированных функциях для нейронных сетей было предпринято несколько успешных попыток разработать на их основе программы для поиска машинно-обучаемых межатомных потенциалов. Тем не менее, несмотря на очевидный прогресс в этой области, все еще существует проблема выбора параметров к функциям Бехлера—Парринелло для описания конкретных систем. В статье обсуждаются особенности подбора этих параметров на примере кристалла лития. Для тренировки нейросети был выполнен отжиг квантовой молекулярной динамикой с последующим обучением на полученных результатах.

Ключевые слова: нейросеть, межатомный потенциал, молекулярная динамика, функции Бехлера-Парринелло.

1. Введение

Являясь мощным инструментом теоретической химии, метод классической молекулярной динамики, тем не менее, имеет существенный недостаток. Для его использования необходимо знать межатомный потенциал, конструирование которого представляет известную сложность и требует больших временных затрат. С другой стороны, квантовая молекулярная динамика [1, 2] не имеет таких трудностей, однако необходимость больших вычислительных ресурсов делает этот метод малоприменимым для исследования больших структур.

Долгое время связующим звеном между этими двумя методиками было построение межатомных потенциалов по результатам квантово-химической модели. Однако с развитием аппаратных средств и программного обеспечения для нейронных сетей [3, 4] стало возможным «перекинуть мост» от квантовой молекулярной динамики к классической без значительных сложностей.

Важной вехой во внедрении нейронных сетей в молекулярную динамику стоит считать разработку специальных функций, описывающих ближнее окружение конкретного атома, например, полиномов Чебышева [5, 6] или функций Бехлера—Парринелло [7, 8]. Они позволили перейти от декартовой системы координат к описанию системы, допускающей трансляционную симметрию и, как следствие, неограниченное масштабирование. В последние несколько лет появился ряд программных пакетов, позволяющих обучать нейронные сети по результатам квантово-химической молекулярной динамики с использованием как центральных процессоров, так и связки процессора с графическими сопроцессорами. Одним из таких пакетов является нейронная сеть Fortnet [9].

Тем не менее, неразрешенными остаются сложности с подбором входных параметров для обучения нейронной сети. Выяснилось, что ожидаемый результат может быть получен за приемлемое время при наличии качественных исходных данных, однако эффективность обучения нейросети напрямую зависит от количества нейронов в скрытых слоях, вида и числа симметричных функций, их собственных параметров, а также от радиуса отсечки, сужающего диапазон рассматриваемых атомов в окружении. Недостаточное число нейронов приводит к невозможности точной аппроксимации даже при точно подобранных функциях, в то время как ширина и сдвиг центра гауссиан, угловое разрешение и другие дополнительные параметры, влияющие на эффективность их использования, также нуждаются в вычислении. Следует отметить, что в современных пакетных реализациях нейронных сетей, как правило, предложены собственные способы их расчета, однако полученные формулы на данный момент выводятся эмпирическим путем.

2. Теоретическая часть

Перед тем, как использовать функции Бехлера—Парринелло на реальных задачах, необходимо ознакомиться с их формулами и базовыми свойствами. Их проверка на простых тестовых примерах [10] показала наличие важных особенностей, связанных, в том числе, с порядком получаемой величины, которые нужно будет учитывать при дальнейшей работе.

2.1 Функции Бехлера—Парринелло

Ниже в (2)–(6) приведено математическое описание каждой из симметричных функций. Формула (1) содержит функцию отсечки, используемую для определения атомов, входящих в атомное окружение заданного атома.

$$f_c(R_{ij}) = \begin{cases} 0,5 \cdot \left[\cos\left(\frac{\pi R_{ij}}{R_c}\right) + 1 \right] & \text{для } R_{ij} \leq R_c \\ 0 & \text{для } R_{ij} > R_c \end{cases} \quad (1)$$

$$G_i^1 = \sum_j f_c(R_{ij}) \quad (2)$$

$$G_i^2 = \sum_j e^{-\eta(R_{ij}-R_s)^2} \cdot f_c(R_{ij}) \quad (3)$$

$$G_i^3 = \sum_j \cos(\kappa R_{ij}) \cdot f_c(R_{ij}) \quad (4)$$

$$G_i^4 = 2^{1-\zeta} \sum_{j,k \neq i}^{all} (1 + \lambda \cos \theta_{ijk})^\zeta e^{-\eta(R_{ij}^2 + R_{ik}^2 + R_{jk}^2)} f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk}) \quad (5)$$

$$G_i^5 = 2^{1-\zeta} \sum_{j,k \neq i}^{all} (1 + \lambda \cos \theta_{ijk})^\zeta \cdot e^{-\eta(R_{ij}^2 + R_{ik}^2)} \cdot f_c(R_{ij}) \cdot f_c(R_{ik}) \quad (6)$$

Здесь η – ширина гауссиан, регулирующая эффективное радиальное расширение функций симметрии; R_s – сдвиг центра гауссиан на определенный радиус; κ – параметр, определяющий период функции косинуса; θ – угол между всеми комбинациями из трех атомов, содержащих заданный атом; λ – параметр, отвечающий за ориентацию графика $G(\theta)$; ζ – параметр, регулирующий угловое разрешение, то есть плавность изменения значения функции G .

Функции G^1 , G^2 , G^3 относятся к группе радиальных и учитывают только взаимные меж-атомные расстояния. Функции G^4 , G^5 являются угловыми и учитывают более сложные изменения атомных конфигураций. Поскольку функции внутри одной группы обладают довольно схожими свойствами, в сети Fortnet по умолчанию используются только две из них, относящихся к разным категориям: G^2 и G^5 .

2.2 Необходимость нормализации

Для предварительного анализа свойств симметричных функций было осуществлено моделирование химической системы, состоящей из 4 атомов меди, расположенных в ГЦК-решетке, методом молекулярной динамики. Атомы были нагреты до 733 К. Предполагалось, что относительно высокое число степеней свободы в движущейся системе позволит наглядно продемонстрировать влияние не только радиальных, но и угловых функций на возможность аппроксимации ее энергии, график изменения которой с течением времени представлен на рис. 1.

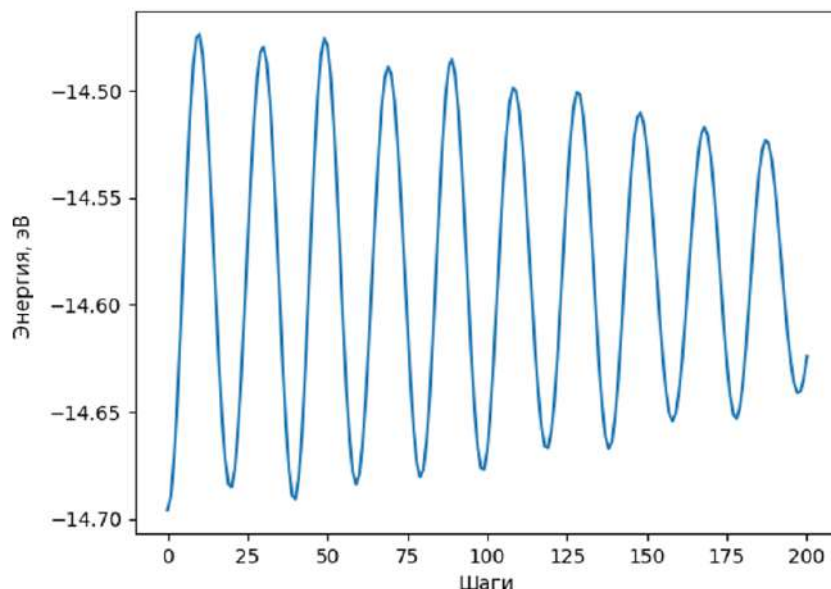


Рис. 1. Колебания энергии в ходе эксперимента с Si_4

Для тестирования зависимости энергии от функций Бехлера—Парринелло были рассчитаны все возможные их линейные комбинации с единичными параметрами. Поскольку в данной системе невозможно выделить единственный неподвижный атом, относительно которого было бы удобнее всего рассматривать изменение всей конфигурации, было решено использовать сумму значений функций для каждого из 4 атомов. Некоторые из полученных графиков зависимости энергии системы от симметричных функций $E(G)$ представлены на рис. 2.

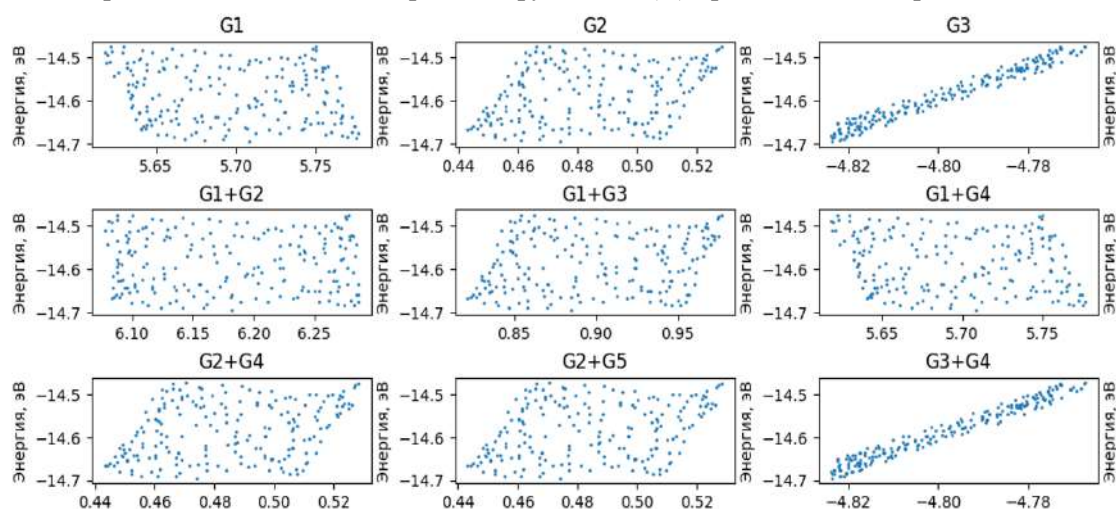


Рис. 2. Графики $E(G)$ для некоторых возможных линейных комбинаций G^1 – G^5

На рисунке видно, что некоторые из полученных графиков почти полностью совпадают, из чего был сделан вывод о наличии внутри симметричных функций определенной иерархии порядка: модули значений некоторых из них существенно превышают модули значений других ($|G^1| > |G^3| > |G^2| \gg |G^5| > |G^4|$), что приводит к несбалансированному их влиянию на конечный результат. Следовательно, необходимо использовать нормализацию при дальнейшей работе для компенсации данного эффекта.

Кроме того, наименьший разброс демонстрируют G^3 -подобные графики, что может быть связано с наличием в ее структуре формулы косинуса, хорошо описывающей затухающие периодические колебания энергии.

3. Практическая часть

В ходе исследования было проведено моделирование кристалла лития с ячейкой на 27 атомов в рамках квантовой молекулярной динамики. С помощью калькулятора ASE[11] / GPAW [12–13] были рассчитаны траектории движения атомов моделируемых систем, а также получены их потенциальная и полная энергии в каждый момент времени. На основании полученных данных было проведено обучение нейросети Fortnet с различными гиперпараметрами, в числе которых количество радиальных и угловых симметричных функций, число нейронов в скрытых слоях и радиус отсечки. Кроме того, были сопоставлены полученные значения G и полная энергия $E(G)$.

3.1 Величина датасета

В первом эксперименте было взято 27 атомов лития, был выполнен молекулярно-динамический отжиг с использованием динамики Langevin. В качестве начального распределения атомов по скоростям было выбрано распределение Максвелла-Больцмана при температуре в 300 К. Сходимость по температуре достигнута спустя 35000 шагов.

Полный объем полученных данных оказался не слишком удобным для первоначального тестирования влияния входных параметров модели Fortnet по причине высокой ресурсоемкости, в связи с чем были взяты лишь 372 точки расчета с шагом в 64 точки (для охвата значительной доли области анализа). Было использовано два скрытых слоя BPNN с 2 нейронами в каждом из них, 10 радиальных и 10 угловых симметричных функций. Радиус отсечки составил 20 Å (достаточно, чтобы захватить все 27 атомов). Обучение происходило в течение 5000 итераций. Сравнение исходной (DFT) и предсказанной (NN) энергий представлено на рис. 3.

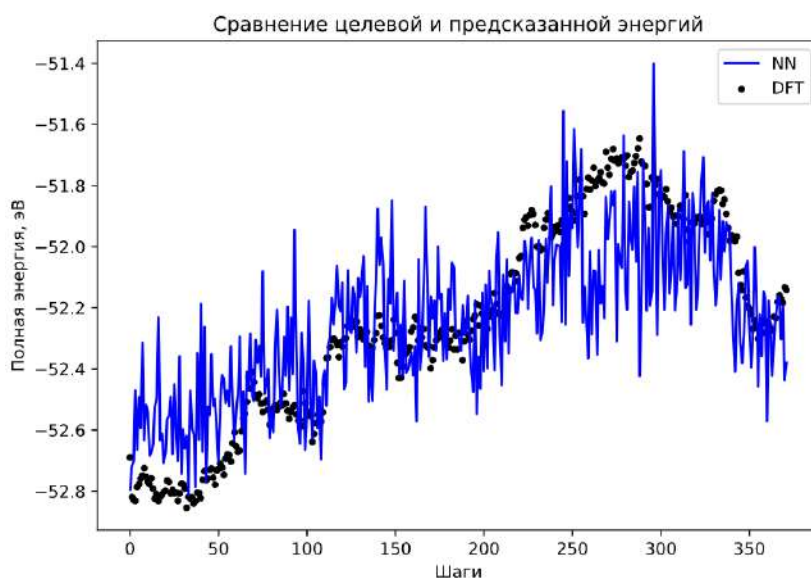


Рис. 3. Результаты работы сети Fortnet для 372 точек и 20 симметричных функций

Как видно, точность предсказания полученной модели крайне низка, она описывает лишь общую тенденцию изменения энергии и демонстрирует слишком большой процент выбросов. На рис. 4 представлен график изменения энергии в зависимости от суммы полученных значений функций Бехлера—Парринелло. Видно, что одним и тем же значениям G соответствует слишком широкий спектр значений E , что может свидетельствовать о том, что данный набор входных параметров не подходит для описания заданной системы.

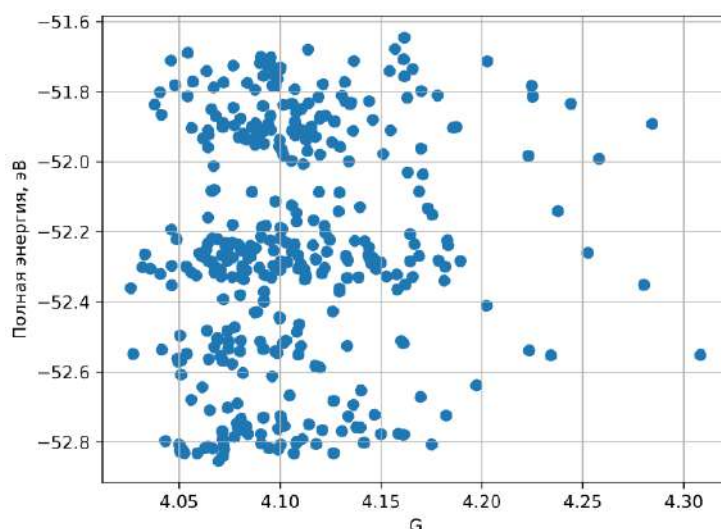


Рис. 4. График $E(G)$ для 372 точек и 20 симметричных функций

На втором этапе тестирования было взято большее количество шагов для предыдущего расчета, были внесены изменения в способ формирования входных данных: вместо охвата всего временного отрезка были взяты лишь первые 2000 значений подряд, что позволило работать с данными, скорость изменения которых приближена к реальной. Было существенно увеличено число нейронов в скрытых слоях: по 27 в каждом из двух слоев BPNN. Кроме того, результаты предыдущего эксперимента дали основание полагать, что число симметричных функций также нуждается в увеличении, поэтому для новой модели было взято 25 радиальных и 25 угловых функций. Для обучения оказалось достаточно 4283 итерации, после чего средняя ошибка MSE составила порядка $5,95 \cdot 10^{-6}$, что меньше целевого значения. Время обучения на 12 ядрах процессора Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz составило 12,2 минут. Итоговый график сравнения исходной и предсказанной энергий можно видеть на рис. 5.

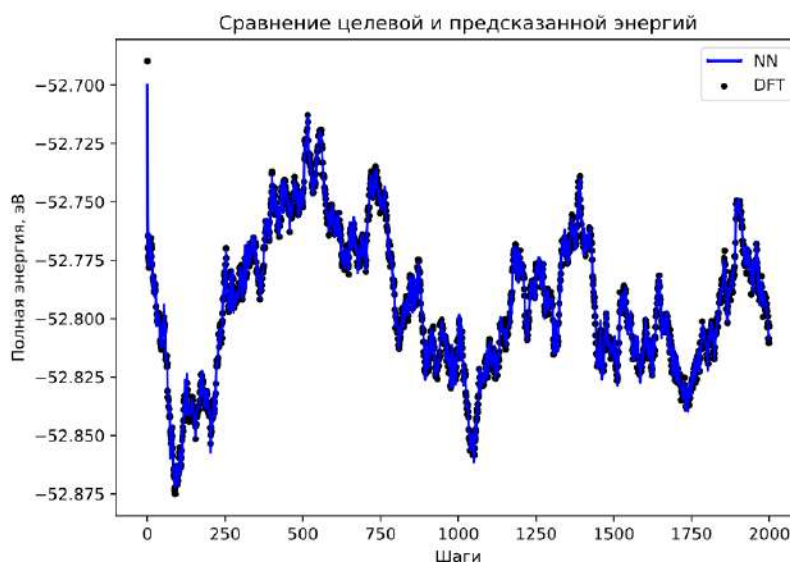


Рис. 5. Результаты работы сети Fortnet для 2000 точек и 50 симметричных функций

На графике видно, что первая точка сильно выделяется из общего диапазона. Она связана с начальным приближением и не имеет значения для последующих расчетов. Кроме того, особое внимание стоит уделить графику $E(G)$, изображенному на рис. 6. Видно, что для одного и того же значения суммы значений всех 50 функций Бехлера—Парринелло относительно нулевого атома степень изменчивости энергии заметно меньше, чем на рис. 4. Тем не менее, она все еще гораздо больше, чем можно наблюдать на рис. 5, что было бы невозможно при использовании

сетью Fortnet линейной комбинации значений G для аппроксимации энергии. Следовательно, полученные 50 значений следует рассматривать как отдельные переменные, которые нельзя информативно изобразить на двумерном графике.

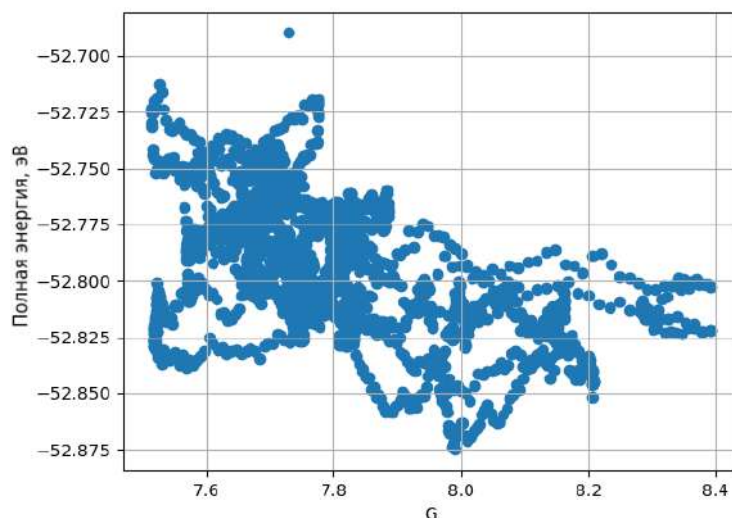


Рис. 6. График $E(G)$ для 2000 точек и 50 симметричных функций

3.2 Влияние радиуса отсечки

Изменение задаваемого радиуса отсечки при сохранении величины датасета, числа нейронов в скрытых слоях и самих симметричных функций способно оказать видимое влияние на результат. С одной стороны, его уменьшение способно упростить формат входных данных и ускорить обучение (450 сек. для $R_c = 8 \text{ \AA}$ против 715 сек. для $R_c = 20 \text{ \AA}$ для 12 нейронов в каждом из 2 скрытых слоев). Однако при детальном рассмотрении можно заметить появление лишних всплесков на графике предсказанной энергии, что свидетельствует о недостаточной точности обучения (ср. рис. 5 и рис. 7).

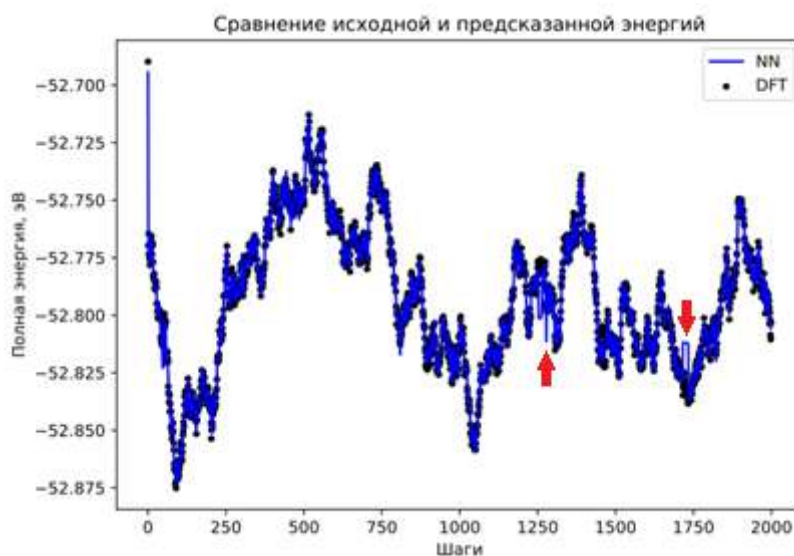


Рис. 7. Результаты работы сети Fortnet для 27 и 27 нейронов, 50 функций, $R_c = 8 \text{ \AA}$

3.3 Вид симметричных функций

Ручное задание разнообразных сочетаний используемых симметричных функций при их общем неизменном числе, равном 24, и радиусе отсечки $R_c = 8 \text{ \AA}$ позволило сделать некоторые выводы касательно их собственной эффективности. Были рассмотрены комбинации $12G^1 + 12G^4$,

$12G^1+12G^5$, $12G^2+12G^4$, $12G^2+12G^5$, $6G^1+6G^2+12G^4$, $6G^1+6G^2+12G^5$, $12G^1+6G^4+6G^5$, $12G^2+6G^4+6G^5$, $6G^1+6G^2+6G^4+6G^5$. Наименее удачной стала комбинация $12G^1+12G^4$ (см. рис. 8). Наиболее эффективной — комбинация $12G^2+6G^4+6G^5$ (см. рис. 9).

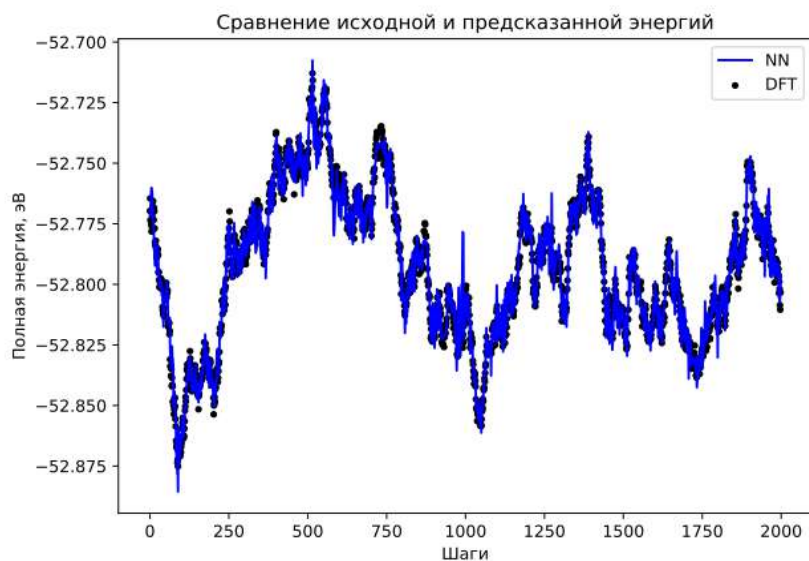


Рис. 8. Результаты работы сети Fortnet для $12G^1+12G^4$, $R_c = 8 \text{ \AA}$

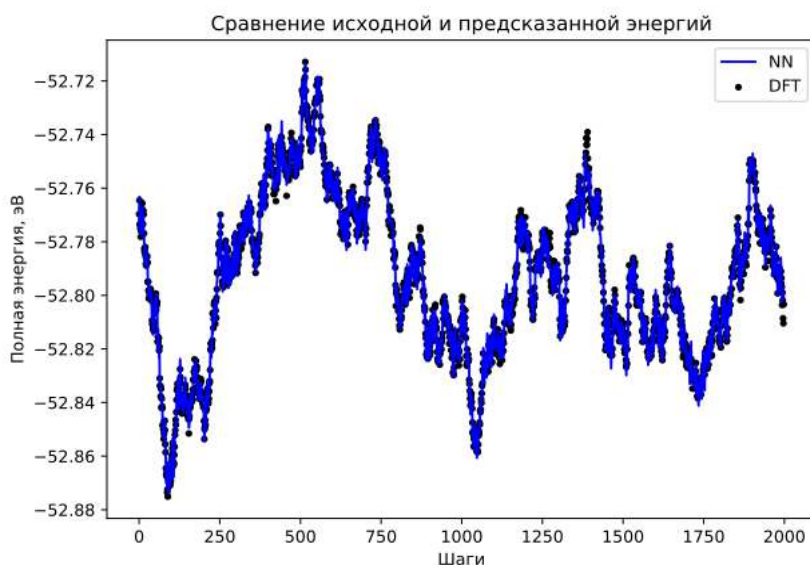


Рис. 9. Результаты работы сети Fortnet для $12G^2+6G^4+6G^5$, $R_c = 8 \text{ \AA}$

3.4 Число симметричных функций

Помимо описанных ранее моделей были опробованы другие варианты изменения гиперпараметров с целью выявления наиболее эффективных сочетаний. В частности, уменьшение числа симметричных функций с 50 до 12 привело, с одной стороны, к значительному замедлению обучения (потребовалось 16507 итераций и 32,9 минут), но, с другой стороны, несколько снизило процент шумовых осцилляций, что могло положительно сказаться на обобщающей способности модели. Результирующий график можно наблюдать на рис. 10.

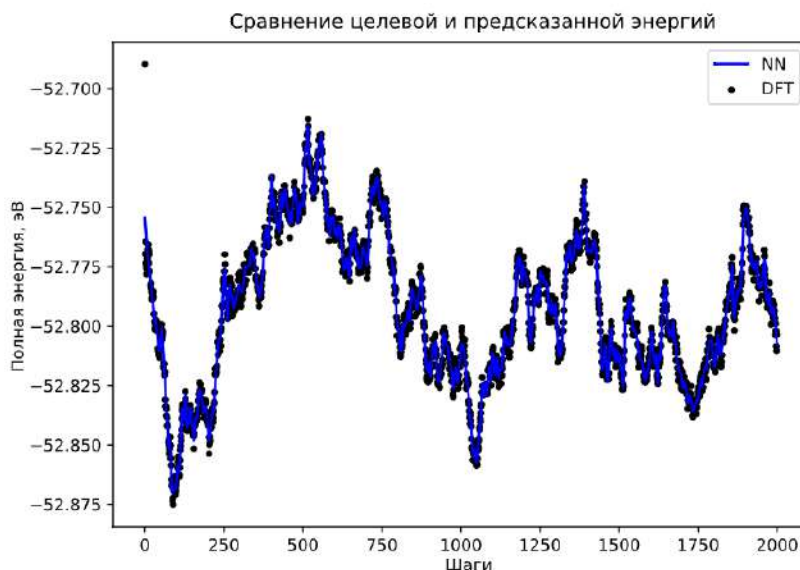


Рис. 10. Результаты работы сети Fortnet для 2000 точек и 12 симметричных функций

Кроме того, попытка уменьшить число нейронов до 13 в каждом из двух скрытых слоев привела к чуть менее точным результатам ($MSE = 1,43 \cdot 10^{-5}$) при практически неизменной скорости обучения (около 11 минут).

4. Заключение

На основе датасета для кристалла лития, полученного методом квантовой молекулярной динамики, показано, что с помощью нейронной сети возможно аппроксимировать значение полной энергии системы в каждый момент времени, основываясь только на данных о взаимных межатомных расстояниях и валентных углах. Переводя декартовы координаты атомов в базисный набор симметричных функций Бехлера—Парринелло, программа учитывает трансляционную и вращательную симметрию получающихся атомных конфигураций, что делает возможным использование результатов обучения нейросетевой модели в классической молекулярной динамике.

Тем не менее, существует набор параметров, оказывающих непосредственное влияние на степень совпадения исходной и предсказанной энергий. В частности, большое значение имеет выбор качественных и достаточно полных исходных данных. Чем более детальным и плавно меняющимся является массив расстояний, углов и энергий, тем более корректными получаются результирующие графики.

Еще одним важным фактором является число нейронов в скрытых слоях нейронной сети. Близкое к числу атомов число нейронов в проведенных экспериментах являлось минимально необходимым, поскольку с его уменьшением точность обучения начинала неизменно падать, а время обучения — расти.

Наконец, количество выбираемых радиальных и угловых симметричных функций напрямую зависит от самой рассматриваемой системы. Чем больше это число, тем более многомерным становится признаковое пространство, подаваемое на вход нейронной сети, задача которой заключается в установлении функциональной зависимости между ним и одномерным целевым вектором энергий. По этой причине становится крайне затруднительно (а при большом числе функций невозможно) оценить возможность нахождения такой связи, исходя из графического отображения зависимости $E(G)$. Увеличивается число параметров нейронной сети, а вместе с ним и вероятность переобучения.

В то же время работа с относительно небольшим числом симметричных функций может быть более эффективной в плане перспективного применения нейросети для оценки новых данных, но занимать больше времени, поскольку гибкость процесса обучения, связанная с быстрой и масштабной коррекцией весовых коэффициентов модели, снижается. Так или иначе,

получаемые модели способны выдать корректные результаты для самых разных входных параметров, причем лучше всего показало себя сочетание функций $G^2+G^4+G^5$.

В дальнейшем планируется расширить набор исследуемых систем, чтобы выявить новые закономерности в использовании различных параметров нейронной сети и функций Бехлер—Парринелло.

Литература

1. Marx D., Hutter J. Ab initio Molecular Dynamics: Basic Theory and Advanced Methods // Cambridge University Press. 2009. Vol. 63, no. 3. 567 p. DOI: 10.1017/CBO9780511609633.
2. Kirchner B., di Dio P.J., Hutter J. Real-World Predictions from Ab Initio Molecular Dynamics Simulations // Topics in Current Chemistry. 2011. Vol. 307. P. 109–153. DOI: 10.1007/128_2011_195.
3. Kyuhyun L., Dongsun Y., Wonseok J. et al. SIMPLE-NN: An efficient package for training and executing neural-network interatomic potentials // Computer Physics Communications. 2019. Vol. 242. P. 95–103. DOI: 10.1016/j.cpc.2019.04.014.
4. Blank T.B., Brown S.D., Calhoun A.W. et al. Neural network models of potential energy surfaces // The Journal of Chemical Physics. 1995. Vol. 103, no. 10. P. 4129–4137. DOI: 10.1063/1.469597.
5. Lindsey R.K., Fried L.E., Goldman N. ChIMES: A force matched potential with explicit three body interactions for molten carbon // Journal of Chemical Theory and Computation. 2017. Vol. 13, no. 12. 8 p. DOI: 10.1021/acs.jctc.7b00867.
6. Lindsey R.K., Fried L.E., Goldman N. et al. Active learning for robust, high-complexity reactive atomistic simulations // The Journal of Chemical Physics. 2020. Vol. 153, no. 13. 16 p. DOI: 10.1063/5.0021965.
7. Behler J. Constructing High-Dimensional Neural Network Potentials: A Tutorial Review // International Journal of Quantum Chemistry. 2015. Vol. 115, no. 16. 19 p. DOI: 10.1002/qua.24890.
8. Behler J. Atom-centered symmetry functions for constructing high-dimensional neural network potentials // The Journal of Chemical Physics. 2011. Vol. 134, no. 3. 14 p. DOI: 10.1063/1.3553717.
9. van der Heide T., Kullgren J., Broqvist P. et al. Fortnet, a software package for training Behler Parrinello neural networks // Computer Physics Communications. 2023. Vol. 284. 13 p. DOI: 10.5281/zenodo.5969907.
10. Потапов А.Г., Романов А.В. Моделирование межатомного потенциала по результатам квантово-химического расчета с использованием симметричных функций Бехлера-Парринелло // Труды молодых ученых факультета компьютерных наук ВГУ. 2024. № 4. С. 215–222.
11. Larsen A.H., Mortensen J.J., Blomqvist J. et al. The Atomic Simulation Environment – A Python library for working with atoms // J. Phys.: Condens. 2017. Vol. 29, no. 27. 31 p. DOI: 10.1088/1361-648X/aa680e.
12. Mortensen J.J., Larsen A.H., Kuisma M. et al. GPAW: An open Python package for electronic structure calculations featured // J. Chem. Phys. 2024. Vol. 160, no. 9. 42 p. DOI: 10.1063/5.0182685.
13. Mortensen J.J., Hansen L.B., Jacobsen K.W. Real-space grid implementation of the projector augmented wave method // Phys. Rev. B. 2005. Vol. 71, no. 3. 11 p. DOI: 10.1103/PhysRevB.71.035109.

Квантово-механический расчет термодинамических свойств гексагонального и кубического рутения

А.А. Резник^{1,2}, С.А. Горохов^{1,2}, А.А. Резванов^{1,2}

¹АО «Научно-исследовательский институт молекулярной электроники»,
²ФГАОУ ВО «Московский физико-технический институт

(национальный исследовательский университет)»

В работе представлены результаты теоретического расчета структурных и термодинамических свойств гексагонального и кубического рутения. Применен квантово-механический подход с использованием метода конечных смещений и теории возмущений функционала плотности. Рассмотрен оптимизированный для графических ускорителей алгоритм диагонализации гамильтониана Кона—Шэма. С использованием обменно-корреляционного приближения Wu-Cohen достигнуты наиболее близкие к экспериментальным значениям физические свойства гексагонального рутения, а именно: параметры решетки, коэффициенты линейного и объемного теплового расширения, изобарная теплоемкость. Эти же свойства рассчитаны и для кубического рутения.

Ключевые слова: рутений, теория функционала плотности, квазигармоническая аппроксимация, температура, параметр решетки, теплоемкость.

1. Введение

Межсоединения в интегральных микросхемах представляют собой тонкие металлические дорожки, по которым передается электрический сигнал между электрическими компонентами микросхемы. Важнейшей характеристикой межсоединений является скорость передачи сигнала, которая зависит от собственных сопротивлений и емкостей компонентов микросхемы (RC-задержка) [1]. Задержки начинают превалировать над временем переключения транзисторов при технологических нормах меньше 180 нанометров (нм), что приводит к потере выигрыша производительности с дальнейшим масштабированием [2].

Долгое время основным материалом для межсоединений является алюминий (Al), но с уменьшением проектных норм и переходу к трехмерной интеграции с несколькими уложенными активными слоями, проблема RC-задержек и, соответственно, межсоединений стала еще более значимой. С точки зрения материаловедения, медь (Cu), по сравнению с алюминием, имеет значительно меньшее удельное сопротивление (что подразумевает меньшее рассеивание мощности и меньшие RC-задержки) и более высокую устойчивость к электромиграции. Более того, RC-задержка и рассеиваемая мощность могут быть дополнительно уменьшены за счет использования слоев диэлектриков с низкой диэлектрической проницаемостью (low- κ) [3, 4]. Однако внедрение медных межсоединений, начиная с технологических норм 90 нм, сопровождается серьезной проблемой: атомы меди быстро диффундируют в кремний или в диэлектрик между проводящими дорожками, и по этой причине обычно требуется диффузионный барьерный слой, чтобы избежать эффекта электрической утечки в диэлектрике. Барьер должен быть максимально тонким, но достаточно толстым, чтобы противостоять диффузии меди. Наличие барьеров делает ситуацию с RC-задержками еще более сложной из-за роста удельного сопротивления. Другой проблемой является резкий рост удельного сопротивления меди при ширине дорожки менее 30 нм. Снизить рост удельного сопротивления возможно за счет увеличения высоты проводников, однако, при этом возникают технологические сложности, связанные с травлением диэлектрика и химико-механической полировкой. Таким образом, требуется производить поиск альтернативных материалов, которые позволят добиться оптимальных характеристик диффузионного барьера или вовсе отказаться от меди при формировании межсоединений.

Начиная с 2000-х годов в качестве потенциального материала для создания барьерных слоев для технологических норм меньше 28 нм исследуется рутений (Ru) и соединения на его основе. Этот твердый металл, характеризующийся высокой химической инертностью и темно-серым

цветом, используется во многих сферах, в т.ч. и электронике. Рутений обладает некоторыми из основных желаемых характеристик для барьерных слоев: отсутствие интерметаллидов на фазовой диаграмме Cu-Ru и хорошей адгезией к Cu. Последнее свойство позволяет осуществлять прямое электроосаждение Cu, избегая необходимости использования затравочного слоя. По этим причинам он широко исследуется как потенциальный диффузионный барьер в технологии Cu/low-к. Вместе с другими материалами, такими как сплавы кобальта (Co), рутений является перспективным кандидатом на замену современных технологий диффузионных барьеров [5].

Рутений и молибден (Mo) могут стать подходящими материалами для межсоединений с технологией 28 нм из-за их различных преимуществ по сравнению с медью: они обладают более низким удельным сопротивлением, чем Cu, при ширине проводника менее 10 нм, не диффундируют в SiO₂ и поддаются травлению. Недавние исследования по оценке электромиграции, а также электромагнитных свойства с точки зрения параметров, повышающих удельное сопротивление, которые зависят от геометрии и дополнительных технологических процессов изготовления, подтверждают целесообразность использования Ru в качестве многообещающего кандидата для применения в системах межсоединений следующего поколения, обеспечивающих повышенную надежность по сравнению с межсоединениями на основе Cu [6]. При переходе к технологическим нормам суб-10 нм в качестве материалов межсоединений рассматриваются бинарные соединения (например, NiSi, CoSi₂, Cu₃Ge), тройные соединения (TiSiC, TiAlN) или двумерные материалы (графен).

Наиболее стабильной (при нормальных условиях) и экспериментально наблюдаемой кристаллографической модификацией рутения является плотноупакованная гексагональная с пространственной группой P6₃/mmc (№194). Экспериментальное определение фононных дисперсий и термодинамических свойств гексагонального рутения, как и находящегося с ним в одной группе осмия (Os) затруднено. Для Os с помощью рамановской спектроскопии было измерено лишь несколько фононных частот [7], а для Ru фононные дисперсии были получены с помощью неупругого рассеяния нейтронов [8]. В другой работе также сообщается об экспериментальных фононных частотах вдоль направления [001] [9]. Фононные спектры Ru были теоретически исследованы в работе [10], причем авторы применили квазигармонический подход, включающий оценку теплового расширения, аналогично работе [11]. Имеющиеся термодинамические и кристаллографические свойства Ru приведены в работах [12–14]. Стоит также отметить, что экспериментально был получен метастабильный объемноцентрированный гексагональный рутений, который демонстрировал намагниченность при низкой (10K) и комнатной температуре [15]. Квантово-механические расчеты некоторых предсказанных кристаллографических фаз рутения представлены в работе [16].

Все предыдущие теоретические квазигармонические результаты рассматривают гексагональный рутений и опираются на ультрамягкие псевдопотенциалы (англ. *Ultrasoft pseudopotential*, USPP) или проекционные присоединенные псевдопотенциалы (англ. *Projector augmented wave*, PAW), и рассчитываются только на основе фононных частот. В этой же работе, используя теорию функционала электронной плотности, рассматриваются структурные и термодинамические свойства гексагонального и кубического рутения. Термодинамические параметры кристаллов получены с использованием квазигармонической аппроксимации (англ. *Quasi-harmonic approximation*, QHA) как с использованием подхода конечного смещения, так и с точным вычислением силовых констант.

2. Теория и методы расчета

2.1 Параллельные вычисления

Обычно, моделируемые системы представляют собой большие ячейки с большим количеством атомов, для которых время, затрачиваемое на вычисления на графических ускорителях (GPU), превышает время, необходимое для передачи данных с процессора (CPU) на графический ускоритель. Высокосимметричные ячейки, например, кубические, состоящие из одного сорта атомов, обладают малым количеством атомов (1–4 шт.), что накладывает требование использовать густую сетку k-точек для дискретизации поверхности Ферми. Расчет фононных дисперсий таких систем с множеством геометрий для термодинамических расчетов, является проблемой,

для решения которой можно было бы использовать преимущества новых суперкомпьютеров, но было обнаружено, что использование GPU с существующими кодами не всегда удобно, а иногда может и замедлить расчеты по сравнению с использованием только CPU.

Quantum ESPRESSO (QE) имеет несколько уровней распараллеливания на центральном процессоре. Можно разделить k -точки на группы (называемые пулами) и назначить каждой группе набор ядер. Пулы могут быть разделены на группы, каждая из которых работает с набором полос (распараллеливание полос), и, наконец, каждая группа ядер, работающая с набором полос, может быть разделена по векторам обратной решетки (G) и работать только с их подмножеством (распараллеливание G -векторов). Именно на этом этапе можно ввести ускорение с помощью GPU. Стандартный метод использования GPU заключается в выделении переменных в памяти устройства и вызове процедур, разработанных Nvidia, которые выполняют операции линейной алгебры (cuBLAS) и быстрого преобразования Фурье (cuFFT) над данными находящимся в памяти.

Основной вычислительной задачей является решение уравнений Кона—Шэма (англ. *Kohn-Sham equations*, KS), которые минимизируют полную энергию системы. Эти уравнения представляют собой задачу на собственные значения для псевдопотенциалов, сохраняющих норму, и обобщенную задачу на собственные значения для ультрамягких или проекционных присоединенных псевдопотенциалов. Стандартный алгоритм Дэвидсона последовательно работает с каждой k -точкой пула и решает уравнения Кона—Шэма вызывая процедуры, использующие гамильтониан Кона—Шэма H_{KS} и матрицу перекрытий S . Поэтому загрузка памяти GPU увеличивается линейно с ростом числа k -точек. Когда размер задачи невелик, может случиться так, что библиотечные процедуры матричных умножений и быстрых преобразований Фурье не смогут использовать всю мощность GPU, поскольку у них слишком мало данных для работы. В результате вычисления на GPU могут стать даже медленнее, чем вычисления только на CPU. Распараллеливание на G -векторах только еще больше уменьшает размер данных, выделяемых на GPU для каждой k -точки, и не помогает в данном случае.

Подход к ускорению кода на графическом процессоре, недавно предложенный в работе [20] представлен на рис. 1. В память GPU помещаем как можно больше волновых функций (т.е. k -точек) сколько возможно в блоке из N_k k -точек и одновременно выполняем над всеми этими k -точками операции алгоритма Дэвидсона, необходимые для диагонализации гамильтониана. Каждый пул процессорных ядер работает со своим набором из k -точек, назначенных при распараллеливании пула в QE, и только эти k -точек разделяются на блоки для ускорения GPU. Распараллеливание полос и распараллеливание G -векторов в настоящее время не поддерживается данным подходом. Основная оптимизация графического процессора была выполнена в процедуре, которая применяет H_{KS} и S к волновым функциям ψ_{kv} , но также было получено некоторое ускорение при параллельном выполнении операций алгоритма Дэвидсона во многих k -точках. Такой подход позволил получить ускорение расчетов до 5 раз по сравнению со стандартной диагонализацией.

Алгоритм 1. Стандартная диагонализация для CPU и GPU

```
for ik = 1, nks do                                ► nks = #k points per pool
    build  $H_{KS}^k$  and  $S^k$ 
    compute  $\epsilon_{kv}$  and  $\psi_{kv}$  by Davidson ( $N_b$ )
end for
```

Алгоритм 2. Оптимизированная GPU диагонализация

```
for ikb = 1, nkblock do                            ► nkblock = #k points blocks
    build in parallel  $H_{KS}^k$  and  $S^k$  on GPU threads ( $N_k$ )
    compute  $\epsilon_{kv}$  and  $\psi_{kv}$  by Davidson on GPU threads ( $N_b \times N_k$ )
end for
```

Рис. 1. Алгоритмы, используемые в стандартном подходе и в оптимизированном под GPU для диагонализации гамильтониана [20]

Рассматриваемые в данной работе примитивные ячейки рутения являются металлами с малым количеством атомов, а значит предложенный алгоритм ускорения операции диагонализации должен ускорить проведение расчетов. Типичной размерностью k -сетки, рекомендуемой авторами кода, является $40 \times 40 \times 40$ или более. Такая высокая размерность потребует сотни гигабайт памяти, что накладывает соответствующие требования на объем оперативной памяти и видеопамати.

2.2 Подходы к расчету фононов в твердых телах

Тепловые свойства твердых тел при постоянном объеме могут быть рассчитаны из плотности состояний фононов как функции частот. Существует два популярных типа фононных расчетов, основанных на теории функционала плотности (англ. *Density functional theory*, DFT): метод конечных смещений в суперячейке (англ. *finite-displacement supercell approach*, FDSA) и теория возмущений функционала плотности (англ. *Density-functional perturbation theory*, DFPT) [17].

В подходе конечных смещений, расчеты на основе первых принципов используются в качестве источника для получения атомных сил в модели кристаллической структуры суперячейки. Силовые константы рассчитываются из достаточного количества суперячеек с различными наборами смещений и соответствующих сил, полученных в результате первопринципного расчета. Фононы вычисляются точно (т.е. без интерполяции) из силовых констант суперячейки на волновых векторах, соизмеримых с формой суперячейки относительно примитивной ячейки. На других волновых векторах фононы получают как интерполированные результаты. На практике использование суперячейки размером в несколько сотен атомов часто дает приемлемые результаты по фононам путем интерполяции, хотя требуемая точность зависит от целей расчета.

В подходе DFPT силовые константы точно вычисляются во взаимном пространстве при произвольных волновых векторах путем решения вариации орбиталей Кона—Шэма. Опционально силовые константы в прямом пространстве, аналогичные силовым константам суперячеек, вычисляются путем преобразования Фурье из силовых констант во взаимном пространстве. Силовые константы в прямом пространстве снова преобразуются Фурье в константы во взаимном пространстве при произвольных волновых векторах.

Частота фононов меняется под действием деформации. Это ангармонический эффект, который иногда называют квазигармоничностью, хотя в разных контекстах это слово может означать разные явления.

2.3 Квазигармоническая аппроксимация

В рамках QHA свободная энергия Гельмгольца F твердого тела является функцией температуры T и объема (единичной ячейки) V . Она может быть записана следующим образом:

$$F(V, T) = U(V) + F_{vib}(V, T) + F_{el}(V, T), \quad (1)$$

где $U(V)$ – энергия, вычисленная из DFT, $F_{vib}(V, T)$ – вклад колебаний решетки, $F_{el}(V, T)$ – энергия, обусловленная электронными возбуждениями. В адиабатическом приближении каждый член рассматривается отдельно. Для заданного объема вычисляется вклад колебаний на ячейку, как в гармоническом приближении:

$$F_{vib}(V, T) = \frac{1}{2N} \sum_{\mathbf{q}, \nu} \hbar \omega(\mathbf{q}, \nu, V) + \frac{k_B T}{N} \sum_{\mathbf{q}, \nu} \ln \left[1 - \exp \left(\frac{-\hbar \omega(\mathbf{q}, \nu, V)}{k_B T} \right) \right], \quad (2)$$

где первый член в правой части приведенного уравнения – нулевая энергия, второй член – фононный вклад при конечных температурах. Суммы в уравнении (2) берутся по частотам фононов $\omega(\mathbf{q}, \nu, V)$, где ν обозначает различные ветви фононов, а $\mathbf{q}$ – волновые векторы в пределах первой зоны Бриллюэна. k_B – постоянная Больцмана, $\hbar$ – приведенная постоянная Планка, а N – число элементарных ячеек в твердом теле.

Вклад электронных возбуждений F_{el} в свободную энергию рассчитывается как:

$$F_{el} = U_{el} - TS_{el}, \quad (3)$$

где U_{el} – вклад электронного возбуждения в энергию, который определяется следующим образом:

$$U_{el} = \int_{-\infty}^{\infty} EN(E)f(E, T, \mu)dE - \int_{-\infty}^{E_F} EN(E)dE, \quad (4)$$

где E_F – энергия Ферми, $N(E)$ – плотность электронных состояний, $f(E, T, \mu)$ – распределение Ферми-Дирака, μ – химический потенциал. S_{el} – электронная энтропия, определяемая по формуле:

$$S_{el} = -k_B \int_{-\infty}^{\infty} [f(E, T, \mu) \ln f(E, T, \mu) + (1 - f(E, T, \mu)) \ln(1 - f(E, T, \mu))] N(E) dE. \quad (5)$$

Для гексагонального рутения мы провели расчеты в двух случаях: (I) в предположении постоянного отношения сторон ($c/a = \text{const}$), (II) так и с переменным отношением сторон ($a, c/a$).

I. В первом случае вычисляются фононные частоты при 15 различных геометриях (объемах), сохраняя константу c/a равную значению при 0 К и подгоняя полученные энергии уравнением состояния Мурнагана. Минимум $U(V) + F_{vib}(V, T)$ при каждой температуре соответствует нулевому давлению. Таким образом, температурная зависимость свободной энергии F , объема V и изотермического модуля объемной упругости B_T следует непосредственно из уравнения состояния Мурнагана. Путем взятия производной от объема мы получаем коэффициент объемного теплового расширения β :

$$\beta = \frac{1}{V} \left(\frac{\partial V}{\partial T} \right)_P. \quad (6)$$

Теплоемкость при постоянном объеме (изохорная) была получена из фононных частот, рассчитанных как в гармоническом приближении:

$$C_V = \frac{k_B}{N} \sum_{\mathbf{q}, \nu} \left(\frac{\hbar \omega(\mathbf{q}, \nu)}{k_B T} \right)^2 \frac{\exp(\hbar \omega(\mathbf{q}, \nu)/k_B T)}{[\exp(\hbar \omega(\mathbf{q}, \nu)/k_B T) - 1]^2}. \quad (7)$$

в каждом фиксированном объеме в квазигармонической сетке. Изохорная теплоемкость при каждой температуре затем была получена интерполяцией по зависящим от температуры значениям объема, полученным при каждой температуре из минимизации свободной энергии.

Теплоемкость при постоянном давлении (изобарная) была получена как [18]:

$$C_P = C_V + TV\beta^2 B_T. \quad (8)$$

II. В полностью анизотропном случае расчеты проводились на сетке точек (a и c/a), а затем энергия Гельмгольца при каждом значении T подгонялась с помощью полиномов четвертой степени как функция a и c/a . В общей сложности использовалось 25 точек сетки (5×5) с шагом 0.05 боровских радиусов и 0.02 боровских радиусов для a и c/a , соответственно. Сетка была приблизительно центрирована на равновесных значениях a и c/a при 0 К. Температурные зависимости длин сторон a и c получены из минимизации свободной энергии F .

Значения двух коэффициентов линейного теплового расширения (поскольку в данном гексагональном случае только два члена теплового расширения являются независимыми) получены из температурного изменения параметров решетки:

$$\alpha_1 = \alpha_2 = \frac{1}{a(T)} \frac{da(T)}{dT}, \quad (9)$$

$$\alpha_3 = \frac{1}{c(T)} \frac{dc(T)}{dT}. \quad (10)$$

Объемное тепловое расширение β находится как сумма коэффициентов линейного теплового расширения:

$$\beta = 2\alpha_1 + \alpha_3. \quad (11)$$

2.4. Методы расчета

Квантово-механические вычисления в настоящей работе проводятся с использованием теории функционала плотности на основе базиса плоских волн, реализованного в программном пакете QE [19]. Для одновременного и асинхронного выполнения множества самосогласованных и фононных расчетов используется драйвер на языке Fortran Thermo_pw в основе работы которого лежат процедуры QE, но обладающим дополнительным уровнем распараллеливания по сравнению с QE. Фононные расчеты и квазигармоническая аппроксимация на основе DFPT выполняются на кластере ЦКП «Информатика», используя версию Thermo_pw с поддержкой вычислений на графических ускорителях на основе CUDA Fortran, которая оптимизирует некоторые процедуры QE для решения задач с высокой плотностью выборки k -точек у металлических систем [20].

Вычислительные параметры были выбраны таким образом, чтобы обеспечить хорошую сходимость всех исследуемых свойств и, в частности, фононных частот. Во всех самосогласованных расчетах для вычисления волновых функций использовался итеративный метод диагонализации Дэвидсона [21]. Отсечка по кинетической энергии плоских волн была установлена на 2040 электронвольт (эВ), а отсечка по плотности заряда на 8160 эВ. Интегрирование по зоне Бриллюэна проводилось по сетке Монкхорста—Пака [22] $18 \times 18 \times 12$ для гексагонального рутения (h-Ru) и $20 \times 20 \times 20$ для кубического рутения (c-Ru) с использованием схемы размазывания Марзари—Вандербильта [23] величиной 0.136 эВ. Релаксация ячеек проводилась до тех пор, пока давление не станет меньше 0.003 ГПа, а межатомные силы не станут меньше $2,57 \cdot 10^{-3}$ эВ/Å и разница в энергиях основного состояния между последовательными итерациями не станет меньше $1.36 \cdot 10^{-10}$ эВ. Фононные частоты рассчитывались на основе теории возмущений функционала плотности (DFPT) по q-сетке $6 \times 6 \times 4$.

Фононные расчеты и квазигармоническая аппроксимация на основе метода конечного смещения выполнены в пакете Phonopy, написанный на языке Python [24, 25]. Для получения межатомных сил, возникающих при наличии смещения, для h-Ru и c-Ru использовались суперячейки размерностью $2 \times 2 \times 2$, что позволяет рассматривать 16 и 32 атома Ru для h-Ru и c-Ru, соответственно. Свободная энергия Гельмгольца для суперячеек получена с использованием метода Парлинского—Ли—Кавазоэ [26] с величиной конечного смещения 0.01 Å. Суперячейки большей размерности не рассматривались в виду их большой вычислительной сложности. Интегрирование по зоне Бриллюэна для суперячеек осуществлялось по сетке $9 \times 9 \times 6$ и $10 \times 10 \times 10$ для h-Ru и c-Ru, соответственно.

В качестве обменно-корреляционного функционала были рассмотрены следующие приближения (псевдопотенциалы): WC [27], PBE [28], PBEsol [29], PW91 [30] из библиотеки pseudopotential1.0 [31] и PBE, PBEsol из библиотеки PseudoDojo [32]. Исходные кристаллические структуры гексагонального рутения с пространственной группой $P6_3/mmc$ (№ 194) и кубического рутения с пространственной группой $Fm\bar{3}m$ (№225) для моделирования были взяты из кристаллографической базы данных «Materials Project» [33].

Работа выполнялась с использованием инфраструктуры Центра коллективного пользования «Высокопроизводительные вычисления и большие данные» (ЦКП «Информатика») ФИЦ ИУ РАН (г. Москва).

3. Результаты

Рассчитанные параметры кристаллической решетки h-Ru сначала сравниваются с экспериментальными и теоретическими результатами из различных источников в таблице 1.

Таблица 1. Сравнение структурных параметров гексагонального рутения.
 Теоретические результаты получены при температуре 0 К, а экспериментальные при комнатной температуре (293 К)

Метод	Длина a, Å	Длина c, Å	c/a	Плотность, г/см ³	Объем на атом, Å ³	Источник
NC PBE	2.722	4.296	1.578	12.173	13.787	Эта работа
NC PBEsol	2.692	4.251	1.579	12.577	13.344	Эта работа
PAW PBE	2.721	4.294	1.578	12.186	13.772	Эта работа
PAW PBEsol	2.691	4.250	1.579	12.590	13.330	Эта работа
PAW PW 91	2.724	4.296	1.577	12.162	13.800	Эта работа
PAW WC	2.698	4.260	1.579	12.500	13.427	Эта работа
Эксперимент	2.706	4.283	1.583	12.358	13.575	[13]
Эксперимент	2.706	4.281	1.582	12.364	13.567	[34]
Эксперимент	2.706	4.281	1.582	12.364	13.573	[36]
PAW PBE	2.722	4.293	1.577	12.185	13.774	[37]
PAW PBEsol	2.692	4.249	1.578	12.587	13.332	[37]
NC LDA	2.691	4.322	1.606	12.384	13.552	[11]
PAW GGA	2.740	4.320	1.577	11.950	14.044	[10]
NC LDA	2.701	4.270	1.581	12.442	13.489	[8]

В целом, теоретические расчеты показывают хорошие согласие с имеющимися экспериментальными данными. Как можно заметить, псевдопотенциалы PBE, PW91 и GGA уже при 0 К завышают значения параметров решетки по сравнению с экспериментом, а это значит, что при достижении комнатной температуры при расчете QHA, они явно будут показывать завышенное значение параметров решетки по сравнению с экспериментальными данными. PBEsol, и WC показывают меньшее значение параметров решетки, что справедливо для случая 0 К, причем WC показывает чуть большие параметры по сравнению с PBEsol. По причине того, что в некоторых расчетах желательно использовать параметры решетки при комнатной температуре при нулевом давлении, то мы остановимся на псевдопотенциале WC как наиболее близком к экспериментальным данным.

Так как по чистому кубическому рутению нет экспериментальных кристаллографических данных, то параметры решетки были вычислены только с псевдопотенциалом WC, как наиболее подходящему к гексагональному рутению. Параметр решетки *c*-Ru составляет 3.779 Å, плотность 12.443 г/см³, а объем на атом – 13.488 Å³. То есть результаты очень схожи с таковыми у *h*-Ru. Данные других теоретических работ оценивают параметр решетки кубического рутения как 3.818 Å [38].

Результаты квазигармонической аппроксимации для анизотропной температурной зависимости длин сторон гексагонального рутения представлены на рис. 2.

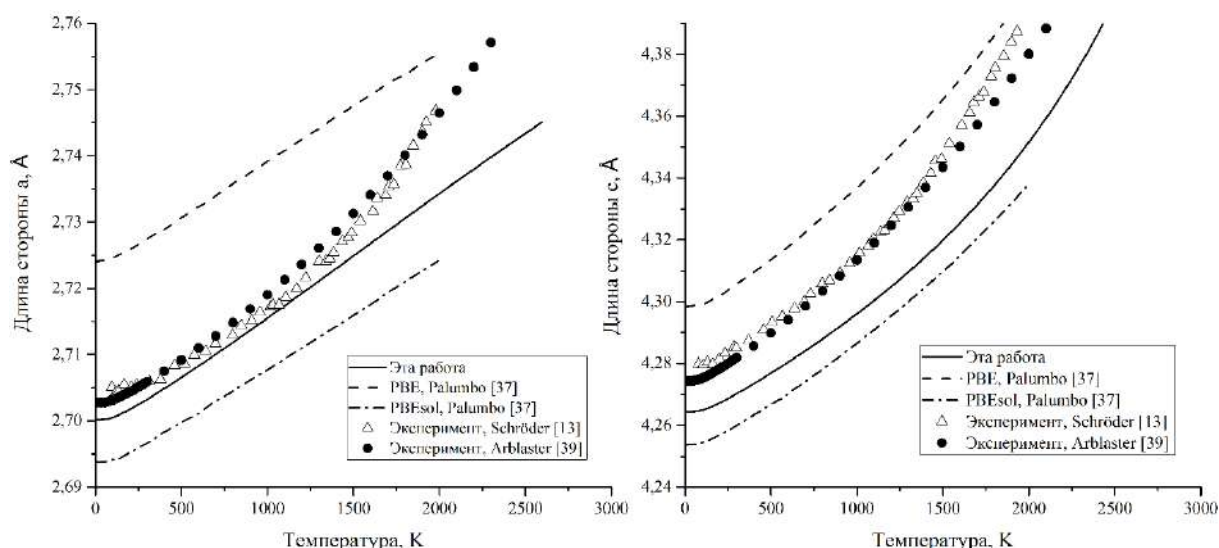


Рис. 2. Температурная зависимость параметров решетки гексагонального рутения: (слева) длина стороны *a*, (справа) длина стороны *c*.

Экспериментальные данные представлены в работах [13] и [39]

Температурная зависимость находится в хорошем согласии с экспериментальными данными, а расхождение можно, в основном, связать с параметрами расчета и обменно-корреляционного функционала. Экспериментальные данные обеих зависимостей находятся между теоретическими результатами, полученными с использованием функционала PBE и PBEsol из [37], а также функционала WC в нашей работе, практически во всем диапазоне температур, показанном на рис. 2. Результаты функционала WC демонстрируют более близкое соответствие экспериментальным параметрам решетки, чем PBEsol из работы [37] и находятся в пределах экспериментального разброса.

На рис. 3 представлены данные о температурной зависимости объема кубической и гексагональной примитивной ячейки рутения, полученной с использованием QHA через фоновый расчет на основе DFPT и FDSA.

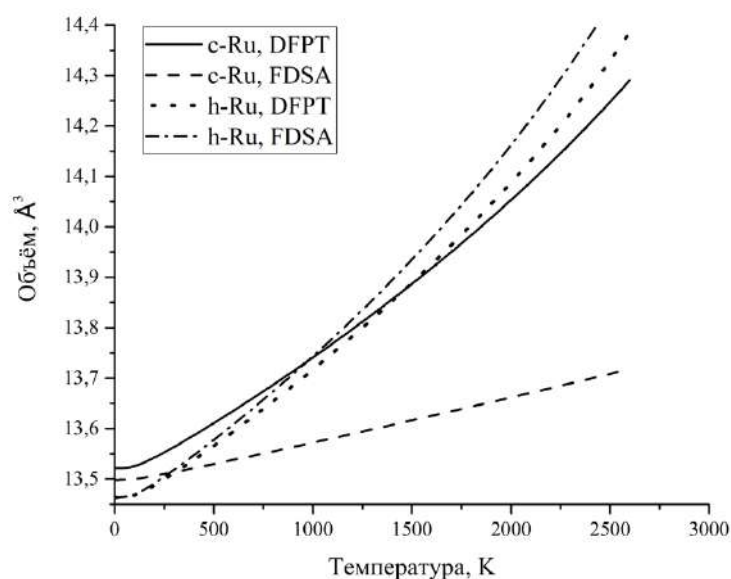


Рис. 3. Температурная зависимость объема примитивной кубической и гексагональной ячейки рутения

Как можно заметить, метод конечных смещений показывает заметно более медленный рост объема примитивной кубической ячейки с повышением температуры, чем теория возмущений функционала плотности. Для примитивной гексагональной ячейки разница в объеме между подходами DFPT и FDSA в области низких температур (до 293 K) пренебрежимо мала, но увеличивается с ростом температуры.

Результаты для коэффициентов объемного и теплового расширения показаны на рис. 4. Связь объемного коэффициента с линейным коэффициентом для гексагональной структуры устанавливается через соотношение (11), а для кубической решетки как сумма трех линейных коэффициентов.

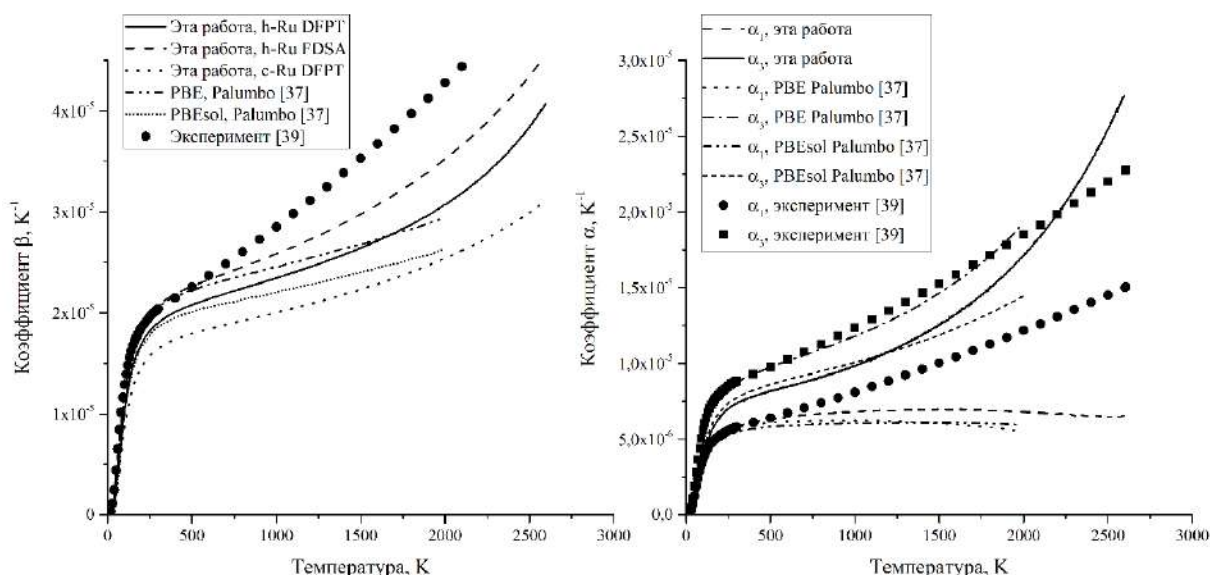


Рис. 4. Температурная зависимость коэффициента объемного теплового расширения β (слева) и линейных коэффициентов теплового расширения α (справа).

На левом рисунке также добавлен результат для кубического рутения

При низких температурах имеется хорошее согласие между экспериментальными значениями и результатами FDSA для коэффициента β для гексагонального рутения. Кривая, рассчитанная при помощи DFPT совпадает по форме зависимости с кривой, рассчитанной по FDSA, но лежит немного ниже по величине. Другие теоретические результаты с использованием

функционала PBE также показывают хорошее совпадение с экспериментом в области низких температур. С увеличением температуры коэффициент объемного теплового расширения для кубического рутения становится меньше, чем для гексагонального. При высоких температурах рассчитанное тепловое расширение вдоль направления a (α_1) сильно занижает экспериментальные значения, а хорошее согласие между результатами PBE и экспериментом вдоль направления c (α_3), по мнению авторов, является случайным.

Зависимость изобарной теплоемкости C_p от температуры представлена на рис. 5.

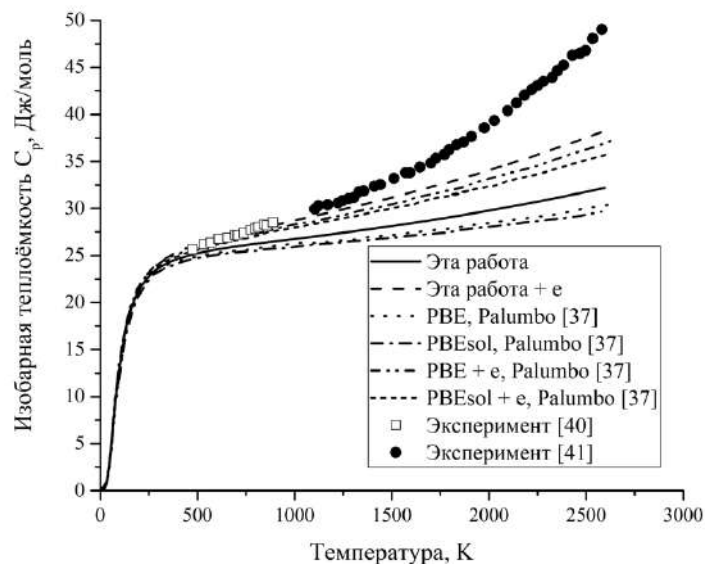


Рис. 5. Температурная зависимость изобарной теплоемкости C_p гексагонального рутения. Обозначение «+ e» в легенде рисунка обозначает учет электронного вклада в теплоемкость согласно формуле (3). Экспериментальные данные представлены в работах [40, 41]

В зависимости от выбранного псевдопотенциала и учета электронного вклада, данные о температурной зависимости теплоемкости рутения отличаются. Более близкий к экспериментальным значениям расчетный результат, достигается при учете электронного вклада. Немного более корректное значение получается при использовании обменно-корреляционного функционала WC по сравнению с PBE и PBEsol. В области температур до примерно 500 К все расчетные подходы показывают схожие результаты [42]. Теплоемкость кубического рутения в области низких температур практически совпадает с теплоемкостью гексагонального рутения, но разница увеличивается с ростом температуры.

4. Заключение

Используя квантово-механический подход на основе квазигармонической аппроксимации были получены температурные зависимости гексагонального и кубического рутения. Применив метод конечных смещений и теорию возмущений функционала плотности найдены температурные зависимости параметров решетки обоих кристаллических структур, коэффициентов линейного и объемного расширения, а также изобарной и изохорной теплоемкостей.

Полученные результаты квазигармонической аппроксимации для коэффициентов теплового расширения α_1 и β , параметров решетки и теплоемкости показывают, что квазигармоническое приближение нарушается для рутения выше определенной температуры, которая зависит от оцениваемой величины и составляет от 500 К для теплоемкости (рис. 5) до 1000 К для параметров решетки (рис. 2).

Обменно-корреляционный функционал WC показывает немного более близкие к экспериментальным данным значения рассчитанных величин, а метод конечных смещений сопоставим с теорией возмущения функционала плотности по рассчитанным значениям, за исключением отдельных случаев, таких как зависимость объема кубической решетки от температуры.

Литература

1. Валеев А.С., Красников Г.Я. Технология изготовления внутрикристалльных и межкуристалльных межсоединений современных СБИС (обзор, концепция развития) // Микроэлектроника. 2015. Т. 44, № 3. С. 180–201.
2. Красников Г.Я., Орлов О.М. Отличительные особенности и проблемы КМОП-технологии при уменьшении проектной нормы до уровня 0.18 мкм и меньше // Российские нанотехнологии. 2008. Т. 3, № 7-8. С. 124–128.
3. Резванов А.А. и др. Исследование уменьшения деградации low-k диэлектрика путем селективного осаждения защитного полимера на стенки пор // Электронная техника. Серия 3: Микроэлектроника. 2016. № 4. С. 10–19.
4. Rezvanov A.A. et al. Effect of terminal methyl group concentration on critical properties and plasma resistance of organosilicate low-k dielectrics // Journal of Vacuum Science & Technology A. 2020. Vol. 38, no. 3. DOI: 10.1116/1.5143417.
5. Bernasconi R., Magagnin L. Ruthenium as diffusion barrier layer in electronic interconnects: current literature with a focus on electrochemical deposition methods // Journal of The Electrochemical Society. 2018. Vol. 166, no. 1. P. D3219. DOI: 10.1149/2.0281901jes.
6. Kim J. et al. Electromigration reliability of barrierless ruthenium and molybdenum for sub-10 nm interconnection // ACS Applied Electronic Materials. 2023. Vol. 5, no. 5. P. 2447–2453. DOI: 10.1021/acsaelm.3c00070.
7. Ponosov Y.S. et al. Raman scattering in osmium under pressure // Physical Review B—Condensed Matter and Materials Physics. 2005. Vol. 71, no. 22. P. 220301. DOI: 10.1103/PhysRevB.71.220301.
8. Heid R. et al. Anomalous lattice dynamics of ruthenium // Physical Review B. 2000. Vol. 61, no. 18. P. 12059. DOI: 10.1103/PhysRevB.61.12059.
9. Smith H.G., Wakabayashi N. Phonon anomalies and superconductivity in the hcp metals, Tc, Re, and Ru // Solid State Communications. 1981. Vol. 39, no. 2. P. 371–374. DOI: 10.1016/0038-1098(81)90693-1.
10. Souvatzis P., Eriksson O. *Ab initio* calculations of the phonon spectra and the thermal expansion coefficients of the 4d metals // Physical Review B—Condensed Matter and Materials Physics. 2008. Vol. 77, no. 2. P. 024110. DOI: 10.1103/PhysRevB.77.024110.
11. Yaozhuang N. et al. *Ab initio* thermodynamics of metals: Pt and Ru // Physica B: Condensed Matter. 2007. Vol. 395, no. 1-2. P. 121–125. DOI: 10.1016/j.physb.2007.03.005.
12. Arblaster J.W. The thermodynamic properties of ruthenium on ITS-90 // Calphad. 1995. Vol. 19, no. 3. P. 339–347. DOI: 10.1016/0364-5916(95)00031-9.
13. Schröder R.H., Schmitz-Pranghe N., Kohlhaas R. Experimentelle Bestimmung der Gitterparameter der Platinmetalle im Temperaturbereich von -190 bis 1709°C // International Journal of Materials Research. 1972. Vol. 63, no. 1. P. 12–16. DOI: 10.1515/ijmr-1972-630103.
14. Ross R.G., Hume-Rothery W. High temperature X-ray metallography: I. A new Debye-Scherrer camera for use at very high temperatures II. A new para-focusing camera III. Applications to the study of chromium, hafnium, molybdenum, rhodium, ruthenium and tungsten // Journal of the Less Common Metals. 1963. Vol. 5, no. 3. P. 258–270. DOI: 10.1016/0022-5088(63)90031-6.
15. Quartermann P. et al. Demonstration of Ru as the 4th ferromagnetic element at room temperature // Nature communications. 2018. Vol. 9, no. 1. P. 2058. DOI: 10.1038/s41467-018-04512-1.
16. Watanabe S. et al. First-principle band calculation of ruthenium for various phases // Journal of magnetism and magnetic materials. 2000. Vol. 220, no. 2-3. P. 277–284. DOI: 10.1016/S0304-8853(00)00466-2.

17. Baroni S. et al. Phonons and related crystal properties from density-functional perturbation theory // *Reviews of modern Physics*. 2001. Vol. 73, no. 2. P. 515. DOI: 10.1103/RevModPhys.73.515.
18. Wallace D.C., Callen H. Thermodynamics of crystals // *American Journal of Physics*. 1972. Vol. 40, no. 11. P. 1718–1719. DOI: 10.1119/1.1987046.
19. Giannozzi P. et al. Quantum ESPRESSO toward the exascale // *The Journal of chemical physics*. 2020. Vol. 152, no. 15. DOI: 10.1063/5.0005082.
20. Gong X., Dal Corso A. An alternative GPU acceleration for a pseudopotential plane-waves density functional theory code with applications to metallic systems // *Computer Physics Communications*. 2025. Vol. 308. P. 109439. DOI: 10.1016/j.cpc.2024.109439.
21. Davidson C.G. The Iterative Calculation of a Few of the Lowest Eigenvalues and Corresponding Eigenvectors of Large Real-Symmetric Matrices // *Journal of Computational Physics*. 1975. Vol. 17. P. 87–94. DOI: 10.1016/0021-9991(75)90065-0.
22. Monkhorst H.J., Pack J.D. Special points for Brillouin-zone integrations // *Physical review B*. 1976. Vol. 13, no. 12. P. 5188. DOI: 10.1103/PhysRevB.13.5188.
23. Marzari N. et al. Thermal contraction and disordering of the Al (110) surface // *Physical review letters*. 1999. Vol. 82, no. 16. P. 3296. DOI: 10.1103/PhysRevLett.82.3296.
24. Togo A. et al. Implementation strategies in phonopy and phono3py // *Journal of Physics: Condensed Matter*. 2023. Vol. 35, no. 35. P. 353001. DOI: 10.1088/1361-648X/acd831.
25. Togo A. et al. First-principles phonon calculations of thermal expansion in Ti_3SiC_2 , Ti_3AlC_2 , and Ti_3GeC_2 // *Physical Review B—Condensed Matter and Materials Physics*. 2010. Vol. 81, no. 17. P. 174301. DOI: 10.1103/PhysRevB.81.174301.
26. Parlinski K., Li Z.Q., Kawazoe Y. First-principles determination of the soft mode in cubic ZrO_2 // *Physical Review Letters*. 1997. Vol. 78, no. 21. P. 4063. DOI: 10.1103/PhysRevLett.78.4063.
27. Wu Z., Cohen R.E. More accurate generalized gradient approximation for solids // *Physical Review B—Condensed Matter and Materials Physics*. 2006. Vol. 73, no. 23. P. 235116. DOI: 10.1103/PhysRevB.73.235116.
28. Perdew J.P., Burke K., Ernzerhof M. Generalized gradient approximation made simple // *Physical review letters*. 1996. Vol. 77, no. 18. P. 3865. DOI: 10.1103/PhysRevLett.77.3865.
29. Perdew J.P. et al. Restoring the density-gradient expansion for exchange in solids and surfaces // *Physical review letters*. 2008. Vol. 100, no. 13. P. 136406. DOI: 10.1103/PhysRevLett.100.136406.
30. Perdew J.P. et al. Atoms, molecules, solids, and surfaces: Applications of the generalized gradient approximation for exchange and correlation // *Physical review B*. 1992. Vol. 46, no. 11. P. 6671. DOI: 10.1103/PhysRevB.46.6671.
31. Dal Corso A. Pseudopotentials periodic table: From H to Pu // *Computational Materials Science*. 2014. Vol. 95. P. 337–350. DOI: 10.1016/j.commatsci.2014.07.043.
32. van Setten M.J. et al. The PseudoDojo: Training and grading a 85 element optimized norm-conserving pseudopotential table // *Computer Physics Communications*. 2018. Vol. 226. P. 39–54. DOI: 10.1016/j.cpc.2018.01.012.
33. Jain A. et al. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation // *APL materials*. 2013. Vol. 1, no. 1. DOI: 10.1063/1.4812323.
34. Hall E.O., Crangle J. An X-ray investigation of the reported high-temperature allotropy of ruthenium // *Acta Crystallographica*. 1957. Vol. 10, no. 3. P. 240–241. DOI: 10.1107/S0365110X57000742.

35. Černohorský M. The ratio method for absolute measurements of lattice parameters with cylindrical cameras // *Acta Crystallographica*. 1960. Vol. 13, no. 10. P. 823–826. DOI: 10.1107/S0365110X60001989.
36. Anderson E., Hume-Rothery W. The equilibrium diagram of the system molybdenum-ruthenium // *Journal of the Less Common Metals*. 1960. Vol. 2, no. 6. P. 443–450. DOI: 10.1016/0022-5088(60)90026-6.
37. Palumbo M., Dal Corso A. Lattice dynamics and thermophysical properties of hcp Os and Ru from the quasi-harmonic approximation // *Journal of Physics: Condensed Matter*. 2017. Vol. 29, no. 39. P. 395401. DOI: 10.1088/1361-648X/aa7dca.
38. Li W.Z. et al. Chemical insights into the design and development of face-centered cubic ruthenium catalysts for Fischer–Tropsch synthesis // *Journal of the American Chemical Society*. 2017. Vol. 139, no. 6. P. 2267–2276. DOI: 10.1021/jacs.6b10375.
39. Arblaster J.W. Crystallographic properties of ruthenium // *Platinum Metals Review*. 2013. Vol. 57, no. 2. P. 127–136. DOI: 10.1595/147106713X665030.
40. Cordfunke E.H.P., Konings R.J.M. The high temperature thermophysical properties of ruthenium and palladium // *Thermochimica acta*. 1989. Vol. 139. P. 99–106. DOI: 10.1016/0040-6031(89)87013-3.
41. Раманаускас Г.Р. и др. Высокотемпературная энтальпия вакуумплавленных рутения и осмия // *Высококачественные материалы*. 1988. № 4. С. 149.
42. Итальянцев А.Г., Константинов В.С. Расчет термодинамических потенциалов орторомбической фазы в оксиде гафния в зависимости от температуры и давления // *Электронная техника. Серия 3: Микроэлектроника*. 2019. № 3. С. 43–50.

Разработка языка программирования для dataflow вычислений

С.М. Салибекян

Национальный исследовательский университет «Высшая школа экономики»

Фокус внимания статьи сосредоточен на описании разработанного автором языка программирования (ОА язык), «природно» ориентированного на описание алгоритма в dataflow парадигме вычислительного процесса: описание обмена данными в виде токенов между устройствами вычислительной системы. Язык обладает рядом преимуществ: предельная простота синтаксиса, что ускоряет процесс программирования; возможность описывать сложных структур данных, например, семантические сети, и обрабатывать их; управление как аппаратной dataflow системой, так и виртуальной, которую можно запустить на ЭВМ классической архитектуры; является мультипарадигменным.

Ключевые слова: dataflow, параллельные вычисления, параллельный язык программирования, высокопроизводительные вычисления, распределенные вычислительные системы.

1. Введение

Настоящая статья посвящена описанию языка программирования (объектно-атрибутный (ОА) язык), созданного для описания алгоритма в dataflow вычислительной системе (ВС) [1]. Отличительной особенностью языка является то, что его синтаксис максимально адаптирован для описания именно в dataflow парадигме вычислений, тогда как большинство языков для подобных целей относятся к функциональной парадигме (Sisal [2], VAL, ID, ПИФАГОР [3]). Язык позволяет управлять dataflow системами, реализованными как на аппаратном, так и на программном уровнях. Его отличительными особенностями являются: возможность настройки устройств, входящих в вычислительную систему; описание сложно структурированных информационных конструкций; трансформация информационных конструкций; описание алгоритма для распределенной ВС. В настоящее время имеется реализация транслятора данного языка. Транслятор создается методом раскрутки, когда первая версия транслятора реализует самую примитивную функциональность языка, затем на созданном языке пишется следующая версия транслятора языка с большей функциональностью и т.д. В настоящее время происходит реализация уже третьей версии ОА-языка.

Следует отметить, что ОА-язык предназначен для управления вычислениями в dataflow ВС, относящейся к классу одноуровневых машин Венну [4]. Т.е. токены (операнды, снабженные служебной информацией) поступают непосредственно на исполнительные устройства. Исполнительные устройства производят накопление в своих внутренних регистрах полного комплекта данных для осуществления определенной операции над ним. ОА ВС, реализованная программным способом, т.е. в ней исполнительные устройства представляются в виде программных блоков, несколько напоминает акторную систему программирования [5, 6]. Актор – это программная функция, которая, однако, получает свои операнды не одновременно, как в классическом программировании, а последовательно в виде токенов. У каждого актора существует интерфейс, каждый слот которого описывает входной операнд или выходное данные. В ОА-языке аналогом актора является функциональное устройство (ФУ). Однако, существует одно отличие – это универсальный интерфейс ФУ. Он принимает только по одному операнду в виде токена, а распознавание операнда осуществляется по атрибуту токена. Таким образом значительно упрощается программирование ОА системы, т.к. избавляет программиста от рутинного описания интерфейса функции актора.

В области создания dataflow языков программирование проделана достаточно серьезная работа. В качестве примера привести статью [7]. Описанный в статье язык предназначен для решения ограниченного класса задач, однако относится, именно, к парадигме dataflow. В статье [8] представлен графический язык для dataflow системы. К потоковой парадигме также можно отнести практически все графические языки программирования (например, LabView и Simulink), т.к. они строятся на базе потокового графа [9], который описывает процесс пересылки токенов между различными вычислительными устройствами ВС.

ОА язык изначально разрабатывался для описания вычислительного процесса в ВС объектно-атрибутной (ОА) архитектуры [10], относящейся к классу dataflow. Однако он может применяться и для других dataflow ВС, относящихся к классам одноуровневых и двухуровневых машин по классификации Венна [4].

Вторая глава статьи посвящена описанию синтаксиса ОА-языка, в том числе и его последний, разрабатываемой в настоящее время, версии. Внимание третьей главы фокусируется на среде ОА программирования и моделирования, предназначенной для создания и запуска программ на ОА языке. В четвертой главе приведены примеры решения различных прикладных вычислительных задач с применением ОА языка.

2. ОА язык: основы

2.1 Синтаксис ОА языка

Основы ОА языка были представлены на конференции ПаВТ 2013 [11]. Философия языка следующая. Весь вычислительный процесс реализуется с помощью набора функциональных устройств (ФУ); по названию ясно, что данное устройства реализует какую-то вычислительную функцию (например, арифметико-логические операции, ввод-вывод на консоль или передача данных по вычислительной сети). Они принимают токены с операндами, накапливают в своих внутренних регистрах комплект данных, необходимый для осуществления операции, производят операцию и пересылают результат своих вычислений на другие ФУ для дальнейшей обработки. Токены в ОА языке именуются милликомандами (МК). Такое название пошло от того, что для осуществления классической команды необходим приход на ФУ нескольких токенов с исходными данными, т.е. одна МК является частью классической команды.

МК в ОА языке обозначается с помощью знака «=», с правой стороны от которого находится описание атрибута МК, слева – нагрузки (т.е. данных, прикрепленных к токену). К атрибуту данных может так же добавляться и имя ФУ, которому направляется МК. Название ФУ отделяется от атрибута данных знаком «.». Например, выполнение операции сложения на ОА языке будет выглядеть следующим образом: `ALU.Sum1=2 ALU.Sum2=2 ALU.Out=y`, где `y` – мнемоника переменной для записи результата вычисления, `ALU` – мнемоника ФУ арифметико-логическое устройство (АЛУ). МК `Sum1` служит для индикации первого слагаемого, `Sum2` – для индикации второго, по МК `Out` осуществляется вывод результата вычислений, полученный на ФУ АЛУ. Следует отметить, что последовательность прихода МК `Sum1` и `Sum2` может быть любой, однако `Out` должна появиться только после вычисления результата. Для того, чтобы выдача результата производилась автоматически по получению результата можно воспользоваться МК `OutMkSet` – установить МК для результата. Данная МК записывает атрибут, который прикрепляется к результату вычислений во время его выдачи. Например, мы можем выполнить `ALU.OutMkSet=Console.Print`, где `Console.Print` – МК вывода на консоль. Теперь программка преобразуется в `ALU.OutMkSet=Console.Out ALU.Sum1=2 ALU.Sum2=2`. Тогда при выполнении данной последовательности МК ФУ, получив два необходимых операнда, осуществит выдачу МК с результатом автоматически. Результат будет оформлен в виде МК с атрибутом, установленным ранее (в нашем случае `Console.Print`), которая будет иметь вид: `Console.Print=Rez`, где `Rez` – результат вычисления, полученный АЛУ. Сгенерированная МК передана выредается на ФУ-консоль, которая осуществит вывод данных.

Последовательность МК может объединяться в так называемую информационную капсулу (ИК). ИК хранит к себе упорядоченную последовательность МК. Например, ИК может хранить программу обработки ошибки, которую может зафиксировать ФУ. Указатель на ИК с програм-

мой может быть записан во внутренние регистры ФУ с помощью МК ErrProgSet. Например, фрагмент программы может выглядеть так: `ALU.ErrProgSet={Console.Print="Arithmetic error" ALU.Stop}`. Данная программа будет запущена при возникновении ошибки при вычислениях, выполняемых ФУ АЛУ. В ОА языке под выполнением программы подразумевается последовательная выдача МК из ИК. В результате запуска на консоль будет выведено сообщение об ошибке, а также произведена остановка работы ФУ АЛУ.

В ОА ВС имеется набор типов ФУ. Например, в разработанной среде ОА программирования и моделирования имеется порядка 25 типов ФУ. ФУ каждого типа может обрабатывать токены с определенным набором МК. Создание ФУ в ОА языке происходит по МК `NewFU={Mnemo="ALU1" FUType=FUStreamFloatALU}`, где Mnemo – атрибут мнемоники ФУ, FUType – атрибут типа ФУ. Так по вышеприведенной МК будет создано ФУ с именем «ALU1» и типом FUStreamFloatALU (дробное потоковое АЛУ). Вышеприведенный синтаксис составляет ядро ОА языка, на котором можно создавать программы.

ОА язык, например, позволяет легко описать вычисления, заданные посредством потокового графа [12] (операции в узлах потокового графа могут быть как мелкозернистыми (т.е. элементарные операции), так и крупнозернистыми (крупнозернистая операция может быть эквивалентна программе с несколькими сотнями или тысячами строк). Пусть, например, нужно вычислить арифметическое выражение следующего вида: $Y = A * x^2 + B * x + C$. Потоковый граф для данного выражения приведен на рис. 1. Реализация данного графа основывается на применении ФУ типа потоковое АЛУ. Его особенность состоит в том, что оно работает в потоковом режиме: принимает операнды в любой последовательности прихода, выполняет только одну операцию за время вычислительного процесса, автоматически выдает результат вычислений.

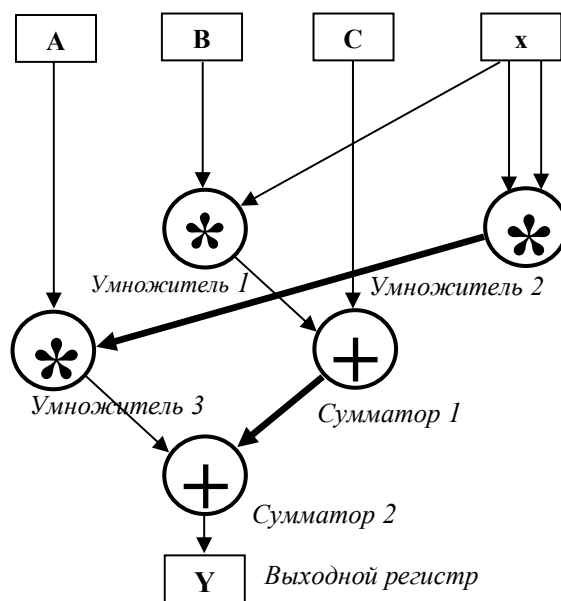


Рис. 1. Пример потокового графа

В ОА языке для реализации потокового графа на рис. 1 необходимо задать одну переменную, три константы, 6 ФУ. Программа с комментариями к коду приведена на рис. 2.

Программа на листинге выше начинается с команд создания ФУ. Пять из них имеют тип FUStreamFloatALU (потоковое АЛУ). Шестое – тип FUConsole (консоль ввода-вывода), которое служит для вывода результата вычислений. Далее идет секция инициализации переменных и описания именованных констант. Затем расположена секция описания пересылок данных между ФУ. МК ReceiverMkSet устанавливает для ФУ код атрибута МК с результатом выполнения арифметико-логической операции. Например, `Mul1.ReceiverMkSet=Sum1.Add` программирует ФУ Mul1 на то, чтобы оно выдало результат своих вычислений с атрибутом Sum1.Add – в итоге данное число будет передано на ФУ Sum, а по атрибуту Add ФУ Sum1 поймет, что данный операнд является слагаемым. Аналогичным образом происходит настройка пересылок и между другими ФУ. А для ФУ, которое получает окончательный результат, приемщиком является консоль, выводящая этот результат на экран (МК Console.Print).

```
NewFU={Mnemo="Sum1" FUType=FUStreamFloatALU} // Создание виртуального ФУ
NewFU={Mnemo="Sum2" FUType=FUStreamFloatALU}
NewFU={Mnemo="Mul1" FUType=FUStreamFloatALU}
NewFU={Mnemo="Mul2" FUType=FUStreamFloatALU}
NewFU={Mnemo="Mul3" FUType=FUStreamFloatALU}
NewFU={Mnemo="Console" FUType=FUConsole}

// Инициализация переменных и констант
x=0.3 // Инициализация переменной
A#5 // Константа
B#-2 // Константа
C#28 // Константа

// Описание пересылок токенов
Mul1.ReceiverMkSet=Sum1.Add // Описание пересылки операнда из Умн1 на Сум1
Mul2.ReceiverMkSet=Mul3.Mul // из Умн2 на Умн3
Mul3.ReceiverMkSet=Sum2.Add // из Умн3 на Сум2
Sum1.ReceiverMkSet=Sum2.Add // из Сум1 на Сум2
Sum2.ReceiverMkSet=Console.Print // Выдача результата с Сум2 на консоль

// Выдача исходных данных
Mul1.Mul=B
Mul1.Mul=x
Mul2.Sqr=x
Mul3.Mul=A
Sum2.Add=C
```

Рис. 2. Реализация ОА программы для потокового графа на рис. 1

Только после настройки пересылок данных между ФУ в ОА ВС можно подать исходные данные. Данные снабжаются соответствующими атрибутами, и сформированные таким образом МК выдаются на ФУ для обработки. Например, когда оба операнда поступают на ФУ Mul1, оно вычисляет их произведение, оформляет его в виде токена с атрибутом Sum1.Add и выдает МК, которая передается на ФУ Sum1 и порождает дальнейшие вычисления. Если во время выполнения вычислительного процесса данные придут сразу на несколько ФУ, то ФУ будут работать параллельно.

Как видно из приведенного примера кода ОА программы, синтаксис ОА языка весьма прост, однако он представляет собой весьма мощный инструмент параллельного программирования, т.к. позволяет описывать программы с самораспараллеливанием вычислений. Дело в том, что в dataflow парадигме, синхронизация вычислений происходит по данным: как данные поступили на исполнительное устройство и набрался полный комплект для выполнения операции, операция начинает выполняться, а результат выполнения пересылается на другие ФУ. Заметим, что программист не заботится о распараллеливании вычислительного процесса, процесс как бы сам распараллеливается в зависимости от приходящих данных.

2.2 Дальнейшее развитие ОА языка

В листинге на рис. 2 приведена программа на второй версии ОА языка (компилятор реализуется с 2008 года). А в настоящее время проводится работа по реализации третьей версии, обладающей более широким набором синтаксических конструкций. Расширение функциональности ОА языка позволит значительно снизить трудоемкость создания ОА программы. Отметим, что компилятор ОА языка сам пишется на ОА языке, т.к. ОА язык, помимо прочего, оказался весьма удобным в том числе и для создания трансляторов языков высокого уровня [13]. Перечислим нововведения, которые ожидаются в третьей версии.

Во-первых, арифметико-логические выражения. Заметим, что во второй версии была возможность присваивать переменным только константные значения, а также записывать в них результаты вычислений с ФУ. Если же была необходимость вычислить какое-то значение, то это

приходилось делать с помощью ФУ АЛУ и вручную прописывать алгоритм вычисления выражения. Во-вторых, введение возможности работы с векторами. В-третьих, добавление высокоуровневых императивных конструкций типа if-then-else, while, for. Отметим, что императивные конструкции в основном предполагают последовательное выполнение вычислений, что противоречит основной задаче dataflow – максимальному распараллеливанию. Однако в некоторых случаях последовательные вычисления приемлемы, например, в случае, если алгоритм по своей природе последовательный и его невозможно распараллелить, или, если вычислительный процесс удобно разбить на несколько параллельных вычислительных нитей, каждая из которых выполняется последовательно. К тому же, такие высокоуровневые синтаксические конструкции значительно упрощают код программы и делают его более читабельным. В-четвертых, в синтаксис добавляется много разных приятных мелочей, упрощающих создание кода.

В качестве примера таких мелочей можно привести возможность создания вектора ФУ. Каждое ФУ в этом векторе имеет свой индекс, по которому к нему можно обращаться. Например, `ALU[3].Add=10`, т.е. мы пересылаем константу 10 в качестве слагаемого для АЛУ с индексом 3. Если же возникает необходимость передать МК сразу нескольким ФУ, то можно указать сразу несколько ФУ или диапазон индексов ФУ, например, `ALU[2,6,9].Set=0` (установить в нуль аккумуляторы ФУ с индексами 2, 6, 9) или `ALU[3..7].Set=0` (установить в нуль аккумуляторы ФУ с индексами от 3 до 7). Также можно создать и матрицу ФУ, тогда каждое ФУ будет адресовать сразу двумя индексами, 3-мерный массив АЛУ и т.д. Такая возможность значительно облегчит описание вычислительных сеток, состоящих из ФУ (подробнее о вычислительных сетках будет рассказано в главе 4). Также полезным нововведением в синтаксис является применение табуляции в качестве программных скобок (аналогично языку Python).

В синтаксис ОА языка добавляются следующие императивные конструкции:

- Тернарная конструкция [14]. Данная условная конструкция может быть вставлена в арифметико-логическое выражение.

- Оператор бинарного ветвления if. Синтаксис оператора следующий: if условие {прогр. блок 1} elif {прогр. блок 2} else {прогр. блок 2}.

- Операторы цикла for и after. Параметры цикла записываются в трех секциях, разделенных знаком «:». Синтаксис оператора for: for секция 1 : секция 2 : секция 3 {тело цикла}. В секции 1 указывается начало диапазона генерируемой числовой последовательности, в секции 2 шаг последовательности, в секции 3 конец последовательности. Например, `for 2:10:2 { генерирует последовательность чисел 2, 4, 6, 8 (правый диапазон, аналогично питоновскому range в диапазон не входит) }`. В секциях также можно указывать условие продолжения цикла. Например, `for i=0: i<x+y: i++ { ... }`, тогда цикл становится аналогом циклу while языка C. Оператор after аналогичен for и отличается тем, что это – цикл с постусловием: первая итерация цикла выполняется безусловно и только со второй итерации начинается проверка условия продолжения цикла.

В дальнейшем предполагается расширение ОА языка возможностью программирования распределенных ВС. Так, появится функциональность описания секций программы, каждую из которых можно будет приписать к определенному вычислительному узлу распределенной ВС. Также предполагается создание различных средств автоматического распределения сегментов программы по различным вычислительным узлам.

3. ОА среда программирования и моделирования

Для программирования на ОА языке была создана среда ОА программирования и моделирования. Она представляет собой совокупность текстового редактора кода, различных инструментальных средств, компилятора ОА-языка. Также среда позволяет создавать виртуальную реализацию ОА ВС, на которой и происходит выполнение алгоритма, описанного на ОА языке (рис. 3).

Создание среды ОА программирования началось с 2008 года. Среда первой версии позволяла производить трансляцию программы на ОА языке во внутреннее представление программы для ОА ВС, и в таком виде она могла быть запущена на виртуальной реализации ОА ВС. Среда предоставляла программисту текстовый редактор, консоль вывода данных, различные полезные инструментальные средства для программирования: контекстное меню, всплывающие подсказки, предложение вариантов быстрого ввода синтаксических конструкций и т.д.

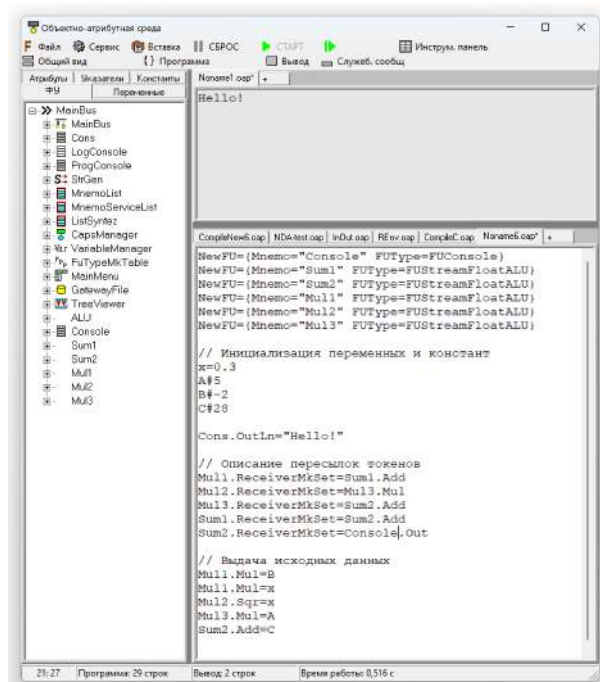


Рис. 3. Среда ОА программирования и моделирования

Вторая версия среды отличается более развитым компилятором ОА-языка, позволяющим создавать полноценные описания алгоритмов. В настоящее время рабочей является вторая версия ОА среды, хотя уже ведется реализация версии третьей на базе редактора VS Code. У третьей версии функциональность будет расширена следующими возможностями: подкраска текста программы, возможность создания многомодульных программ, подключение к git. К тому же, использование уже готового функционала редактора VS Code значительно улучшит качество работы в данном редакторе.

Среда ОА программирования и моделирования предоставляет пользователю обширные возможности по вводу-выводу и визуализации данных, имитационному моделированию вычислительного процесса. Так, для имитационного моделирования в состав виртуальной ОА-ВС входят специальные ФУ, с помощью которых можно задавать параметры параллельной ВС и параметры процесса моделирования: количество исполнительных устройств, конфигурация распределенной ВС, время выполнения вычислительных операций и пересылки данных. Имитационная модель ОА ВС строится на базе процессной сети Кана [15].

Среда второй версии была использована для моделирования выполнения многочисленных вычислительных задач на базе ОА ВС: перемножение матриц, алгоритм кластеризации данных Mean Shift [16], разностная схема решения дифференциальных уравнений, решение задачи «рюкзак» методом динамического программирования, семантический анализ естественного языка и т.д. Благодаря такому моделированию были оценены сложности решения такого типа задач на ОА ВС.

4. Примеры описания решений вычислительных задач на ОА языке

4.1 Задача «рюкзак»

Задача «рюкзак» формулируется следующим образом. Имеется рюкзак ограниченного объема V и множество из N предметов. Каждый предмет имеет два атрибута: объем v_i и цена p_i . Пусть набор целых чисел $\omega = \{\omega_1, \omega_2, \dots, \omega_N\}$ есть набор объемов предметов, $p = \{p_1, p_2, \dots, p_N\}$ есть набор стоимостей предметов. Нужно найти набор бинарных величин $\beta = \{\beta_1, \beta_2, \dots, \beta_N\}$, где $\beta_i = 1$, если предмет p_i включен в набор, $\beta_i = 0$, если предмет p_i не включен, и такой, что:

1. $\beta_1 \omega_1 + \dots + \beta_N \omega_N \leq V$
2. $\beta_1 p_1 + \dots + \beta_N p_N$ есть максимальна.

Т.е. суммарный объем предметов не должен превышать объем рюкзака, а суммарная стоимость всех предметов, поместившихся в рюкзак, должна быть максимальной. Существует несколько методов решения данной задачи, например, жадный и генетический алгоритмы, методы ветвей и границ. Однако наше внимание привлек метод решения с помощью динамического программирования [17], т.к. он дает точный результат, имеет небольшую вычислительную сложность, хорошо подходит для решения именно на базе ОА ВС. О других методах решения различных вычислительных задач посредством динамического программирования можно узнать в [18].

Методика решения задачи рюкзака с помощью динамического программирования следующая. Строится 2-мерное фазовое пространство решений задачи W размерностью $N \times V$, где V – множество дискретных значений объема, занятого предметами, помещенными в рюкзак. В этом пространстве по столбцам откладываются возможные значения объем рюкзака, а строка обозначает номер добавляемого предмета. С каждой точкой фазового пространства W_{ij} связана величина, показывающая максимальную стоимость предметов от 1 до i , занимающих объем j единиц рис. 4. Наклон косых линий по оси x перед линией i равен объему i -го предмета, который на данном шаге добавляется в рюкзак.

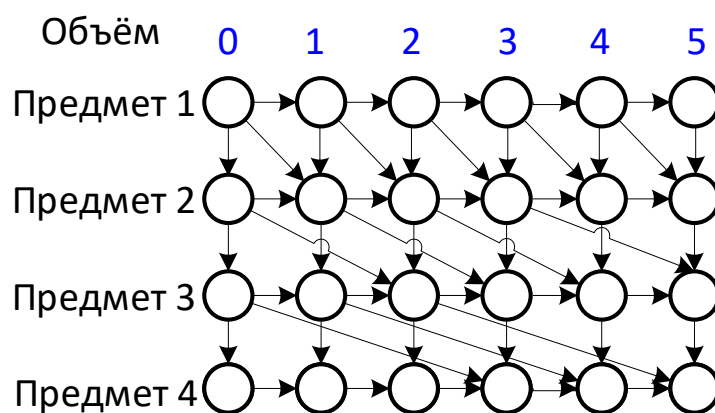


Рис. 4. Среда ОА программирования и моделирования

В начале выполнения алгоритма значения во всех точках фазового пространства равны 0. Заполнение пространства значениями идет сверху вниз. На каждой строке значение узла сетки вычисляется по формуле:

$$W[i,k]=\max(W[i-1, k-\omega[i]]+p[i], W[i-1, k]),$$

где W – множество точек фазового пространства;

i – номер добавляемого предмета;

k – объем рюкзака;

$p[i]$ – цена i -го предмета;

$\omega[i]$ – объем i -го предмета.

Причем, значение i лежит в пределах от v_1 до N , т.к. объем рюкзака меньший v_1 предметом занят быть не может.

Решением задачи будет максимальное значение, находящееся на последней строке фазового пространства $W[N]$. Т.е. $\max(W[N, k])$, $n=1 \dots NV$ есть решение задачи.

Где NV – количество разбиений объема рюкзака,

N – количество предметов.

Для решения данной задачи была сгенерирована вычислительная сетка, состоящая из устройств типа дробное потоковое АЛУ. Данный тип ФУ адаптирован к работе в составе вычислительной сетки. Для организации и управления сеткой применялось ФУ менеджер АЛУ.

Программа писалась на второй версии ОА языка, где нет описаний циклов, поэтому циклические действия организовывались с помощью ФУ менеджер, который обладает функциональностью по организации циклов.

На рис. 5 приведен фрагмент ОА программы, ответственный за формирование горизонтальных связей в вычислительной сетке. Связи между ФУ «налаживаются» посредством записи указателя на соседа во внутренний регистр ФУ. Данная ссылка используется ФУ для отправки информации соседнему ФУ или запроса информации у соседнего ФУ.

```
// Горизонтальные связи в сетке
IntAlu1.Set=V
IntAlu1.Dec // Количество итераций по одной строке V-1
Manager.ProgSet={ // Тело цикла, выполняемого менеджером
    FUCellularAutomatManager.Neighbor1To2Append // Установить МК для данных
    установить ссылку между соседями
    FUCellularAutomatManager.NeighborMk1Append=FUCellularAutomat.In_0_Set
    // Установить МК для данных
}
IntAlu.For={ // Цикл по строкам
    IntAlu.OutMk=Manager.Ind1Set=Manager.Ind2Set//Задать индексы соседних ФУ
    Manager.Ind2Add=1 // Задать индексы соседних ФУ
    IntAlu1.OutMk=Manager.ForExec // Выполнить цикл по столбцам строки
    Manager.Neighbor1Append // Добавить нулевую ссылку для последнего ФУ
    Manager.NeighborMk1Append=0 // Добавить нулевую ссылку для последнего ФУ
}
```

Рис. 5. Фрагмент ОА программы для генерации вычислительной сетки для решения задачи рюкзака

Вкратце объясним работу этого фрагмента программы. Здесь генерируются горизонтальные линии вычислительной сетки. Для этого организуется вложенный цикл, где цикл внешний производит обход всех строк сетки, а внутренний «бежит» по строке и настраивает связь между соседними ФУ (ссылка от левого соседа правому). Для налаживания связи необходимо не только передать ссылку на соседа, но и установить МК, прикрепляемое к передаваемым данным, чтобы приемник мог эти данные идентифицировать.

После настройки всей вычислительной сетки начинается выдача исходных данных, исходные данные порождают вычислительную волну, которая проходит по сетке. По окончании вычислений значения с последней строки сетки передаются на ФУ АЛУ, которое определяет максимум из них. Это значение является максимальной суммарной стоимостью предметов, которые можно поместить в рюкзак.

4.2 Кластеризация данных методом Wean Shift

Представим реализацию еще одной вычислительной задачи – кластеризация данных методом Mean Shift (сдвиг среднего значения). Алгоритм решения этой задачи на базе ОА ВС описан в [19]. Сейчас же речь пойдет о программной реализации данной задачи с помощью ОА языка.

Итак, алгоритм Mean Shift [20, 21] относится к классу кластеризации по плотности точек в фазовом пространстве. Пусть имеется множество точек T , положение которых в фазовом пространстве задается по N координатам. Каждая координата пространства, по сути, является мерой одной из характеристик объекта. В пространстве вводится определенная метрика например, ев-

клидово расстояние $\rho(t,t')=\sum_{i=1}^N\sqrt{t_i^2+(t'_i)^2}$ или $\rho(t,t')=\sum_{i=1}^N|t_i^2+(t'_i)^2|$, где $t,t'\in T$ и т.п. Для поиска кластера в фазовое пространство вводится шарообразная область C с центром m . Алгоритм подразумевает выполнение последовательности итераций, на каждой из которых вычисляется новый центр масс m' всех точек, попавших в область C , затем кластер передвигается в новую точку с центром масс m' (т.е. $m=m'$) и начинается следующая итерация. Математически доказано, что после конечного числа итераций центр масс смещаться не будет, т.е. $m=m'$; и, именно в этой точке и наблюдается максимальная концентрация точек.

Для реализации алгоритма нам пришлось разработать три новых специализированных типа ФУ: точка фазового пространства, кластер, и менеджер. Менеджер отвечает за управление всеми ФУ, участвующими в вычислениях по алгоритму Mean Shift. Данный пример иллюстрирует та-

кой подход к ОА программированию: если функциональности существующих типов ФУ не хватает для реализации вычислительной задачи, то можно создать новый тип ФУ, благо в виртуальной ОА ВС он создается программно. В нашем случае код, реализующий поведение ФУ, был написан на языке C++.

Также как и для примера из раздела 4.1, из ФУ, ассоциированных с точками фазового пространства, была сгенерирована вычислительная сетка. Т.к. точки из множества T расположены в пространстве нерегулярно, наилучшим решением было применение триангуляции Делоне, когда сетка строится в виде треугольников [22]. ФУ-кластер хранит в своем внутреннем регистре ссылку на ФУ-точку ближайшую к его центру масс, и через эту точку производит опрос других точек пространства, запуская опросные токены по вычислительной сетке. Структура вычислительной сетки приведена на рис. 6.

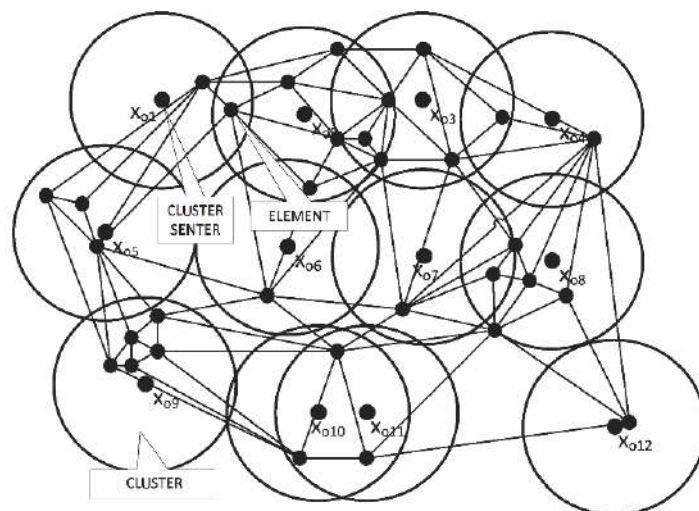


Рис. 6. Вычислительная сетка для реализации алгоритма Mean Shift

Для сбора данных от ФУ-точек ФУ-кластер связывается точкой, ближайшей к его центру масс и запускает вычислительную волну от нее для опроса близлежащих точек. Все точки, получившие опросный токен (прямой токен) и находящиеся внутри области кластера, посылают в начальную точку опроса обратный токен со сведениями о себе и передают опросный токен далее по вычислительной сетке. Если прямой токен приходит к точке за пределами кластера, то точка обратный токен не отправляет и прекращает дальнейшую передачу прямого токена, завершив тем самым продвижение опросной волны. Обратные токены собираются на ФУ-кластере. Исходя из собранных координат точек, ФУ-кластер может вычислить новый центр масс и перейти к новой итерации алгоритма (рис. 7)

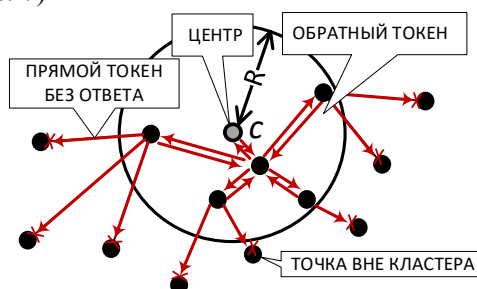


Рис. 7. Вычислительная сетка для реализации алгоритма Mean Shift

Созданные программные модули для реализации алгоритма Mean Shift были интегрированы в среду ОА программирования и моделирования, также была расширена функциональность компилятора ОА языка для обеспечения работы с новыми типами ФУ. После этого на ОА языке была реализована программа для имитационного моделирования вычислительного процесса. Результаты работы программы представлены на рис. 8, где иллюстрируется перемещение центров кластеров по 2-мерному фазовому пространству в направлении области с наибольшей концентрации

точек. В данном случае моделировалось движение сразу 16-ти кластеров, первоначально расположенных в виде прямоугольной сетки.

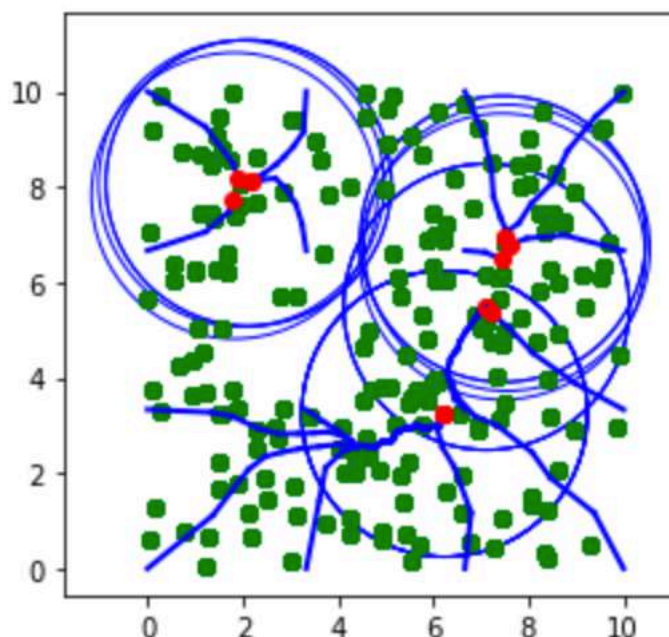


Рис. 8. Миграция 16-ти кластеров диаметром 5 на поле из 100 точек

5. Заключение

Описанный в статье ОА язык программирования представляется нам весьма перспективным и в том числе как средство описания алгоритмов для высокопараллельных и распределенных ВС. Это подтверждается тем, что он ориентирован на dataflow парадигму, которая обеспечивает самораспараллеливание вычислений. Как видно из приведенных фрагментов кода, ОА программа достаточно проста и понятна, что значительно упрощает процесс создания кода параллельной программы.

Язык, прежде всего, ориентирован на применение для ОА ВС, однако может быть использован и в качестве самостоятельного языка программирования. Язык дает большие преимущества: возможность организации сложных структур данных, удобное описание параллельных вычислений; кроме того, возможность создания и трансформации семантических сетей. Также ОА язык оказался весьма удобным для создания компиляторов языков высокого уровня и может быть применен подобно таким платформам, как LLVM [23]. Эффективность в области создания трансляторов языков подтверждается тем, что на ОА языке был написан его же компилятор, и в настоящее время создается компилятор 3-й версии.

ОА язык применяется для имитационного моделирования вычислительного процесса на базе ОА ВС, и он на практике показал свое удобство для создания параллельных программ. Так, программист не утруждает себя описанием вычислительных нитей и их синхронизацией – синхронизация происходит автоматически по данным непосредственно во время вычислительного процесса.

В настоящее время происходит создание компилятора новой версии, что повысит удобство программирования на ОА языке. Удобство будет обеспечено, прежде всего, за счет введения в ОА язык высокоуровневых конструкций: императивные конструкции, программные функции, арифметико-логические выражения. Для компилятора создается новая среда программирования на базе редактора VS Code, обеспечивающая подкраску синтаксиса программы, контекстные меню для выбора вариантов ввода, всплывающие подсказки, навигатор по информационным конструкциям, созданным в программе.

В дальнейшем предполагается применение ОА языка и среды программирования для него в научно-исследовательских целях, в частности, для имитационного моделирования вычислитель-

ного процесса на базе ОА ВС. Основной целью исследования будет выяснение емкостной и вычислительной сложности вычислительного процесса для решения различных прикладных задач. Вычислительные задачи для моделирования будут выбираться из разных областей: высокопараллельные и распределенные вычисления, системы автоматизации, системы искусственного интеллекта, анализа естественного языка [24], сеточные вычисления и т.д.

ОА язык сможет найти применение и для создания ВС на базе ПЛИС (FPGA). Для этого будет необходимо реализовать конвертер из ОА языка на языки схемотехнического моделирования (например, VHDL или Verilog), для которых существует САПР для записи программы на ПЛИС.

Литература

1. Milutinovic V., Salom J., Veljovic D. et al. DataFlow Supercomputing Essentials, Research, Development and Education. Springer, 2017.
2. Kasyanov V. Sisal 3.2: Functional language for scientific parallel programming // Enterprise Information Systems. 2013. Vol. 7, no. 2. P. 227–236. DOI 10.1080/17517575.2012.744854.
3. Легалов А.И. Функциональный язык для создания архитектурно-независимых параллельных программ // Вычислительные технологии. 2005. Т. 10, № 1. С. 71–89.
4. Veen H. Dataflow Machine Architecture. Center for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands. 1987.
5. Hewitt C., Bishop P., Steiger R. Universal Modular ACTOR Formalism for Artificial Intelligence // IJCAI. 1973. P. 235–245.
6. Wipliez M., Roquier G., Nezan J.F. Software Code Generation for the RVCCAL Language // Journal of Signal Processing Systems. 2011. Vol. 63, no. 2. P. 203–213. DOI: 10.1007/s11265-009-0390-zff.fhal-00407950f.
7. Климов А.В., Левченко Н.Н., Окунев А.С. Преимущества потоковой модели вычислений в условиях неоднородных сетей // Информационные технологии и вычислительные системы. 2012. № 2. С. 36–45.
8. Климов А.В., Окунев А.С. Графический потоковый метаязык для асинхронного распределенного программирования // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. № 2. С. 151–158.
9. Hammad A. Applying Automated Program Repair to Dataflow Programming Languages // 2021 IEEE/ACM International Workshop on Genetic Improvement (GI), 2021.
10. Салибекийн С.М., Панфилов П.Б. Вопросы автоматного-сетевого моделирования вычислительных систем с управлением потоком данных // Информационные технологии и вычислительные системы. 2015. № 1. С. 3–9.
11. Салибекийн С.М., Панфилов П.Б., Гончарук Г.Н. Объектно-атрибутивная модель программирования для параллельных вычислительных систем с управлением потоком данных // Параллельные вычислительные технологии 2013 (ПаВТ'2013): Труды международной научной конференции (Челябинск, 1-5 апреля 2013 г.). Челябинск: Издательский центр ЮУрГУ, 2013. С. 616.
12. Kavi K., Buckles B., Bhat U. A Formal Definition of Data Flow Graph Models // IEEE Transactions on Computers. 1986. Vol. C-35, no. 11. P. 940–948. DOI: 10.1109/TC.1986.1676696.
13. Салибекийн С.М. Трансляция языков высокого уровня, управляемая потоком // Информационные технологии. 2024. Т. 30, № 5. С. 261–268. DOI: 10.17587/it.30.261-268.
14. Оператор ?: (C#) URL: <https://msdn.microsoft.com/ru-ru/library/ty67wk28.aspx> (дата обращения: 28.01.2025).

15. Ключев А.О., Кустарев П.В., Ковязина Д.Р., Петров Е.В. Программное обеспечение встроенных вычислительных систем. СПб.: СПбГУ ИТМО, 2009. 212 с.
16. Тюрин А.Г., Зуев И.О. Кластерный анализ, методы и алгоритмы кластеризации // Вестник МГТУ МИРЭА. 2014. № 2 (3). С. 86–97.
17. Art L., Mauch H. Dynamic Programming A Computational Tool. Springer, Berlin, 2007.
18. Cai Y., Judd K.L., Thain G. et al. Solving Dynamic Programming Problems on a Computational Grid // *Comput Econ.* 2025. Vol. 45. P. 261–284. DOI: 10.1007/s10614-014-9419-x.
19. Salibekyan S.M., Vishnekov A., Ivanova E. Methodology of Mean Shift Clustering Algorithm Implementation Based on Dataflow Computer // *Proceedings of 2019 XVI International Symposium “Problems of Redundancy in Information and Control Systems” (REDUNDANCY)*. IEEE, 2019. P. 177–180. DOI: 10.1109/REDUNDANCY48165.2019.9003312.
20. Тюрин А.Г., Зуев И.О. Кластерный анализ, методы и алгоритмы кластеризации // Вестник МГТУ МИРЭА. 2014. № 2 (3). С. 86–97.
21. Comaniciu D., Meer P. Mean Shift: A Robust Approach Toward Feature Space Analysis // *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2002. P. 603–619.
22. Скворцов А.В. Триангуляция Делоне и ее применение. Томск: Изд-во Том. ун-та, 2002. 128 с.
23. Вьюкова Н.И., Галатенко В.А., Самборский С.В. LLVM как инфраструктура разработки компиляторов для встроенных систем // *Программная инженерия*. 2013. № 6. С. 2–10.
24. Салибекян С.М. Объектно-атрибутный подход для семантического анализа естественного языка // *Информационные технологии*. 2021. Т. 27, № 5. С. 267–274.

Использование сетей Петри для оптимизации параллельных бизнес-процессов

К.В. Самойлова¹, Е.Б. Замятина²

¹Пермский государственный национальный исследовательский университет,

²Пермский филиал Национального Исследовательского университета
«Высшая школа экономики»

В современной науке наблюдается тенденция к использованию методов, объединяющих сведения из различных областей знаний. Методы, изначально разработанные для решения задач в области параллельных и асинхронных систем, могут быть адаптированы для исследования бизнес-процессов, предоставляя новые возможности для повышения их эффективности. В статье рассматривается использование сетей Петри и систем массового обслуживания (СМО) для анализа параллельных бизнес-процессов на примере работы маркетплейса. Применение этих математических схем дает возможность выявить узкие места бизнес-процесса, а затем принять решение об устранении этих узких мест, используя методы онтологического моделирования. Новизна работы заключается в применении для оптимизации бизнес-процессов многомодельного подхода, объединяющего имитационное и онтологическое моделирование.

Ключевые слова: сети Петри, СМО, параллельный бизнес-процесс, онтологическое моделирование.

1. Введение

В последние годы наблюдается устойчивая тенденция к усложнению бизнес-процессов. Это приводит к увеличению расхода ресурсов, удлинению времени выполнения задач и снижению производительности, что, в свою очередь, является причиной разного рода ошибок и потерь. Традиционные методы анализа не всегда способны в полной мере исследовать бизнес-процессы, выявить их узкие места и предложить методы по усовершенствованию бизнес-процессов. В ответ на эти вызовы, в современной науке наблюдается тенденция к интеграции знаний из различных областей, что позволяет применять более эффективные и гибкие решения.

Цель данной статьи – исследовать возможности применения методов, изначально разработанных для параллельных и асинхронных систем для анализа и оптимизации бизнес-процессов. В частности, исследуется использование сетей Петри и систем массового обслуживания (СМО) для моделирования и анализа бизнес-процессов. Эти методы позволяют рассматривать процессы с разных точек зрения, выявляя их сильные и слабые стороны, и предлагать пути для улучшения этих процессов. Применение онтологического моделирования позволяет структурировать знания о процессах, принимать более обоснованные решения по их реорганизации и оптимизации. Этот многомодельный подход открывает новые возможности для повышения эффективности бизнес-процессов, учитывая как количественные, так и качественные аспекты. Его применение позволяет не только выявлять и устранять узкие места, но и оценивать влияние изменений в отдельных частях процесса на всю систему в целом. Использование имитационного моделирования дает более реалистичное представление о функционировании процессов в условиях неопределенности, снижает риски и помогает минимизировать возможные потери. В статье продемонстрировано применение этого подхода на примере работы маркетплейса.

2. Применение сетей Петри и СМО в анализе бизнес-процессов

Изначально разработанные как математический инструмент для моделирования систем с параллельными компонентами [1], сети Петри все чаще используются для анализа и оптимизации бизнес-процессов. Этот математический аппарат особенно эффективен при моделировании

сложных процессов, позволяя исследовать параллельные подпроцессы и их взаимное влияние друг на друга.

В работе [2] сделан вывод, что возможность анализировать достижимость и ограниченность отдельных участков сети Петри способствует выявлению потенциальных проблем и неэффективных элементов бизнес-процесса. Автор представляет алгоритм, позволяющий трансформировать бизнес-процессы, описанные в формате ARIS EPC, в одноцветные сети Петри. Автор отмечает, что графические методы представления моделей, хотя и были разработаны для упрощения описания и оптимизации бизнес-процессов, недостаточно удобны для проведения глубокого анализа. В то же время сети Петри обладают широкими аналитическими возможностями для исследования дискретных систем.

Предложенный алгоритм состоит из следующих шагов: каждому элементу модели EPC сопоставляется переход сети Петри, каждой связи – позиция; вводятся дополнительные позиции для обозначения начала и завершения процесса; логические операторы преобразуются в набор переходов и позиций. Такое преобразование позволяет проводить более глубокий анализ и оптимизацию бизнес-процессов, выявлять узкие места и тупики, а также определять все возможные варианты развития процесса.

Ван дер Аалст провел большое исследование, в котором сети Петри рассматриваются как мощный инструмент для формализации и оптимизации бизнес-процессов. В статье [3] основное внимание уделяется разработке и использованию подкласса сетей Петри, известного как Business-Procedure nets (BP-сети), которые специально адаптированы для представления и анализа процедур, лежащих в основе бизнес-процессов. Они имеют два специальных места: i (начало процесса) и o (конец процесса). Задачи моделируются переходами, а частичный порядок выполнения задач – местами, соединяющими эти переходы.

Одной из ключевых тем статьи является надежность (soundness) процедур, моделируемых с помощью BP-сетей. Надежность – это свойство, которое гарантирует, что для любого случая процедура завершится корректно, и в конце процесса в месте "о" будет находиться метка, в то время как все остальные места будут пустыми. Автор предлагает метод проверки звучания в полиномиальное время, что делает этот подход практически применимым для анализа сложных бизнес-процессов. Это свойство особенно важно в контексте реинжиниринга бизнес-процессов, где корректность и надежность процессов имеют первостепенное значение.

В работе [4] авторы выбрали использование сетей Петри для реинжиниринга бизнес-процессов. Они приходят к выводу, что для такого сложного и динамичного процесса как реинжиниринг, традиционных методов недостаточно. Сети Петри, напротив, могут быть адаптированы для моделирования различных аспектов бизнес-процессов, включая временные параметры, стохастические элементы и ресурсные ограничения. Авторы разработали программное обеспечение под названием PETROMAN, которое основано на стохастических цветных сетях Петри. Это программное обеспечение предназначено для управления сложными проектами, включая реинжиниринг бизнес-процессов.

В работе показан пример применения PETROMAN на процессе разработки продукта. Были смоделированы все этапы, начиная от создания чертежей и заканчивая испытаниями и подготовкой документации. Авторы провели симуляцию процессов, чтобы оценить их поведение в различных условиях. Например, в процессе разработки продукта они могли наблюдать, как распределяются ресурсы и как изменения в одном этапе процесса влияют на другие этапы. Они обнаружили, что определенные этапы, такие как подготовка чертежей и испытания, занимали больше времени, чем планировалось, что указывало на недостаточно четкую и правильную организацию процессов. Это позволило им предложить альтернативные варианты процесса, которые могли бы сократить время его выполнения и улучшить использование ресурсов.

Работа [5] исследует проблему количественного анализа бизнес-процессов. Авторы отмечают, что традиционные методы количественного анализа бизнес-процессов, такие как анализ потоков и симуляция, часто сталкиваются с компромиссом между точностью и скоростью. Анализ потоков быстр, но не учитывает вероятностные распределения, что снижает его точность. Симуляция, напротив, обеспечивает высокую точность, но требует значительных вычислительных ресурсов и времени, что делает ее неэффективной для оперативного анализа и оптимизации в реальном времени. Поэтому основная задача, которая решается в статье, заключается в разра-

ботке метода, который позволил бы быстро и точно оценивать временные характеристики бизнес-процессов, такие как ожидаемое время задержки, обработки и загрузка ресурсов. Авторы выбрали метод анализа на основе систем массового обслуживания из-за его способности сочетать быстродействие и точность. СМО позволяют моделировать сложные взаимодействия и вероятностные распределения, что делает их подходящими для анализа бизнес-процессов. Однако, существующие модели СМО часто имеют ограничения, такие как недостаточная поддержка параллельных задач и распределения ресурсов на основе ролей. Для решения этих проблем авторы работы [5] предложили новый метод, который включает следующие ключевые элементы: преобразование процессов из BPMN (Business Process Model and Notation – язык моделирования бизнес-процессов) в СМО, моделирование совместного использования ресурсов и алгоритм для анализа временных характеристик.

Для иллюстрации предложенного метода авторы приводят пример анализа сквозного процесса в телекоммуникационной компании. Этот процесс моделируется с использованием BPMN и затем преобразуется в сеть массового обслуживания. Результаты анализа показывают, что метод позволяет точно оценивать временные характеристики процесса и выявлять узкие места.

В статье [6] авторами так же ставится вопрос исследования и оптимизации времени выполнения бизнес-процессов. Основная цель – разработать метод оценки времени выполнения бизнес-процессов, который позволил бы оптимизировать операционные процессы в компаниях, особенно в условиях нестабильной и сложной бизнес-среды. Авторы предлагают математическую модель бизнес-процесса «Запрос-Ответ» в виде открытой экспоненциальной сети массового обслуживания с несколькими классами клиентов. Это позволяет моделировать различные типы запросов и их маршрутизацию через различные подпроцессы, такие как управление контактами, обработка запросов и проверка удовлетворенности клиентов.

С использованием формул теории СМО авторы работы [6] выводят выражения для расчета среднего времени пребывания заявки в системе, что соответствует времени выполнения бизнес-процесса. Для оптимизации времени выполнения предлагается использовать симплекс-метод линейного программирования. Это позволяет определить минимальное время выполнения процесса при заданных ограничениях на пропускную способность узлов. Предложенный метод проиллюстрирован на примере процесса «Запрос-Ответ» в телекоммуникационной компании, что подчеркивает применимость метода в реальных бизнес-сценариях.

3. Описание программного комплекса

Как было рассмотрено ранее, эффективное управление бизнес-процессами становится критически важным для успеха компаний, при этом традиционных методов недостаточно для их анализа и оптимизации. В данной работе предложен программный комплекс, предназначенный для исследования бизнес-процессов с использованием многомодельного подхода. В частности, будут использованы модели в основании, которых находятся такие математические формализмы как СМО и сети Петри. Онтологический подход, интегрированный в программный комплекс, обеспечивает принятие решений на основе экспертных знаний, что позволяет более точно и эффективно управлять бизнес-процессами.

Итак, первый шаг – это построение и валидация концептуальной модели M (концептуальной моделью считается модель, полученная в результате интервьюирования заказчика). После этого описывают модель бизнес-процесса M_x в одной из нотаций, используемых для моделирования бизнес-процессов (x – обозначение нотаций BPMN, IDEF, UML и пр.). Далее выполняют трансформацию модели бизнес-процесса M_x в имитационную модель M_yz , описанную с применением языка имитационного моделирования (здесь y – обозначение одного из языков имитационного моделирования, например GPSS, AnyLogic, NetLogo), основанного на математических формализмах (z), таких как системы массового обслуживания, сети Петри, цепи Маркова, теории графов и т.д. Трансформация проводится по определенным правилам с использованием онтологического подхода. После проведения трансформации необходимо убедиться, что полученная имитационная модель адекватно описывает изначально описанный бизнес-процесс. Для оценки достоверности имитационной модели происходит ее валидация с вычислением семантической близости онтологий имитационной модели и концептуальной модели бизнес-процесса [8]. При подтверждении валидности имитационной модели запускают имитационный эксперимент, в противном

случае имитационную модель проектируют заново. Если в ходе имитационного эксперимента выявляются риски, то из онтологии рисков извлекают знания о способах предотвращения рисков. Эти знания используют для модификации исходной модели M_x . Онтология рисков разрабатывается на основании экспертных знаний и тем самым помогает формализовать знания о процессе, интегрировать информацию о различных его аспектах и оценить эффект от возможных изменений. На рис. 1 представлен алгоритм описанного программного комплекса.

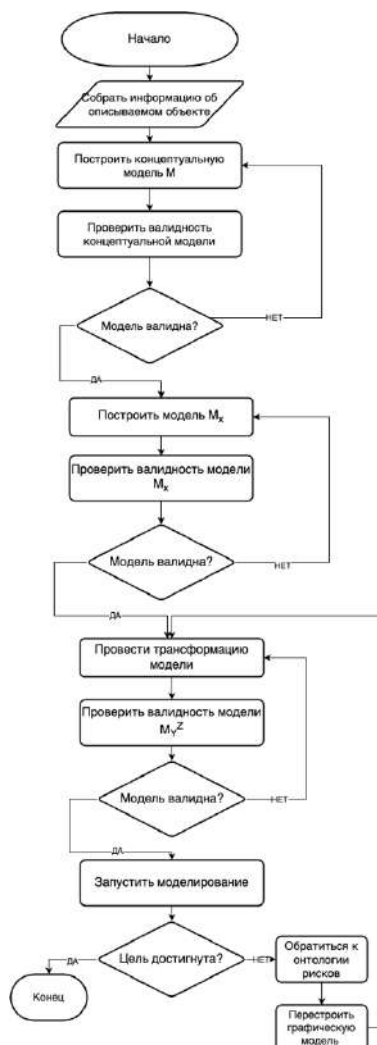


Рис. 1. Алгоритм определения надежности бизнес-процесса, выявления и устранения рисков

Таким образом, программный комплекс представляет собой мощный инструмент, обеспечивающий комплексный анализ бизнес-процессов в динамике. Отдельным преимуществом является то, что для применения предложенного комплекса нет жестких требований к использованию конкретной нотации или языка имитационного моделирования, что позволяет бизнес-аналитикам выбирать любые доступные инструменты.

4. Проектирование бизнес-процесса работы маркетплейса

Для демонстрации предложенного подхода рассмотрим пример бизнес-процесса работы маркетплейса. В процессе участвует 4 вида агентов: Поставщик товара, покупатель, который этот товар хочет купить и два работника (прием и сбор товара). Этот бизнес-процесс включает параллельные операции. В то время, когда поставщик предоставляет товар покупателю, покупатель может выбирать и складывать его в корзину в клиентском приложении. Для визуального представления работы маркетплейса выбрана нотация BPMN (рис. 2).

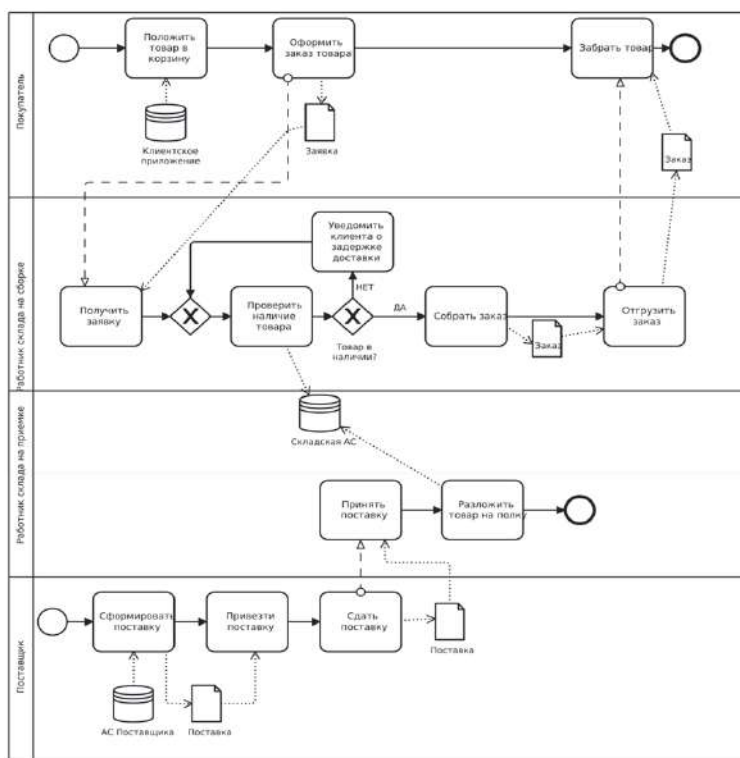


Рис. 2. Бизнес-процесс работы маркетплейса

Далее модель M_x рекомендуется преобразовать в имитационную модель M_y с использованием определенных правил трансформации [9]. В конкретном случае имитационная модель представлена в среде AnyLogic (математическая схема – система массового обслуживания (СМО)). Известно, что среднесуточный объем заказов составляет 100 единиц, поставок – 120. Сборка одного заказа занимает в среднем 1 час. Время доставки товара до клиента варьируется от 12 до 48 часов. Доставка до склада занимает от 12 до 120 часов при среднем значении 18 часов. Приемка и размещение товара на складе требует от 2 до 7 часов.

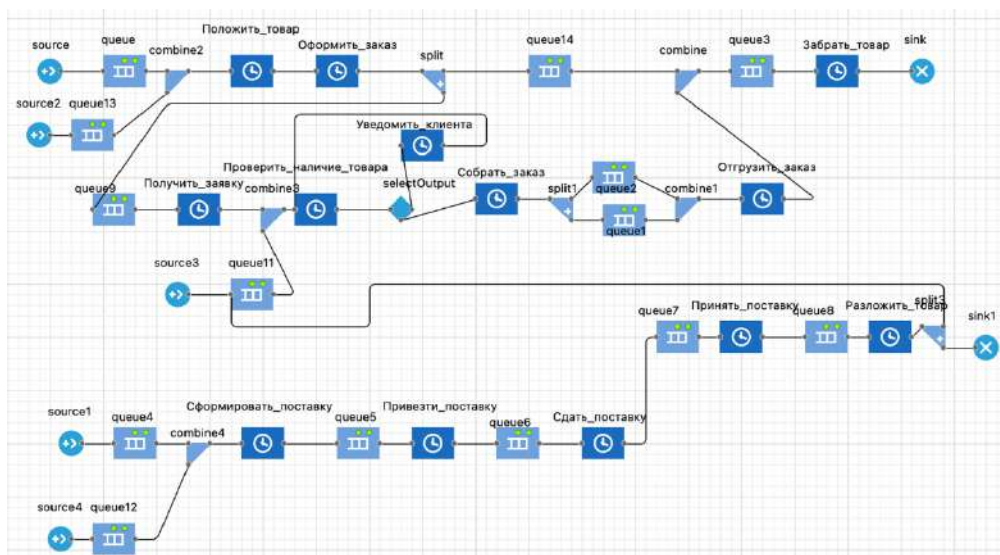


Рис. 3. Бизнес-процесс работы маркетплейса (СМО)

В результате имитационного эксперимента риски в бизнес-процессе не были выявлены и эксперимент остановлен после запуска максимально допустимого числа динамически создаваемых агентов. При этом по количеству агентов в разных каналах обслуживания видно, что все привезенные поставки были приняты, а из 7476 заявок на покупку товара только 8 отгружено.

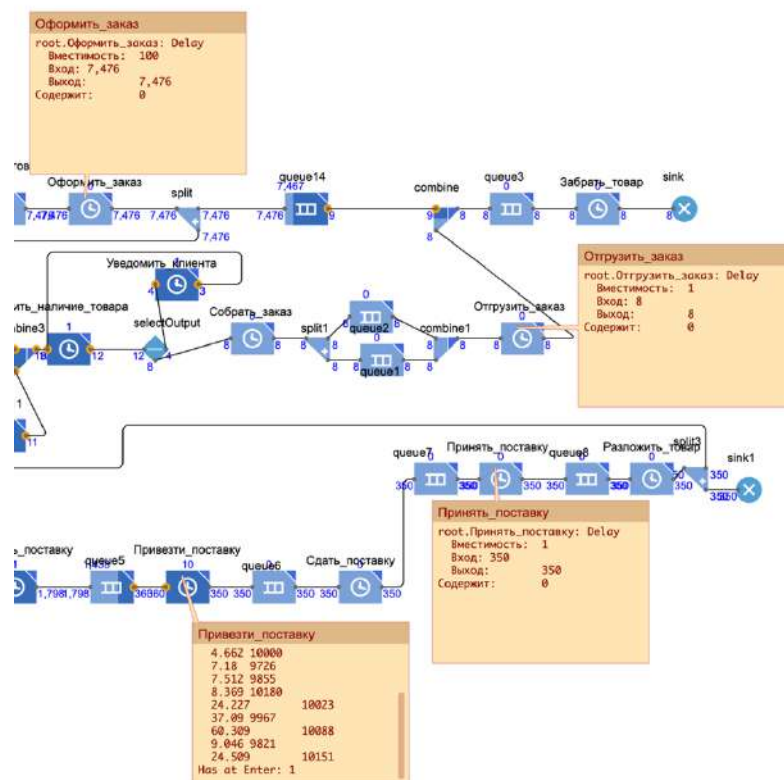


Рис. 4. Результат имитационного эксперимента

После анализа системы массового обслуживания, которая моделирует бизнес-процесс работы маркетплейса, становится очевидным, что он имеет сложную параллельную структуру. В частности, заказы от клиентов поступают и обрабатываются одновременно с поставками товаров от продавцов, что создает асинхронную среду, где несколько операций могут происходить одновременно. Для более глубокого исследования и визуализации таких параллельных процессов особенно подходят сети Петри. Изначально разработанные для моделирования асинхронных и параллельных систем, сети Петри позволяют наглядно представить взаимодействие между различными компонентами бизнес-процесса. Это делает их идеальным инструментом для выявления узких мест, конфликтов и других потенциальных проблем, которые могут возникнуть в процессе работы маркетплейса. Поэтому произведем трансформацию BPMN модели в имитационную модель, в основании которой лежат сети Петри. Представим модель на языке Netlogo [10]. Полученная сеть Петри представлена на рис. 5.

Начнем анализ модели, полученной в результате трансформации BPMN модели в сеть Петри. Методы анализа сетей Петри можно разделить на две группы, каждая из которых фокусируется на различных аспектах поведения и структуры сети [11].

К первой группе относится анализ поведенческих свойств. Сюда относится проверка сети на достижимость. В контексте бизнес-процессов это означает, что мы можем определить, возможно ли достичь определенного состояния (например, завершения заказа) из начального состояния. Анализ достижимости помогает убедиться, что все желаемые состояния процесса доступны и что процесс может быть завершен успешно.

К этой же группе относится проверка сети на живость. Живость сети Петри связана с отсутствием тупиковых ситуаций и гарантирует, что система может продолжать функционировать бесконечно без блокировок [12]. В бизнес-процессах это означает, что все переходы могут быть активированы в какой-то момент времени, и процесс не застрянет в определенном состоянии. Анализ живости помогает выявить потенциальные тупики и обеспечить бесперебойное выполнение процесса.

Еще один подход – проверка на наличие мертвых переходов. Мертвыми называются переходы, для которых не существует такого состояния сети, при котором переход срабатывает. Наличие таких переходов в сетях бизнес-процесса говорит о наличии неиспользуемых подпроцессов.

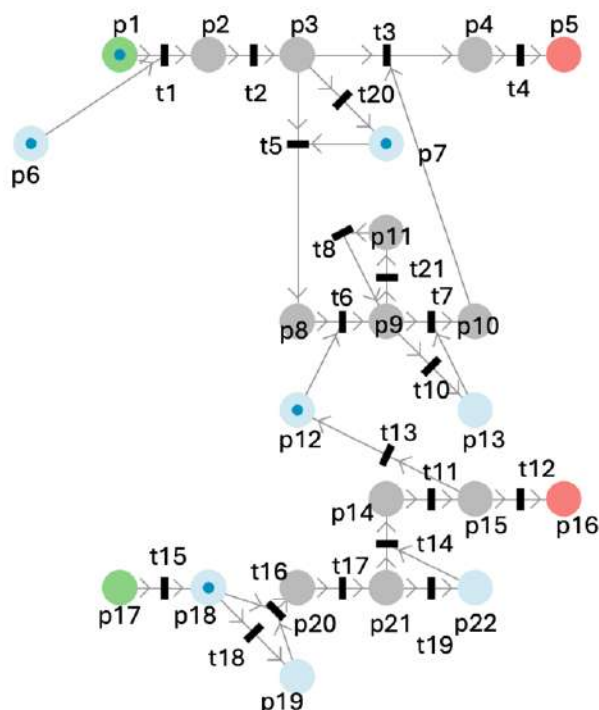


Рис. 5. Бизнес-процесс работы маркетплейса (сеть Петри)

Ко второй группе относятся методы, которые нацелены на выявление множества позиций и переходов, которые могут влиять на поведение системы. В работе [13] автор называет среди них ловушку, сифон и цикл (рис. 6).

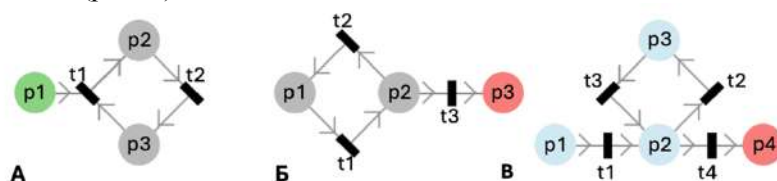


Рис. 6. Ловушка (А), Сифон (Б), Цикл (В)

Ловушка – это состояние, из которого нет переходов в другие состояния. Таким образом, маркер, попавший в ловушку больше, не может ее покинуть. Сифон – это набор позиций, из которого маркер, будучи туда помещенным, тем же переходом сразу извлекается. Поэтому маркер не может попасть в это состояние. Циклы – это замкнутые пути в сети Петри, где система может многократно проходить через одни и те же состояния. Необходимо отметить, что ловушка в отличие от цикла срабатывает при любой маркировке сети, а цикл только при определенной [13].

Итак, рассмотрим полученную сеть Петри бизнес-процесса работы маркетплейса. Для проверки на достижимость необходимо выбрать состояния сети, которых она должна достигнуть. Говоря о работе маркетплейса, бизнесу важно, чтобы продавец мог хранить свой товар на складе, а покупатель мог его купить. По результатам имитационного эксперимента видим, что в конечных позициях, которые представляют собой события «Принять заказ» и «Разложить товар» находятся маркеры, а значит это состояние достижимо (рис. 7).

При анализе сети на живость важно убедиться, что она способна выполняться бесконечно, то есть все переходы будут срабатывать в процессе выполнения. Для активации переходов необходимо, чтобы во входных позициях присутствовали маркеры. Следовательно, необходимо гарантировать, что все стартовые позиции всегда будут содержать маркеры.

Будем считать, что позиции p1 и p17 всегда содержат маркеры, так как они описывают процесс создания корзины клиентом и формирования поставки продавцом. Рассмотрим участок сети p1 – p6 – t1 – p2, который описывает процесс добавления выбранного товара в корзину с помощью клиентского приложения. Если позиция p6 не будет содержать маркер, то переход t1 не сможет сработать, и сеть не будет считаться живой (см. рис. 8). С точки зрения бизнес-процесса

это означает, что недоступность клиентского приложения представляет собой риск для выполнения всего бизнес-процесса.

Если же позиция p6 содержит маркер в ходе имитационного эксперимента срабатывает каждый переход, а значит они не мертвые.

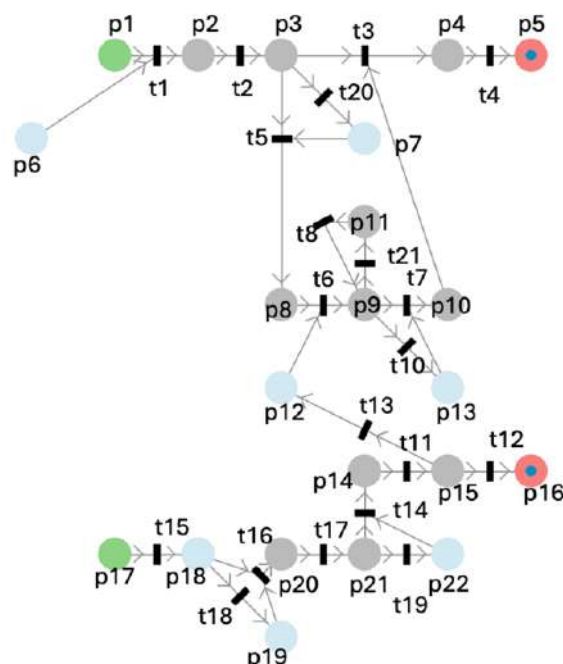


Рис. 7. Достижимость состояний

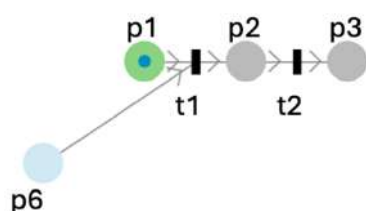


Рис. 8. Мертвый переход t1

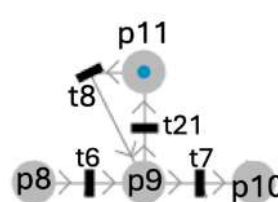


Рис. 9. Цикл

Так как в итоге все маркеры достигают конечных позиций, мы можем говорить об отсутствии ловушек и сифонов, так как они характеризуются застреванием маркеров в одном состоянии. Однако в рассматриваемой сети есть участок $t6 - p9 - t21 - p11 - t8 - p9$, который является циклом (рис. 9).

В бизнес-процессе данный участок отражает следующую ситуацию: после получения клиентского заказа происходит проверка наличия товара на складе. Если товар отсутствует, клиенту отправляется уведомление о задержке поставки, после чего проверка наличия товара повторяется. Этот цикл продолжается до тех пор, пока товар не будет готов к сборке. С точки зрения бизнеса, этот участок представляет собой узкое место, которое может препятствовать своевременному завершению бизнес-процесса и требует оптимизации.

Итак, при анализе полученной сети Петри были выявлены ключевые узкие места в бизнес-процессе работы маркетплейса. Одним из таких узких мест является участок сети, где после получения клиентского заказа происходит проверка наличия товара на складе. Этот цикл может привести к задержкам в обработке заказов и снижению удовлетворенности клиентов. Кроме того, существует риск, связанный с недоступностью клиентского приложения, что может привести к остановке процесса добавления товаров в корзину и, как следствие, невыполнению бизнес-процесса.

После того как в ходе исследования сети Петри бизнес-процесса были выявлены узкие места, необходимо обратиться в онтологию рисков (рис. 10). Онтология разработана на основе знаний эксперта в предметной области и состоит из 3 классов: «Риски», «Контрмеры» и «Последствия». Для извлечения экспертных знаний предлагается использовать SPARQL.

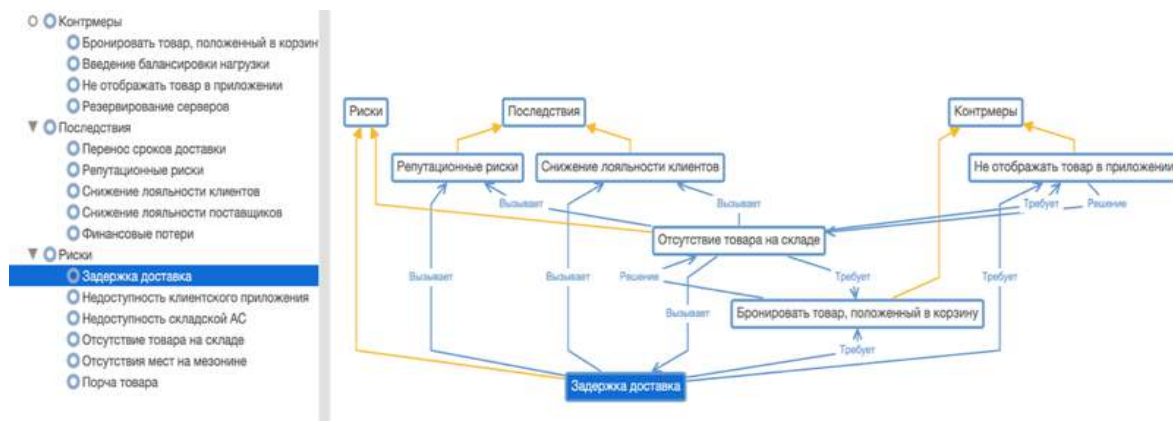


Рис. 10. Онтология рисков

Выявив риск недоступности клиентского приложения и отсутствия товара на складе, обращаемся к онтологии (рис. 11). В результате принимаем решение не отображать товар в приложении, если его нет на сайте.

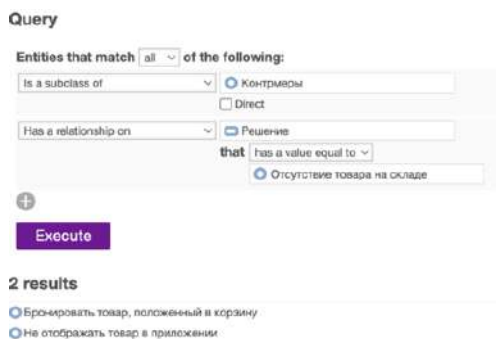


Рис. 11. SPARQL

Далее изменяем первоначальную модель в соответствии с принятым решением (рис. 12).

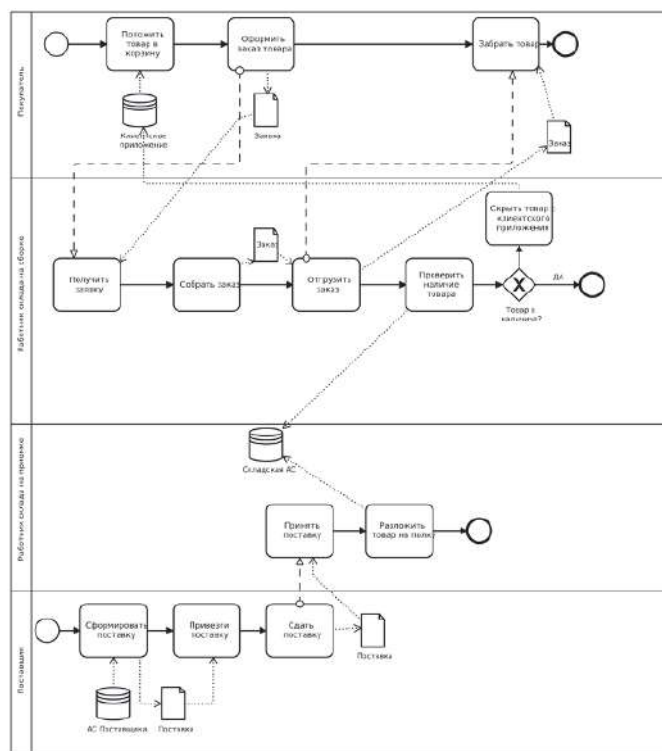


Рис. 12. Оптимизированный бизнес-процесс работы маркетплейса

5. Заключение

В работе рассмотрено применение сетей Петри и теории очередей для анализа и оптимизации бизнес-процессов. Исследование показало, что использование сетей Петри позволяет эффективно выявлять узкие места в бизнес-процессах и предлагать пути их оптимизации. Применение СМО открывает новые возможности для быстрой и точной оценки временных характеристик процессов, что позволяет повысить их эффективность и снизить затраты. Онтологический подход помогает принимать решения, основанные на экспертных знаниях с учетом особенностей процесса и контекста, в котором он происходит. В результате, предложенные изменения в моделях бизнес-процессов способствуют повышению их эффективности и снижению рисков, связанных с их выполнением.

Основное внимание было уделено исследованию и оптимизации бизнес-процессов с использованием сетей Петри, что позволило получить глубокое понимание их динамики и выявить ключевые узкие места. Сети Петри, благодаря своей способности моделировать параллельные и конкурентные процессы, оказались особенно эффективными для анализа сложных бизнес-процессов. Использование сети Петри позволило исследовать достижимость финального состояния бизнес-процесса, оценка на наличие мертвых переходов выявила потенциальное узкое место. Проверка сети на наличие определенного множества позиций и переходов, выявила цикл в процессе, который влияет на скорость работы всего бизнес-процесса в целом.

Итак, предложенный подход позволяет оптимизировать бизнес-процесс, минимизировать задержки и избежать блокировок. Итеративный процесс дает возможность анализировать, как изменения в одной части бизнес-процесса влияют на весь процесс в целом. Использование имитационных моделей помогает выявить узкие места, которые замедляют процесс или требуют чрезмерных ресурсов. Это позволяет получить более реалистичное представление о функционировании бизнес-процесса в реальных условиях, снизить риски и минимизировать последствия.

Литературы

1. Петерсон Дж. Теория сетей Петри и моделирование систем. М.: Мир, 1984. 264 с.
2. Доррер М.Г. Алгоритм преобразования моделей бизнес-процессов в одноцветные сети Петри // Моделирование и анализ информационных систем. 2010. Т. 17, № 2. С. 5–16.
3. van der Aalst W.M.P. A class of Petri nets for modeling and analyzing business processes // Computing Science Report. Vol. 9526. Eindhoven: Eindhoven University of Technology, 1995.
4. Somasundaram K., Krishnaiahchetty O. Managing Business Process Reengineering with Petri Nets. 2001. P. 280–285.
5. Peters S., Kerner Y., Dijkman R. et al. Fast and accurate quantitative business process analysis using feature complete queueing models // Information Systems. 2022. Vol. 104. P. 1–17. DOI: 10.1016/j.is.2021.101892.
6. Samouylov K., Gaidamaka Y., Zaripova E. Analysis of Business Process Execution Time with Queueing Theory Models // Communications in Computer and Information Science. 2016. P. 363–378. DOI: 10.1007/978-3-319-44615-8_28.
7. Zamyatina E., Churin D., Lanin V. et al. Simulation Model Validation Based on Ontological Engineering Methods // Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2022). KEOD. SciTePress, 2022. P. 237–244.
8. Ngom A., Kamara-Sangare F. Assessing Similarity Value between Two Ontologies // Proceedings of the 10th International Joint Conference on Knowledge Discovery, 2018. P. 343–350.
9. Kalenkova A.A., van der Aalst W.M.P., Lomazova I.A., Rubin V.A. Process mining using BPMN: relating event logs and process models // Software and Systems Modeling. 2015. Vol. 16, no. 4. P. 1019–1048.

10. Banos A., Lang C., Marilleau N. Agent-Based Spatial Simulation with NetLogo. ISTE Press, Elsevier, 2015. 222 p.
11. Мараховский В.Б., Розенблюм Л.Я., Яковлев А.В. Моделирование параллельных процессов. Сети Петри. СПб.: Профессиональная литература, 2014. 400 с.
12. Murata T. Petri Nets: Properties, Analysis and Applications // Proceedings of the IEEE. 1989. Vol. 77, no. 4. P. 541–580. DOI: 10.1109/5.24143.
13. Федоров И.Г. Метод отображения исполняемой модели бизнес-процесса в сети Петри // Статистика и экономика. 2013. № 4. С. 147–149.

Обработка потока задач с зависимостями на нескольких вычислительных кластерах

Тимофей Владимирович Санников, Алексей Николаевич Сальников

Московский государственный университет им. М.В. Ломоносова
timohaj1@yandex.ru, alexey.salnikov@gmail.com

Аннотация В работе рассматривается проблема планирования задач с зависимостями по данным между вычислительными кластерами. Исследуется разработка алгоритмов планирования задач: модификация жадного алгоритма backfill, GRAPHENE и Spear. Включает экспериментальное исследование на вычислительных кластерах. Ожидаемый результат — создание эффективных алгоритмов планирования и инструментария для оценки их производительности.

Ключевые слова: планирование · backfill · жадный алгоритм · grid

Введение

В современном мире перед исследовательскими группами встают задачи, требующие значительных вычислительных ресурсов. Для обработки таких сложных и ресурсоемких вычислений применяются вычислительные кластеры — группы компьютеров, объединенных высокоскоростной сетью с низкими задержками.

Основными характеристиками вычислительного кластера являются его вычислительная мощность, измеряемая в FLOPS, а также количество одновременно задействованных вычислительных устройств: процессорных ядер, графических сопроцессоров и других компонентов.

Примеры задач, требующих значительных вычислительных ресурсов:

- **Исследования в медицине (например, изучение работы мозга)** [6]. В рамках этих исследований регистрировались электрокардиограммы (ЭКГ) у 30 испытуемых в различных условиях. Обработку полученных данных требовалось выполнить с высокой скоростью и точностью.
- **Моделирование физических процессов.** В этой области вычислительные кластеры используются для симуляции сложных явлений, таких как турбулентность в жидкостях, распространение электромагнитных волн и прочие задачи, требующие высокой точности расчетов. Примером такой задачи является [7], в рамках которой происходил модельный эксперимент, требующий большого количества вычислительных ресурсов.
- **Обучение нейросетей.** Для обучения многих нейросетей, необходимо использовать вычислительные кластеры. Обучение в исследованиях зачастую происходит на различных данных, соответственно появляются множество задач на обучение, которые необходимо распределить по различным вычислительным кластерам [1].

Для работы с вычислительными кластерами пользователи получают удаленный доступ к узлам системы через протокол SSH (Secure Shell). Это позволяет выполнять вычисления, запускать программы и управлять задачами, находясь на другом компьютере. Через SSH можно подключаться как к управляющему узлу, который распределяет задачи между вычислительными узлами, так и непосредственно к отдельным узлам, если это требуется. Такой подход обеспечивает безопасный и эффективный способ взаимодействия с кластерными ресурсами.

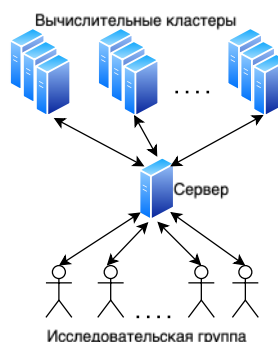
Помимо классических вычислительных кластеров, существуют Grid-системы, которые позволяют объединять распределенные вычислительные ресурсы, находящиеся в разных географических зонах. В отличие от традиционных кластеров, где задачи исполняются на локально связанных узлах, Grid-системы используют более гибкий подход, объединяя мощность нескольких независимых кластеров и серверов. Однако большинство существующих Grid-систем обладают высокой сложностью в установке, настройке и использовании, требуя значительных временных и технических затрат. Кроме того, их использование зачастую требует куда большего доступа, чем тот, который предоставляется по средствам протокола SSH, что ограничивает их применение для широкого круга пользователей.

Дополнительной сложностью является наличие зависимостей между задачами. В реальных сценариях выполнения вычислений часто требуется, чтобы одна задача начиналась только после завершения другой. Это приводит к необходимости управления процессами таким образом, чтобы учитывать порядок выполнения, минимизировать задержки и оптимально использовать вычислительные ресурсы.

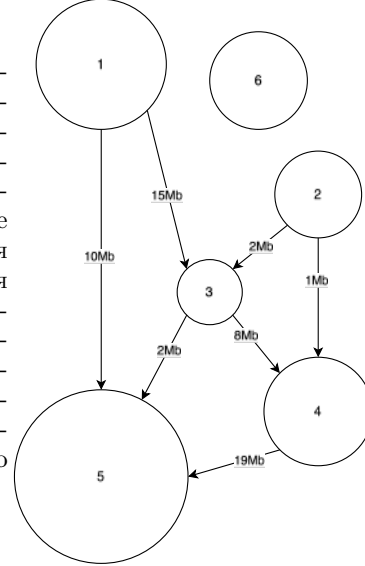
Таким образом, возникает необходимость в создании инструмента, который позволит упростить работу с Grid-системами и автоматизировать управление зависимостями между задачами. Такой инструмент должен снижать сложность конфигурации, быть адаптированным под работу через протокол SSH и предоставлять эффективные механизмы распределения вычислительных ресурсов.

1 Описание проблемы использования нескольких кластеров

Нами рассмотрен только частный случай, когда группа ученых устанавливает у себя в организации сервер, на который приходят запросы на выполнение задач со стороны клиентов (рабочих мест сотрудников). Сервер определяет, на каком вычислительном кластере будет выполнена задача, и отправляет необходимые данные и команду для постановки задачи в очередь на соответствующем кластере.



Поток задач — набор задач, которые поступают на сервер от исследовательской группы. У каждой задачи есть необходимые ей ресурсы (количество потоков), максимальное время выполнения задачи, необходимые для ее выполнения файлы и функция для вычисления времени выполнения задачи на выбранном кластере (опционально). Для представления потока задач используется представление направленного графа в виде файла формата json, который загружается в систему и привязывается к полю topology базы данных задач.



Для достижения меньшего времени выполнения потока задач, необходимо учитывать скорость соединения между сервером и кластером, так как перед постановкой задачи в очередь на кластере возможна передача файлов.

В рамках данной работы будем минимизировать время выполнения всего потока. Для оценки времени выполнения конкретной задачки с момента ее постановки будем использовать следующую функцию (1):

$$\min_{j \in \mathcal{J}} \left(T_{\text{total}}^j \right) \quad (1)$$

$$T_{\text{total}}^j = \max_{i \in \mathcal{I}} \left(t_{\text{all}}^i \right), \quad (2)$$

где для каждой задачи i :

$$t_{\text{all}}^i = t_{\text{start}} + t_{\text{tr}}^i + t_{\text{que}}^i + t_{\text{exec}}^i,$$

$$t_{\text{tr}}^i = \frac{S_i}{v} + t_{\text{tr\_prev}}^i.$$

Ограничения:

- Порядок выполнения, определяемый зависимостями: $j \prec i \Rightarrow t_{\text{all}}^j \leq t_{\text{start}}^i$.
- Запрет на превышение ресурсов кластера: $\sum_{i \in C_k} r_i \leq R_k, \forall k$.
- Запрет гранулярности выполнения задачи: $x_{ik} \in \{0, 1\}$.

Выполнение на разных кластерах возможно только при наличии всех необходимых файлов и ПО для обмена текущим состоянием задачи, включая оперативную память. Учитывая ограниченный доступ к кластеру без возможности просмотра памяти и необходимость передачи файлов, это увеличит время, необходимо выполнять задачу только на одном кластере (в обзоре назовем этот пункт целостностью).

Обозначение	Описание
$\mathcal{J}$	Множество расписаний
$\mathcal{T}$	Множество всех задач
T_{total}^j	Общее время выполнения потока при расписании j
t_{all}^i	Полное время выполнения задачи i
t_{tr}^i	Время передачи данных для задачи i
t_{exec}^i	Время выполнения задачи i на кластере
t_{que}^i	Время ожидания задачи i в очереди
S_i	Объем данных задачи i (бит)
v	Скорость передачи данных (бит/с)
$t_{\text{tr}}^i_{\text{prev}}$	Время окончания передачи данных предыдущих задач на кластер
$j \prec i$	Задача i зависит от задачи j
R_k	доступные ресурсы кластера k
r_i	ресурсы, требуемые задачей i
x_{ik}	Задача i выполняется на вычислительном кластере k

Для достижения эффективного распределения ресурсов необходимо учитывать:

- зависимости между задачами;
- время выполнения задачи;
- время передачи данных, необходимых для выполнения задачи;
- время в очереди, с учетом других задач;
- целостность задачи.

2 Обзор существующих алгоритмов планирования в вычислительных кластерах

2.1 Описание алгоритмов планирования

Federated Scheduling Алгоритм, который описан в статье [4], разделяет задачи на две группы: быстрозавершаемые и долгосрочные. Это позволяет уменьшить задержки выполнения и эффективно распределять ресурсы кластера. Планирование происходит относительно локальных анализаторов каждого кластера. Данный метод снижает конкуренцию между разными типами задач и увеличивает общую пропускную способность системы.

Spear Scheduling Алгоритм из статьи [3] использует нейросетевую модель, которая обучается в процессе работы и адаптируется к текущему состоянию кластера. Такой метод позволяет предсказывать оптимальное распределение задач и минимизировать задержки, возникающие из-за неоптимального выбора узлов.

Bucket Algorithm Метод планирования, описанный в статье [5], предполагает перераспределение задач через регулярные промежутки времени с учетом изменяющихся параметров системы. Такой подход особенно эффективен в средах с высокой изменчивостью нагрузки и позволяет минимизировать простои вычислительных ресурсов.

GRAPHENE Алгоритм статьи [2] сочетает часть принципов Federated Scheduling и Bucket Algorithm, а также учитывает специфику кеширования данных. В контексте вычислительных кластеров оптимизирует процесс передачи файлов между сервером и узлами, что позволяет значительно повысить эффективность работы системы.

Модифицированный Backfill Алгоритм статьи [8] является усовершенствованной версией классического Backfill, включающей учет времени передачи данных и времени в очереди. В отличие от стандартного Backfill, он анализирует не только свободные временные интервалы в расписании, но и ожидаемые задержки, связанные с передачей файлов и завершением зависимых задач. Такой подход позволяет более точно прогнозировать время начала выполнения задач, минимизировать простои и увеличить загрузку вычислительных узлов. Но в нем нет учета зависимостей между задачами.

2.2 Сравнение алгоритмов планирования

Таблица 1: Соответствие алгоритмов поставленным требованиям

Название алгоритма	Учет зависимостей между задачами	Учет времени работы	Учет времени передачи	Учет времени в очереди	Учет времени в других задачах	Целость задачи	Реализуем для нашего случая
Federated Scheduling	Да	Да	Нет	Нет	Да	Да	Да
Spear Scheduling	Да	Да	Нет	Нет	Да	Да	Да
Bucket Algorithm	Да	Да	Нет	Нет	Да	Да	Да
GRAPHENE	Да	Да	Нет	Нет	Да	Да	Да
Modified Backfill	Нет	Да	Да	Да	Да	Да	Да

Из таблицы 1 видно, что большинство алгоритмов не учитывают время передачи данных. Все алгоритмы можно модифицировать так, чтобы они учитывали время передачи данных. Модифицированный BackFill не готов к использованию при наличии потока задач и нацелен на минимизацию времени выполнения конкретной задачи, а не всего потока. Для решения этих проблем, необходимо объединить его с идеей алгоритма Bucket о перераспределении задач, а также добавить учет зависимостей между задачами.

Для реализации были выбраны алгоритмы:

- **GRAPHENE**, из-за преимуществ в скорости расстановки по сравнению с Bucket. Дополнительной модификацией будет учет времени перердачи данных и времени в очереди;
- **Spear**, в котором при проходе по дереву поиска будет дополнительно учитываться время передачи и время в очереди. А DRL будет оценивать текущее состояние кластеров и, основываясь на этом, изменять шансы на выбор кластера;

- Модифицированный Backfill, с учетом новых модификаций, позволяющих делать расстановку с учетом зависимостей.

3 Обход графа потока задач

В нашем случае обход графа осуществляется посредством изменения статуса задач. На графиках 1:

- **Темнозеленый** – задача выполнена;
- **Светлозеленый** – задача готова к выполнению и соответственно распределению на кластер;
- **Оранжевый** – задача учитывается при построении расписания, но не ставится на выполнение (в случае возможности отправки данных для задачи, отправка данных происходит).

Идея заключается в том, что при ярусном обходе графа количество «светлозеленых» и «оранжевых» задач достаточно ограничено, чтобы произвести полный перебор вариантов расстановки среди них. Причем приоритет выполнения всегда у «светлозеленых» задач. То есть в случае рисунка 1b будут рассматриваться только 3 варианта постановки задач в планировщик:

- $6 \rightarrow 3 \rightarrow 4$;
- $3 \rightarrow 6 \rightarrow 4$;
- $3 \rightarrow 4 \rightarrow 6$.

В последующих главах темнозеленый и светлозеленый уровни светлозеленый ярусы будут называться 1 и 2 ярусами соответственно.

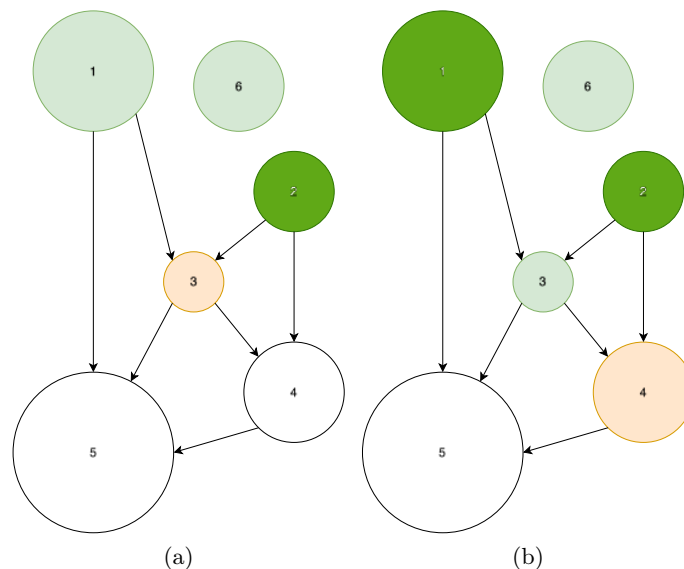


Рис. 1: Графики обработки вычислений

4 Новый алгоритм, основанный на BackFill

Модифицированный алгоритм Backfill включает в себя несколько улучшений для учета времени передачи файлов и времени нахождения задач в очереди.

4.1 Учет времени передачи необходимых файлов

- Перед постановкой задачи в очередь оценивается **время передачи файлов**, необходимых для выполнения этой задачи.
- Пока файлы не переданы, задача не может быть поставлена в очередь на выполнение.
- В алгоритме появляется **минимальное время начала выполнения задачи** – она не может стартовать раньше, чем все ее данные будут доступны.
- После определения этого минимального времени применяется стандартный алгоритм **Backfill**, но с учетом найденного ограничения.
- Если файлы уже находятся в нужном вычислительном кластере, задача может быть распределена сразу, как в классическом **Backfill**.

4.2 Учет времени в очереди

Введем обозначения:

- $t_{\min}$ – минимальное время, после которого задача может быть запланирована;
- t_{tr} – время окончания передачи файлов;
- $t_{que\ this}$ – время нахождения задачи в очереди (без учета задач, которые еще не завершили передачу файлов).

После завершения передачи всех файлов задача ставится во **внутреннюю очередь кластера**. Минимальное время начала выполнения определяется как:

$$t_{\min} = t_{tr} + t_{que\ this}. \quad (3)$$

Это позволяет **точнее оценить** время выполнения задачи, учитывая как передачу данных, так и загрузку вычислительных ресурсов. В результате выполняется **двухуровневое планирование**:

1. сначала на сервере, где выбирается подходящий кластер;
2. затем внутри самого вычислительного кластера.

4.3 Применение алгоритма для нескольких вычислительных кластеров

В случае, если задействовано несколько вычислительных кластеров, для каждого из них создается отдельный **планировщик**. Планировщики работают по улучшенному алгоритму **Backfill** и управляются с единого сервера.

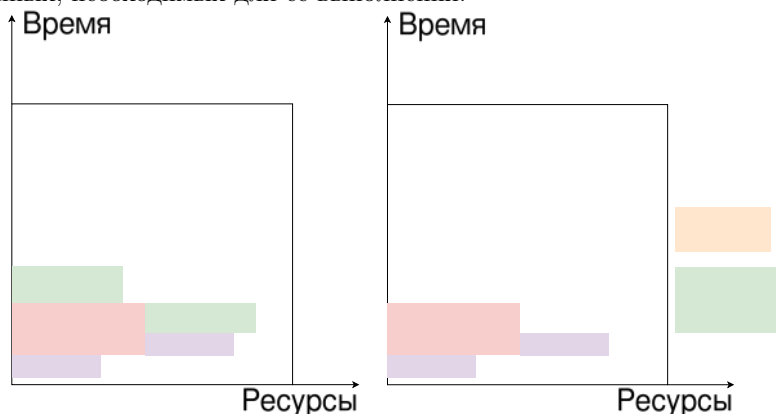
Для каждой задачи проверяется, через сколько времени она сможет стартовать на каждом кластере. Задача отправляется в тот кластер, где **время старта будет минимальным**.

4.4 Минимизация времени выполнения потока и учет зависимостей между задачами

За основу возьмем идею алгоритма Bucket, раз в определенный промежуток времени будем перераспределять задачи по различным кластерам.

4.5 Демонстрация работы алгоритма

Продemonстрируем работу алгоритма в ситуации, если поставленная зеленая задача, имела зависимости в момент, когда ее поставили. Тогда сразу после выполнения одной из других задач (не обязательно на этом кластере) или поступлении новой задачи, происходит перераспределение задач по кластерам. Причем, если данные уже были отправлены, у нее больший шанс попасть на определенный ранее кластер из-за отсутствия времени передачи данных, необходимых для ее выполнения.



5 GRAPHENE

5.1 Доработанный алгоритм Graphene

Алгоритм **Graphene** предназначен для эффективного управления потоками задач в вычислительных кластерах. Основная идея Graphene заключается в том, чтобы минимизировать время ожидания задач за счет:

- динамического прогнозирования доступности ресурсов;
- разделения задач по их вычислительным характеристикам;
- анализа зависимостей между задачами и передачи необходимых файлов перед их выполнением.

5.2 Основные принципы работы алгоритма Graphene

В отличие от Backfill, алгоритм Graphene не просто заполняет пробелы в расписании, а активно формирует **граф зависимостей** задач и ресурсов. Основные этапы работы:

1. **Анализ входящих задач:** оценивается объем вычислений и потребность в ресурсах.
2. **Разделение задач на группы:** задачи классифицируются на **большие** и **малые** (см. раздел 5.3).
3. **Определение зависимостей:** формируется граф, в котором учитываются очередность выполнения задач и их потребность в данных.
4. **Прогнозирование выполнения:** оценивается, когда ресурсы станут доступными для каждой задачи.
5. **Планирование с учетом передачи данных:** задачи ставятся в очередь на выполнение с учетом времени их передачи и доступности вычислительных мощностей.

5.3 Разделение задач на большие и малые

Важной особенностью Graphene является **разделение задач** на две группы:

- **Большие задачи** – требующие значительных вычислительных мощностей и большого объема данных.
- **Малые задачи** – менее ресурсоемкие, но потенциально более частые.

Критерии разделения:

- Большая задача:

$$\alpha \cdot t_{\text{exec}_i} + \beta \cdot R_i > T_{\text{threshold}} \quad w_i = \begin{cases} \alpha \cdot \left(t_{\text{exec}_i} + \frac{S_i}{v} \right) + \beta \cdot R_i, & \text{большая} \\ \gamma \cdot \left(\frac{S_i}{v} + \delta \cdot D_i \right), & \text{малая} \end{cases}$$

- Малая задача: иначе

Переменная	Описание
w_i	Вес задачи i
t_{exec_i}	Время выполнения (с)
S_i	Объем данных (бит)
v	Скорость передачи (бит/с)
D_i	Число зависимостей
R_i	Ресурсы (ядра/память)

$\alpha, \beta, \gamma, \delta$ — константы.

Такой подход необходим по следующим причинам:

1. **Эффективность использования ресурсов:** крупные задачи могут блокировать очередь, а малые задачи позволяют заполнять промежутки между большими вычислениями.
2. **Минимизация задержек:** мелкие задачи выполняются быстрее, не мешая работе крупных задач.
3. **Гибкость планирования:** зная, какие задачи являются приоритетными, алгоритм может оптимально распределять вычислительные мощности.

Приоритет выполнения задач определяется на основе анализа графа зависимостей. Если малые задачи не мешают выполнению крупных, они могут быть запущены параллельно и на одном вычислительном кластере.

5.4 Применение алгоритма для нескольких вычислительных кластеров

Graphene позволяет управлять задачами на нескольких вычислительных кластерах. Процесс организации планирования основан на анализе текущей загрузки кластеров:

1. Каждый локальный планировщик анализирует загрузку своего кластера и прогнозирует доступность ресурсов.
2. Глобальный координатор собирает информацию о загрузке кластеров и принимает решение о назначении задачи.
3. Если требуется передача данных между кластерами, алгоритм заранее оценивает ее длительность и учитывает при принятии решений.

То есть на сервере не хранится уже составленное расписание, а распределение происходит в зависимости от текущего состояния кластеров.

6 Spear

6.1 Общая концепция

Модифицированный алгоритм Spear предназначен для оптимального планирования задач в распределенной вычислительной среде с учетом зависимостей между задачами и неоднородных требований к ресурсам. В основе алгоритма лежит комбинация *Монте-Карло Дерева Поиска (MCTS)* и *Глубинного обучения с подкреплением (DRL)*, что позволяет динамически адаптироваться к изменяющимся условиям кластера и минимизировать время выполнения всего задания (makespan).

6.2 Процесс планирования

Процесс планирования задач состоит из следующих этапов:

1. Формирование состояния кластера

На основе текущего состояния вычислительных узлов и DAG-графа формируется пространство решений, в котором учитываются доступные ресурсы, загрузка системы и незавершенные задачи.

2. Монте-Карло Дерево Поиска (MCTS)

MCTS используется для поиска наиболее оптимального расписания выполнения задач:

- **Выбор (Selection)**: выбор наиболее перспективного узла дерева решений на основе эвристики *Upper Confidence Bound (UCB)*.
- **Расширение (Expansion)**: создание новых возможных состояний системы путем назначения очередной задачи на вычислительный узел.
- **Симуляция (Simulation)**: оценка возможного времени завершения всего задания путем моделирования выполнения оставшихся задач.

- **Обратное распространение (Backpropagation)**: обновление значений узлов дерева на основе полученных данных.

3. Использование глубинного обучения с подкреплением (DRL)

Чтобы ускорить процесс планирования, в Spear используется обученная нейросетевая модель, предсказывающая наиболее выгодные ходы при выборе задач. DRL позволяет:

- оптимизировать процесс поиска решений за счет предсказания вероятностного распределения возможных действий;
- улучшить симуляцию MCTS, заменяя случайный выбор решений на более осмысленный, основанный на обученных данных;
- минимизировать время вычислений, сокращая число ненужных проверок и переборов.

6.3 Адаптация под вычислительные кластеры

В модифицированной версии Spear учтены особенности работы в распределенной системе:

- Учитываются неоднородные требования к ресурсам.
- Планирование выполняется в реальном времени, минимизируя задержки при принятии решений.
- Алгоритм адаптируется к разным нагрузкам за счет динамической настройки параметров поиска.
- Включены дополнительные эвристики для предотвращения «голодания» задач, зависящих от узких мест в кластере.
- Для передачи данных для кластера и времени в очереди создается отдельный параметр (минимальное время старта), который участвует в расчетах MCTS, а DRL, получает дополнительным параметром вектор значений времен передачи всех необходимых задач данных для каждого кластера. Из-за чего происходит дополнительная корреляция, позволяющая влиять времени передачи данных на предсказание лучшего распределения.

7 Программная реализация

7.1 Инструмент постановки задачи на выполнение

Для работы со всеми алгоритмами была разработана система постановки задач в очередь, которая производит отправку необходимых файлов на вычислительные кластеры, ставит задачи во внутренние очереди вычислительных кластеров и обновляет статусы задачи (ожидает выполнения → отправляется на кластер → в очереди на выполнение → выполняется → завершена).

Хранение всех необходимых данных происходит в базе данных PostgreSQL (см. рис. 2).

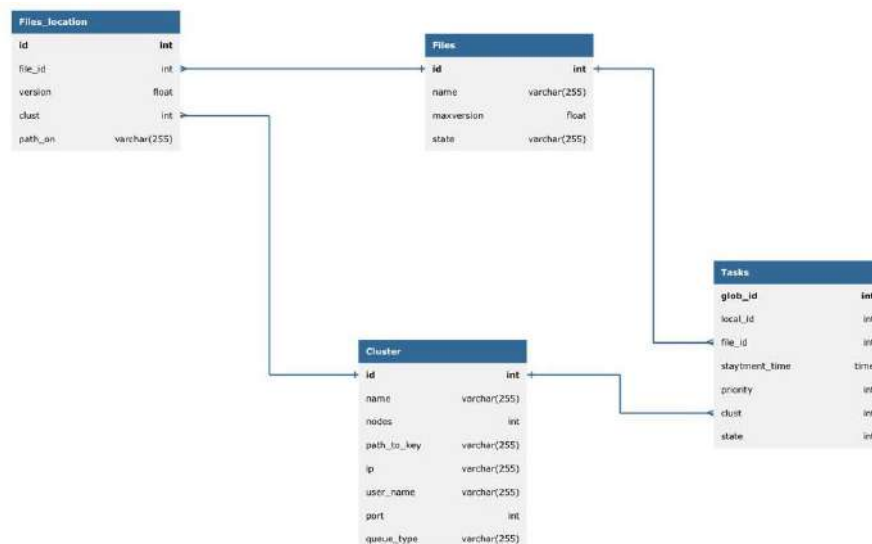


Рис. 2: Схема базы данных

Инструмент для работы с кластерами был написан на языке python, работа с кластером происходит через протокол SSH с использованием библиотеки paramiko.

7.2 Планировщики

Для оптимизации времени обработки потока, написание поланировщиков производилось на языке C++. Планировщики вызываются из Python через **библиотеку ctypes**, обеспечивая минимальные накладные расходы на межъязыковое взаимодействие.

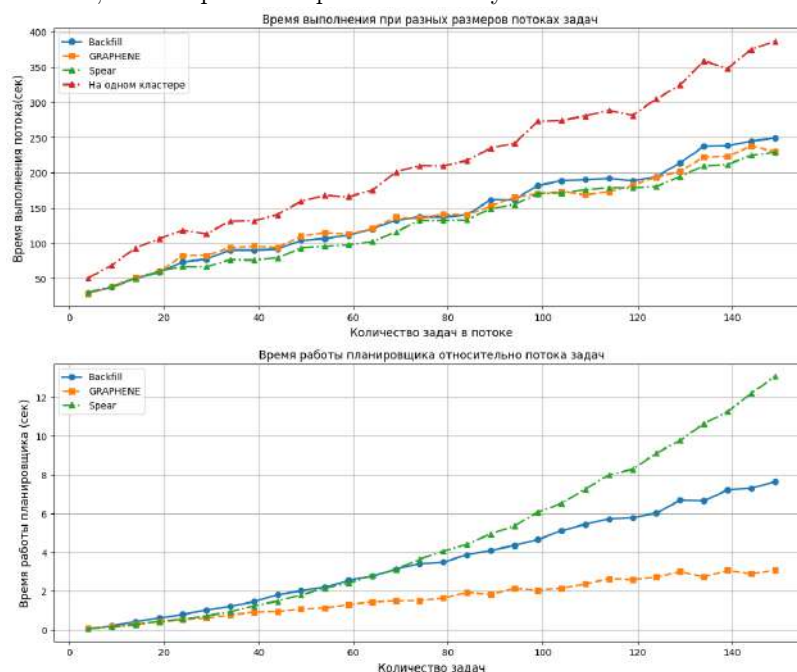
Обмен данными реализован через protobuf, что позволяет значительно уменьшить потери производительности при передаче данных задач и состояний кластеров.

8 Результаты тестирования

Для тестирования эффективности алгоритмов был написан генератор задач. Производилась релаксация матриц разного размера до достижения определенного качества.

Генерировались потоки задач, связанных друг с другом файлом вывода, в который отгружались рассчитанные матрицы. Таким образом можно было проверить корректность работы алгоритма.

Планировщики тестировались в одинаковых условиях, на одинаковых потоках задач. Для тестирования были запущены два компьютера с включенным slurm в режиме отладки, что позволило выставить необходимое число потоков, не смотря на их физическое отсутствие.



В результате тестирования было выявлено, что в большинстве ситуаций алгоритм spear показывает себя лучше всего, но тратит наибольшее количество времени на вычисления. Так что в некоторых ситуациях следует задуматься об использовании алгоритма GRAPHENE.

9 Заключение

В данной работе были рассмотрены вопросы планирования вычислительных задач в среде распределенных вычислительных кластеров. Проведен анализ существующих алгоритмов, таких как Federated Scheduling, Spear Scheduling, Bucket Algorithm, GRAPHENE и модифицированный Backfill, выявлены их преимущества и недостатки.

Для решения проблемы минимизации времени выполнения потока задач с зависимостями предложены модификации алгоритмов. В частности, был разработан новый алгоритм, основанный на Backfill, с учетом времени передачи файлов и времени в очереди на выполнение. Также была доработана схема планирования в алгоритме GRAPHENE и разработана новая формула оценки веса задачи, учитывающая необходимые данные. Кроме того, адаптация алгоритма Spear позволила использовать методы Монте-Карло

Дерева Поиска и глубинного обучения с подкреплением для динамической оптимизации планирования так, чтобы в них учитывались важные параметры при планировании между несколькими вычислительными кластерами.

Лучше всех в плане эффективности расстановки показал себя алгоритм Spear, по времени работы алгоритма лидером стал GRAPHENE, Backfill показал баланс между эффективностью и скоростью работы планировщика. Так что каждый из реализованных алгоритмов может оказаться лучшим для минимизации времени выполнения потока в зависимости от самого потока задач и характеристик сервера.

Таким образом, реализации предложенных алгоритмов позволили повысить эффективность использования вычислительных ресурсов, учесть зависимости между задачами и минимизировать задержки, связанные с передачей данных и ожиданием в очереди на выполнение.

Список литературы

1. Chan, E.R., Monteiro, M., Kellnhofer, P., Wu, J., Wetzstein, G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis (2014). Stanford University
2. Grandl, R., Kandula, S., Rao, S., Akella, A., Kulkarni, J.: Graphene: Packing and dependency-aware scheduling for data-parallel clusters. In: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16). USENIX Association (2016)
3. Hu, Z., Tu, J., Li, B.: Spear: Optimized dependency-aware task scheduling with deep reinforcement learning. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), p. 201. IEEE, Dallas, TX, USA (2019). doi:10.1109/ICDCS.2019.00201
4. Liu, H., Zhou, H., Chen, H., Yan, Y., Huang, J., Xiong, A., Yang, S., Chen, J., Guo, S.: A federated learning multi-task scheduling mechanism based on trusted computing sandbox. *Sensors* **23**(4), 2093 (2023). doi:10.3390/s23042093. URL <https://doi.org/10.3390/s23042093>
5. Stephens, D., Bennett, J., Zhang, H.: Implementing scheduling algorithms in high-speed networks. *IEEE Journal on Selected Areas in Communications* **17**(6), 1145–1158 (1999). doi:10.1109/49.772449
6. Васильев А.Н., Маковская А.Е., Каплан А.Я.: Динамика сенсомоторного ритма ЭЭГ при мысленном повторении за наблюдаемым движением. *Журнал высшей нервной деятельности им. И. П. Павлова* **73**(4) (2023)
7. Комаров А.С., Чертов О.Г., Быховец С.С.: Воздействие осинового насаждения с коротким оборотом рубки на биологический круговорот углерода и азота в лесах бореальной зоны: модельный эксперимент. *Математическая биология и биоинформатика* **10**(2), 398–415 (2015)
8. Санников Т.В., Сальников А.Н.: Двухуровневое планирование потока задач между несколькими вычислительными кластерами. In: Сборник РИНЦ «Программные Системы и Инструменты», pp. 60–73 (2023)

О параллелизации метода Монте-Карло при решении некоторых метрологических задач

А.В. Степанов

ВНИИМ им. Д.И. Менделеева

Рассмотрены некоторые типовые метрологические задачи, для решения которых применим метод Монте-Карло. Показано, что его реализация зачастую допускает простую параллелизацию стандартными средствами используемого языка программирования (в статье – Java). Среди прочих, упомянуты такие задачи, как трансформирование распределений, получение оценок максимального правдоподобия, метрологическая аттестация алгоритмов обработки данных, получение критических значений статистических критериев.

Ключевые слова: метод Монте-Карло, параллелизм, многопоточность, неопределенность измерения, случайная величина, оценка параметров распределений, интервал охвата, коэффициент охвата.

1. Введение

Метод Монте-Карло, представляющий собой метод статистического моделирования, основанный на многократной генерации наборов случайных данных и последующем анализе результатов их обработки, широко используется при решении метрологических задач [1–4]. В качестве одного из применений можно упомянуть трансформирование распределений с целью оценивания неопределенности измерения, описываемого заданной математической моделью, рекомендованный к применению Приложением 1 к Руководству по выражению неопределенностей [5, 6]. Данный метод, в частности, особенно актуален, когда распределения входных величин обладают заметной асимметрией, или (и) модель измерения достаточно сложна (например, существенно нелинейна). Также можно упомянуть такие его преимущества, как отсутствие необходимости делать предположения о виде закона выходного распределения (при классическом подходе оно предполагается нормальным или t -распределением); возможность оценки разнообразных статистических характеристик распределения выходной величины (а не только среднеквадратического отклонения), и т.д. Другими примерами использования метода Монте-Карло являются получение критических значений при выводе критериев проверки статистических гипотез, а также оценка и сравнение мощностей и свойств статистических критериев при конечных объемах выборок [7]; метрологическая аттестация (т.е. исследование точностных свойств в рамках конкретной измерительной задачи) и сравнение характеристик алгоритмов обработки измерительных данных.

Развитие вычислительной техники и повышение ее доступности и производительности, широкое распространение программных библиотек обработки данных (разнообразных математических утилит, генераторов случайных чисел и т.д.), наличие подробной документации, специализированных статей, значительно увеличили доступность численного моделирования для исследователей. Становится возможным смещать фокус с классических алгоритмов обработки данных, постулирующих нормальность рассматриваемых распределений, теория применения которых хорошо проработана, на иные, иногда более релевантные, модели: например, рассматривать асимметричные распределения, учитывать естественные ограничения на множества допустимых значений, и т.д. При этом достаточно давно наблюдается тренд на параллелизацию проводимых численных экспериментов, связанный с распространением многоядерных и распределенных вычислительных систем [8, 9]. Отдельно следует упомянуть технологии проведения поточно-параллельных вычислений на графических процессорах (GPU) [10], таких, как CUDA или ROCm (их использование предъявляет спе-

цифические требования как к аппаратному обеспечению, так и к подготовке исследователя, и в данной статье не они не рассматриваются).

Использование метода Монте-Карло допускает простую и естественную параллелизацию для решения многих вышеупомянутых задач. Ниже будут приведены примеры такой параллелизации с использованием механизмов поддержки многопоточности, когда каждый вычислительный эксперимент метода производится в отдельных потоках исполнения кода, после чего совместные результаты подвергаются постобработке. Указанный подход позволяет значительно ускорить проводящиеся вычисления при использовании многоядерных и многопроцессорных систем. Следующий пункт содержит описание данного подхода, с использованием конкретного языка программирования (Java); далее приведены примеры его применения для исследования свойств одного класса распределений. В конце статьи для некоторых примеров будут приведены цифры, демонстрирующие ускорение вычислений при использовании многопоточности.

2. Общий принцип организации кода

Схема моделирования, в общем случае, представляет собой многократно (порядка 10^6 раз и выше, в зависимости от требуемой точности результатов) повторяемые численные эксперименты (прогоны, trials) обработки сгенерированных данных, накопление и совместную постобработку результатов. Для многих рассматривавшихся приложений цикл обработки моделируемых данных, является относительно однородным, т.е. отдельные эксперименты имеют сходную вычислительную сложность.

Как уже было сказано выше, в данной работе предлагается рассматривать наиболее доступный подход, ориентированный на применение доступных многоядерных (многопроцессорных) вычислительных систем. При решении задач использовался язык программирования Java (версии 8 и выше), тем не менее, данный выбор не принципиален (автор не ставил перед собой задачи сравнения быстродействия различных языков программирования и платформ). Дополнительным преимуществом, помимо личных предпочтений автора, был принцип WORA – «Write Once, Run Anywhere», так как реализованный функционал использовался на различных архитектурах (x86-64, Arm64), на машинах, имеющих от нескольких до нескольких десятков вычислительных ядер. Параллелизация достигалась путем использования механизмов многопоточности, поддерживаемых языком (распределенные вычисления рассматривались исключительно в виде эксперимента). Даже такой простейший подход давал заметный прирост производительности, используемый функционал показал достаточно хорошую масштабируемость при использовании на многоядерных платформах.

В простейшем случае структура приложения может быть описана следующим образом (рис. 1). Заметим, что использование именно Java Parallel Stream API, демонстрируемое в приведенном листинге, не является обязательным (можно выбрать множество способов реализации, основанных на использовании классов пакета `java.util.concurrent`), но код в данном виде выглядит особенно лаконично.

В качестве генератора случайных чисел предлагается использовать класс `ThreadLocalRandom`, который обеспечивает корректную генерацию псевдослучайных чисел при использовании многопоточности.

Безусловно, приведенный код допускает различные модификации. Например, в случае, если выходные данные алгоритма моделирования одномерны (рис. 2), то отпадает необходимость в полях для хранения промежуточных результатов (вообще говоря, от них можно избавиться и в случае многомерных выходных данных, это незначимый технический вопрос организации кода).

```
public class MCExample {

    private final MCPParams params;
    private final int nTrials;
    private final double[] out_1;
    private final double[] out_2;
    ...

    public MCExample(MCPParams params, int nTrials) {
        this.params = params; // use to generate input data
        this.nTrials = nTrials;
        out_1 = new double[nTrials];
        out_2 = new double[nTrials];
        ...
    }

    private double[] generateInputData() { /*...*/ }

    private double dataProcessing1(double[] data) { /*...*/ }
    private double dataProcessing2(double[] data) { /*...*/ }

    private void trial(int i) {
        double data[] = generateInputData();
        out_1[i] = dataProcessing1(data);
        out_2[i] = dataProcessing2(data);
        ...
    }

    private void runTrialsInParallel() {
        IntStream.range(0, nTrials).parallel().forEach(this::trial);
    }

    private void postProcessing() {
        // post-processing of out_1, out_2, ...
    }

    public static void main(String[] args) {
        MCPParams params = new MCPParams(...);
        MCExample mc = new MCExample(params, 5_000_000);
        mc.runTrialsInParallel();
        mc.postProcessing();
    }
}
```

Рис. 1. Общая схема параллелизации метода Монте-Карло с использованием Parallel Stream API

```
...
private double trial(int i) { /*...*/ }

private double[] runTrialsInParallel() {
    return IntStream.range(0, nTrials).parallel().
        mapToDouble(this::trial).toArray();
}

private void postProcessing(double results[]) { /*...*/ }

public static void main(String[] args) {
    MCPParams params = new MCPParams(...);
    MCExample mc = new MCExample(params, 5_000_000);
    double[] results = mc.runTrialsInParallel();
    mc.postProcessing(results);
}
...
```

Рис. 2. Модификация общей схемы при одномерных выходных данных

Заметим, что пользователь имеет возможность управлять числом потоков, выделяемых для параллельной обработки данных, например, используя опцию `-Djava.util.concurrent.ForkJoinPool.common.parallelism` при запуске программы.

3. Примеры использования

Ниже будут приведены примеры применения предлагаемого подхода. Численные эксперименты, выполнявшиеся при написании работ [7, 11–14] (и некоторых других), проводились с его использованием. Заметим, что приведенные ниже примеры не претендуют на полноту описания возможных применений, разнообразие которых крайне велико, а лишь демонстрируют общий для подобного рода задач подход.

3.1. TSP-распределение

В то время как сложившиеся метрологические практики, как правило, предполагают нормальность (иногда – равномерность) рассматриваемых законов распределения, широкое распространение программного обеспечения открывает возможности использования иных семейств распределений для моделирования погрешностей измерений и нечеткой информации. В качестве модельного в дальнейших примерах будет рассмотрено семейство двусторонних степенных (Two-Sided Power, TSP) распределений [15, 16]. Плотность данного распределения является четырехпараметрической функцией, определенной следующим образом:

$$f(x) = f(x|a, m, b, p) = \begin{cases} \frac{p}{b-a} \left(\frac{x-a}{m-a} \right)^{p-1}, & a < x \leq m, \\ \frac{p}{b-a} \left(\frac{b-x}{b-m} \right)^{p-1}, & m \leq x < b, \\ 0, & \text{иначе.} \end{cases} \quad (1)$$

Носителем распределения является интервал (a, b) . Параметр p положителен, при $p = 1, 2$ имеем равномерное и треугольное распределение, соответственно. При $p < 1$ график f имеет U-образную форму (распределение является двухмодальным), при $p > 1$ параметр m представляет собой моду распределения. Факт принадлежности случайной величины x данному распределению будем обозначать следующим образом: $x \in TSP(a, m, b, p)$.

Семейство TSP-распределений, несмотря на крайнюю простоту математического описания, достаточно разнообразно. В работе [16] рассмотрена диаграмма моментов данного распределения, и показано, что разнообразие распределений семейства сопоставимо со случаем Бета-распределением. Также, опираясь на различные критерии, можно аппроксимировать им то или иное непрерывное распределение на заданном отрезке.

Заметим, что функция распределения для (1) и обратная к ней имеют простое математическое выражение [16]:

$$F(y) = \begin{cases} \frac{m-a}{b-a} \left(\frac{x-a}{m-a} \right)^p, & a \leq x \leq m, \\ 1 - \frac{b-m}{b-a} \left(\frac{b-x}{b-m} \right)^p, & m \leq x \leq b, \end{cases}$$

$$F^{-1}(y) = \begin{cases} a + \sqrt[p]{y(m-a)^{p-1}(b-a)}, & 0 \leq y \leq \frac{m-a}{b-a}, \\ b - \sqrt[p]{(1-y)(b-m)^{p-1}(b-a)}, & \frac{m-a}{b-a} \leq y \leq 1, \end{cases}$$

что позволяет легко реализовать генератор случайных чисел на основе метода обратного преобразования (рис. 3).

```

...
private final double q = (m - a) / (b - a);
...
private double[] generateInputData() {
    double res[] = new double[sampleLength];
    for (int i = 0; i < sampleLength; ++i) {
        double u = ThreadLocalRandom.current().nextDouble();
        if (u < q) {
            res[i] = a + Math.pow(u *
                Math.pow(m - a, p - 1) * (b - a), 1 / p);
        } else {
            res[i] = b - Math.pow((1 - u) *
                Math.pow(b - m, p - 1) * (b - a), 1 / p);
        }
    }
    return res;
}
...

```

Рис. 3. Генератор выборок из TPS-распределения

Заметим также, что крайняя простота описания (1) позволяет, например, аналитически выразить границы интервалов охвата (которые в метрологической практике, как правило, служат мерой расширенной неопределенности [17]). Напомним, что отрезок (можно также рассматривать открытый интервал) $I_{P_0} = [c_1, c_2]$ называют интервалом охвата для некоторого непрерывного распределения, отвечающего вероятности (уровню) охвата P_0 , если $P\{c_1 \leq x \leq c_2\} = P_0$, или, что то же самое, $F(c_2) - F(c_1) = P_0$, где F – функция распределения. Очевидно, что данное определение не является однозначным, рассматриваемое распределение для заданного уровня P_0 имеет множество различных интервалов охвата, включая кратчайший из них. Если рассматриваемый интервал охвата является двусторонним и вероятностно-симметричным, т.е.

$$F(c_1) = 1 - F(c_2) = \frac{1 - P_0}{2},$$

то из выражения для $F(x)$ следует, что

$$c_1 = a + (m - a) \sqrt[p]{\frac{1}{2} \frac{b - a}{m - a} (1 - P_0)}, \quad c_2 = b - (b - m) \sqrt[p]{\frac{1}{2} \frac{b - a}{b - m} (1 - P_0)} \quad (2)$$

(считаем, что $m \in [c_1, c_2]$). В случае симметричного распределения, положив $m = 0$, получим: $c_1 = -c_2$, и можно также вывести выражение для коэффициента охвата

$$K = \frac{c_2}{\sqrt{\text{Var}(x)}} = \left(1 - \sqrt[p]{1 - P_0}\right) \sqrt{\frac{1}{2}(p+1)(p+2)}.$$

Построение кратчайшего интервала охвата для TSP-распределения также является несложной вычислительной задачей.

С точки зрения метрологических приложений, иногда более естественным может выглядеть случай, когда пользователь вместо параметров m, p задает среднее значение случайной величины x_0 (которое ассоциируется с измеренным значением) и среднеквадратическое отклонение (СКО) u (которое ассоциируется с неопределенностью измерения), считая при этом диапазон значений известным. В этом случае параметры m, p могут быть выражены через x_0 (для симметричного распределения совпадает с m), u .

3.2. Пример 1. Трансформирование распределений

Рассмотрим частный случай задачи трансформирования распределений [5,6], когда все входные величины принадлежат семейству (1), симметричны, а выходная величина явля-

ется их линейной комбинацией:

$$y = \sum_{i=1}^n \alpha_i x_i, \quad x_i \in TSP(m_i - r_i, m_i, m_i + r_i, p_i).$$

Требуется выяснить, можно ли также аппроксимировать распределение выходной величины y симметричным TSP-распределением, с приемлемым качеством (в общем случае, очевидно, распределение y не принадлежит семейству TSP). В качестве критерия «приемлемости», с метрологической точки зрения, можно выбрать следующий: вероятность охвата $\hat{P}_0$, вычисленная для интервала охвата TSP-аппроксимации y , отвечающего заданному уровню вероятности P_0 , не будет значимо (более чем на 1–2%) отличаться от P_0 в меньшую сторону для фактического распределения y (отличие в большую сторону не так критично, т.к. оно означает переоценку интервала охвата, а не его недооценку).

Легко проверить, что для симметричного распределения $TSP(m - r, m, m + r, p)$ дисперсия величины x имеет вид:

$$u^2 = Var(x) = \frac{2r^2}{(p+1)(p+2)}$$

(здесь u – обозначение для неопределенности, отождествляемой с СКО величины x). И обратно, рассматривая последнее равенство, как уравнение относительно p , можем выразить p через u (или, что то же самое, через $Var(x)$):

$$p = \frac{1}{2} \left(\sqrt{1 + \frac{8r^2}{u^2}} - 3 \right) = \frac{1}{2} \left(\sqrt{1 + \frac{8r^2}{Var(x)}} - 3 \right)$$

(отбросили отрицательный корень квадратного уравнения, т.к. $p > 0$). Математическое ожидание и дисперсия величины y имеют вид:

$$m_y = E(y) = \sum_{i=1}^n \alpha_i m_i, \quad u_y^2 = Var(y) = \sum_{i=1}^n \alpha_i^2 u_i^2, \quad u_i^2 = \frac{2r_i^2}{(p_i+1)(p_i+2)}.$$

Далее, очевидно, $|y - m_y| < r_y = \sum_{i=1}^n |\alpha_i| r_i$. Таким образом, предполагая принадлежность величины y семейству TSP ($y \in TSP(m_y - r_y, m_y, m_y + r_y, p_y)$), для величины p_y получим оценку:

$$\hat{p}_y = \frac{1}{2} \left(\sqrt{1 + \frac{8r_y^2}{u_y^2}} - 3 \right), \quad \implies \quad \hat{K}_y = \left(1 - \sqrt[3]{1 - P_0} \right) \sqrt{\frac{1}{2}(\hat{p}_y + 1)(\hat{p}_y + 2)},$$

и оценка интервала охвата величины y , полученная в предположении о приемлемости аппроксимации величины y TSP-распределением, а также искомое значение $\hat{P}_0$, имеют вид:

$$\hat{I}_{P_0} = [m_y - \hat{K}_y u_y, m_y + \hat{K}_y u_y], \quad \hat{P}_0 = P \left\{ y \in \hat{I}_{P_0} \right\} = P \left\{ |y - m_y| \leq \hat{K}_y u_y \right\}$$

(подчеркнем, что вероятность $\hat{P}_0$ является эмпирической, получаемой в результате моделирования).

Рассмотрим конкретный пример. Пусть $m_0 = m_1 = 0$, $r_1 = r_2 = 1$, $p_1 = 0.8$, $p_2 = 2.7$, $\alpha_1 = 0.3$, $\alpha_2 = 0.7$, тогда $m_y = 0$, $r_y = 1$, $p_y \approx 3.19$, и $K_y \approx 2.01$. Проведенное моделирование показывает, что $\hat{P}_0 \approx 0.957 > P_0$, аппроксимацию можно считать приемлемой. Аналогично, изменим значение параметра p_2 , все остальные оставив прежними: пусть $p_2 = 1$. Тогда $p_y \approx 1.71$, $K_y \approx 1.85$, и $\hat{P}_0 \approx 0.956 > P_0$, построенная TSP-аппроксимация снова является приемлемой.

С точки зрения программной реализации, получение трансформированного распределения величины y является задачей, подлежащей параллелизации; метод (3) генерации входных данных при этом придется очевидным образом доработать, чтобы различные значения выборки принадлежали разным TSP-распределениям.

Заметим, что решение данной задачи в случае, когда распределения входных величин x_i асимметричны, решение данной задачи представляет еще больший интерес. Тем не менее, опустим этот вопрос здесь, во избежание громоздких математических выкладок.

3.3. Пример 2. Получение оценок максимального правдоподобия измеряемого значения при многократных неравноточных измерениях

Обработка неравноточных измерительных данных является одной из распространенных метрологических задач. Предположим, что имеется n результатов измерений $x_i, i = \overline{1, n}$ некоторой величины, каждому из которых приписано свое распределение $TSP(a_i, m_i, b_i, p_i)$. Предположим, что неизвестное измеряемое значение x_0 ассоциируется с математическим ожиданием для каждого распределения, и известны максимальные отклонения $r_{1,i}, r_{2,i}$ от x_0 , т.е. $a_i = x_0 - r_{1,i}, b_i = x_0 + r_{2,i}$ ($r_{1,i}, r_{2,i} > 0$). Выражение для m_i в этом случае имеет вид:

$$m_i = x_0 + \frac{r_{1,i} - r_{2,i}}{p_i - 1}, \quad p_i \neq 1$$

(положим $m_i = x_0 + \frac{r_{2,i} - r_{1,i}}{2}$ при $p_i = 1$). В итоге, $x_i \in TSP(x_0 - r_{1,i}, x_0 + \frac{r_{1,i} - r_{2,i}}{p_i - 1}, x_0 + r_{2,i}, p_i)$ (в случае, когда измеряемое значение ассоциируется с наиболее часто встречаемым значением, т.е. модой распределения, считаем, что $x_i \in TSP(x_0 - r_{1,i}, x_0, x_0 + r_{2,i}, p_i)$).

Для получения оценки $\hat{x}_0$ измеряемого значения x_0 предлагается использовать метод максимального правдоподобия [13], т.е.

$$\hat{x}_0 = \arg \max_{x_0} L(x_0), \quad L(x_0) = \prod_{i=1}^n f \left(x_i | x_0 - r_{1,i}, x_0 + \frac{r_{1,i} - r_{2,i}}{p_i - 1}, x_0 + r_{2,i}, p_i \right),$$

здесь $L(x_0)$ – функция правдоподобия, а f задается формулой (1). Получение оценок $\hat{x}_0$ является задачей, подлежащей параллелизации; метод (3) снова придется очевидным образом изменить, чтобы получать выборку из разных распределений. Итоговая выборка $\{\hat{x}_{0,j}\}_1^N$ (**results**, в обозначениях листинга (2)) подвергается метрологической пост-обработке: в качестве окончательной оценки значения измеряемой величины x_0 вычисляется среднее значение выборки, в качестве мер неопределенности и расширенной неопределенности x_0 вычисляются выборочное СКО и интервал охвата для заданного уровня вероятности. Полученные оценки могут отличаться от таковых для стандартных процедур, предполагающих нормальность распределений (для достаточно больших n они, как правило, близки).

3.4. Пример 3. Метрологическая аттестация алгоритма получения оценок максимального правдоподобия параметров TSP-распределения

Применение метода Монте-Карло позволяет оценить смещения оценок максимального правдоподобия для параметров TSP-распределения, получаемых по выборкам конечной длины. Обозначим $\{x_i\}_1^n$ выборку длиной n из этого распределения, а $x_{(1)} \leq \dots \leq x_{(n)}$ – соответствующий вариационный ряд. В этом данные оценки имеют вид [15, 16]:

$$\hat{m} = x_{(i_0)}, \quad \hat{p} = -\frac{n}{\log M(i_0)}, \quad i_0 = \arg \max_i M(i), \quad M(i) = \prod_{j=1}^{i-1} \frac{x_{(j)} - a}{x_{(i)} - a} \prod_{j=i+1}^n \frac{b - x_{(j)}}{b - x_{(i)}}.$$

Таким образом, в обозначениях кода (1), массивы **out_1**, **out_2** предназначены для сбора оценок $\hat{m}, \hat{p}$ (а получение этих оценок подлежит параллелизации). Пост-обработка включает

в себя вычисление смещения оценок от истинных значений, заданных при моделировании, и соответствующих СКО.

Предположим, что $p > 1$. Моделирование показывает, что в случае асимметричного TPS-распределения оценки параметров m , p являются смещенными. Например, при $a = 0$, $b = 1$ (случай так называемого стандартного TPS-распределения, STSP) и $m > 0.5$ математическое ожидание $E\hat{m}$ смещено, по сравнению со значением m , использовавшимся при моделировании, влево (в сторону нуля) тем сильнее, чем ближе исходный параметр m к единице; величина $E\hat{p}$ смещена относительно p вверх. В табл. 1 ниже приведены относительные смещения $\delta\hat{m}$, $\delta\hat{p}$ (в процентах) оценок параметров m , p , для выборок длиной $n = 20, 50$. Применение смещенных оценок максимального правдоподобия $\hat{m}$, $\hat{p}$ для оценивания интервала охвата по формулам (2) приводят к его недооценке при малых n , особенно в случае значений p , близких к единице, что влечет необходимость коррекции полученных оценок $\hat{m}$, $\hat{p}$. В то же время, при длинах выборки более 50 (и особенно ближе к 100) недооценка интервала охвата не является практически значимой (пренебрежимо мала).

Таблица 1. $\delta\hat{m}$, $\delta\hat{p}$, %

n	20						50					
m	0.75		0.90		0.95		0.75		0.90		0.95	
p	$\delta\hat{m}$	$\delta\hat{p}$	$\delta\hat{m}$	$\delta\hat{p}$	$\delta\hat{m}$	$\delta\hat{p}$	$\delta\hat{m}$	$\delta\hat{p}$	$\delta\hat{m}$	$\delta\hat{p}$	$\delta\hat{m}$	$\delta\hat{p}$
1.5	-7.4	15.8	-10.0	15.3	-10.6	14.9	-2.6	5.9	-3.9	5.9	-4.4	5.8
2.0	-2.0	12.3	-3.2	12.6	-3.7	12.5	-0.7	4.5	-1.1	4.6	-1.3	4.7
2.5	-1.0	11.0	-1.6	11.4	-2.0	11.6	-0.4	4.0	-0.6	4.2	-0.7	4.3
3.0	-0.7	10.4	-1.0	10.8	-1.3	11.1	-0.3	3.8	-0.4	3.9	-0.4	4.0
3.5	-0.5	10.0	-0.7	10.4	-0.9	10.8	-0.2	3.7	-0.3	3.8	-0.3	3.9
4.0	-0.4	9.7	-0.6	10.1	-0.7	10.5	-0.2	3.6	-0.2	3.7	-0.2	3.8
4.5	-0.3	9.5	-0.5	9.9	-0.6	10.3	-0.1	3.5	-0.2	3.6	-0.2	3.7
5.0	-0.3	9.4	-0.4	9.8	-0.5	10.2	-0.1	3.5	-0.2	3.6	-0.2	3.7

3.5. Пример 4. Подбор наиболее подходящего распределения из семейства по выборочным данным

Метод выбора распределения из конечного набора непрерывных распределений, наилучшим образом согласующегося с имеющимися экспериментальными данными, рассмотрен в работе [18]. Применительно к семейству (1), он выглядит следующим образом: для имеющейся выборки длины n экспериментальных данных $\{x_i\}_1^n$ из конечного множества $S = \{(a, b, m, p)\}$ наборов параметров распределений данного семейства выбирается набор $\hat{s} \in S$, доставляющий минимум функционалу следующего вида:

$$\hat{s} = (\hat{a}, \hat{m}, \hat{b}, \hat{p}) = \arg \min_{s \in S} \sum_{i=1}^n \left(F_s^{-1} \left(\frac{i}{n} \right) - x_{(i)} \right)^2.$$

Здесь $\{x_{(i)}\}_1^n$ – вариационный ряд для рассматриваемой выборки, а F_s – функция распределения, отвечающая конкретному набору параметров s .

В качестве критерия метрологической аттестации, в частности, можно рассмотреть вероятность выбора верного распределения из множества предложенных вариантов. Код, ответственный за выбор распределения, наилучшим образом согласующегося со сгенерированной выборкой, при этом является телом метода `trial` (в обозначениях рис. 1), а эмпирические вероятности выбора того или иного распределения основаны на счетчиках выбора предложенных альтернатив (т.е. массив `out` содержит столько счетчиков, сколько рассматривается вариантов выбора). Пусть, например, $n = 200$, $a = -1$, $b = 1$,

$S = \{m = -0.1, 0, 0.1\} \times \{p = 1.5, 2, 2.5\}$, и входные (моделируемые) данные принадлежат распределению $TSP(-1, 0, 1, 2)$ (т.е. $s_0 = (0, 2)$). Вероятности выбора наборов $s \in S$ приведены в табл. 2.

Таблица 2. Значения $P(s)$

$s = (m, p)$	$P(s)$
$(-0.1, 1.5)$	0.001
$(-0.1, 2.0)$	0.053
$(-0.1, 2.5)$	0.005
$(0.0, 1.5)$	0.017
$(0.0, 2.0)$	0.823
$(0.0, 2.5)$	0.072
$(0.1, 1.5)$	0.001
$(0.1, 2.0)$	0.026
$(0.1, 2.5)$	0.002

Естественно, что вероятность $P\{s = s_0\}$ (выделена жирным шрифтом) существенным образом зависит от длины выборки n (увеличивается с ростом n). Для $n = 250, 300$, например, она составляет 0.882 и 0.920, соответственно. Заметим также, что преимуществом рассмотренного метода является возможность расширять множество альтернатив распределениями из других семейств (например, проводившиеся численные эксперименты показали, что распределения семейства TSP достаточно уверенно отделяются от усеченных нормальных распределений, при достаточной длине выборки). Также можно рассматривать его в качестве критерия подбора аппроксимирующего распределения из семейства (1) для произвольного непрерывного распределения.

В работе [11] задача подбора TSP-распределения по выборочным данным рассмотрена более подробно (в частности, показано, что при подборе единственного параметра степени p для симметричного распределения данный метод дает схожие оценки, по сравнению с методом максимального правдоподобия).

3.6. Пример 5. Оценивание интервалов охвата по выборочным размаху и СКО

Рассмотрим вопрос об оценивании интервалов охвата симметричного TSP-распределения по выборочным данным. В качестве оценки интервала охвата, отвечающего уровню вероятности P_0 , рассмотрим отрезки вида $I = [c_1, c_2] = [\bar{x} - C_d d, \bar{x} + C_d d]$, где $\bar{x}$ – выборочное среднее, d – выборочная статистика, характеризующая меру разброса данных, например, СКО S или размах R . Воспользуемся методом Монте-Карло для выбора коэффициента C_d , исходя из условия $P\{F(c_2) - F(c_1) \geq P_0\} = P_1$, т.е. предполагается, что рассматриваемый отрезок содержит с заданной вероятностью P_1 некоторый интервал охвата для данного уровня вероятности P_0 . Вычисление значений $F(c_2) - F(c_1)$ для пробных значений константы C_d подлежит параллелизации. Ниже в табл. 3, 4 в качестве примера приведены некоторые значения оценок коэффициентов C_R, C_S при $P_0 = 0.95$ и $P_1 = 0.95, 0.99$ (заметим, что величина C_d возрастает с ростом P_1 , а также убывает с ростом длины выборки n , для каждого рассматриваемого распределения).

В рассмотренных примерах обе оценки обеспечивают, в среднем, сопоставимую длину отрезка $[c_1, c_2]$.

Таблица 3. Значения C_R, C_S ($P_0 = 0.95, P_1 = 0.95$)

	n	p						
		0.75	1	1.5	2	3	5	7
C_r	5	1.82	1.84	1.90	1.99	2.14	2.34	2.44
	7	1.20	1.25	1.33	1.39	1.49	1.63	1.70
	10	0.90	0.95	1.01	1.05	1.13	1.22	1.26
	15	0.74	0.76	0.80	0.83	0.88	0.94	0.97
C_s	7	3.23	3.40	3.63	3.82	4.14	4.51	4.71
	10	2.65	2.82	3.04	3.21	3.48	3.77	3.94
	15	2.32	2.45	2.65	2.80	3.02	3.27	3.40
	20	2.16	2.28	2.46	2.61	2.81	3.02	3.13

Таблица 4. Значения C_R, C_S ($P_0 = 0.95, P_1 = 0.99$)

	n	p						
		0.75	1	1.5	2	3	5	7
C_r	5	3.23	3.00	2.96	3.07	3.33	3.68	3.87
	7	1.80	1.78	1.82	1.91	2.07	2.29	2.40
	10	1.19	1.22	1.27	1.34	1.45	1.59	1.66
	15	0.88	0.91	0.95	0.99	1.07	1.16	1.20
C_s	7	4.88	4.87	5.00	5.23	5.70	6.31	6.65
	10	3.50	3.64	3.85	4.06	4.43	4.88	5.13
	15	2.78	2.93	3.15	3.33	3.63	3.97	4.16
	20	2.50	2.63	2.83	2.99	3.26	3.55	3.71

3.7. Пример 6. Получение критических значений для статистических критериев

Примеры применения метода Монте-Карло в данном случае крайне разнообразны. Можно еще раз упомянуть получение критических значений для критериев однородности дисперсий выборок из TSP-распределений [7].

Здесь же рассмотрим простейший критерий согласованности данных. Предположим, что у исследователя имеется массив накопленных данных из симметричного TPS-распределения (выборка длины n). Предположим, что имеется новый массив данных из того же самого распределения длины $k < n$. Рассмотрим статистику $C_{k,n} = \frac{R_k}{S_n}$, где R_k – размах второй выборки, а S_n – выборочное СКО первой. Соответствующие критические значения для некоторых значений параметров p, k, n (параметр масштаба не имеет значения) и уровня вероятности 0.95 приведены в табл. 5.

Таблица 5. Критические значения $C_{k,n}$

n	k	p				
		1.0	1.5	2	3	5
50	7	3.39	3.75	3.99	4.29	4.60
	10	3.49	3.91	4.20	4.59	4.99
100	7	3.33	3.68	3.91	4.20	4.49
	10	3.42	3.82	4.11	4.48	4.85

В качестве другого тривиального примера можно рассмотреть использование выборочного коэффициента асимметрии $G_1 = m_3 \cdot m_2^{-3/2}$, где m_i – i -й центральный выборочный момент, для проверки гипотезы о симметричности распределения (ожидаем, что его абсолютное значение будет ниже некоторого порога, с вероятностью P_0). Например, критические значения величины $|G_1|$ при $n = 100$ и $P_0 = 0.95$ составляют 0.303, 0.385, 0.549 для $p = 2, 3, 5$, соответственно.

В обоих приведенных выше примерах вычисления статистик параллелизовались (при этом в первом примере требовалось генерация двух выборок различных длин на эксперимент).

3.8. Ускорение при использовании многопоточности

Репозиторий [19] содержит несколько примеров кода, соответствующих описанным задачам. Все они носят демонстрационный характер, на практике вычисления, как правило, более объемны. В таблице ниже приведены данные, позволяющие оценить ускорение при использовании многопоточности, по сравнению с последовательным (для оценивания последнего комментировался метод `parallel`). Вычисления проводились на ноутбуке Huawei MateBook D16, с процессором Intel Core i5-12450H (8 ядер / 12 потоков, Hyper-Threading), под управлением OS Windows 11.

Таблица 6. Ускорение, полученное для некоторых примеров

Пример	Время выполнения кода, сек.		Ускорение
	последовательное	многопоточное	
2	1376.11	264.13	5.21
3	486.04	45.07	10.78
4	1296.71	154.65	8.38
6	85.80	8.38	10.24

Заметим, что ускорение для примеров 3, 6 близко к идеальному, т.е. эти задачи очень хорошо масштабируемы. Пример 4 также демонстрирует очень хороший результат, пример 2 – достаточно средний. Естественно, что при использовании большего числа ядер можно добиться более впечатляющих результатов (к сожалению, конкретные значения коэффициента ускорения, полученные при работе на более производительных многоядерных системах, не были сохранены автором).

4. Заключение

Современные версии популярных языков программирования (C++, Java, Go, Python, ...) содержат мощную и, в то же время, синтаксически лаконичную поддержку механизмов параллелизации исполнения кода (в частности, поддержку многопоточности), использование которой зачастую не требует экспертных навыков работы с таким кодом от разработчиков прикладного (научного) программного обеспечения. Данный инструментарий хорошо документирован, снабжен многочисленными примерами использования и, в случае необходимости, легко может быть освоен в сжатые сроки, т.е. порог вхождения при его использовании на описанном (относительно тривиальном) уровне перестает быть традиционно высоким. С другой стороны, широкое распространение получают персональные компьютеры с достаточно большим количеством ядер (на данный момент – 12, 14, 16 и выше); в случае необходимости, не слишком сложно получить доступ к машинам и с большим количеством (48, 64, ...) вычислительных ядер. Существует достаточно обширный класс алгоритмически не слишком сложных и вычислительно однородных (хорошо масштабируемых) задач, для которых сочетание указанных факторов позволяет значительно (в разы, иногда – на порядок и более, в зависимости от используемого аппаратного обеспечения) ускорить процесс математического моделирования и обработки результатов, даже не прибегая к использованию специализированной вычислительной техники и соответствующих библиотек, что повышает скорость и доступность проведения вычислительных экспериментов, заметно ускоряет поиск ошибок и валидацию полученных результатов, а также позволяет добиться повышения точности результатов за счет увеличения числа испытаний метода Монте-Карло. Безусловно, использование распределенных вычислений позволило бы добиться дальнейшего ускорения экспериментов, но оно уже накладывает на исследователя и его работодателя некоторые издержки как по организации и поддержке соответствующей вычислительной инфраструктуры, так и финансовые расходы на ее приобретение (или аренду) и обслуживание. Использование же встроенных механизмов поддержки многопоточности позволяет быстро получить заметный эффект уже на этапе прототипирования или выполнения исследовательских задач в инициативном порядке (например, при подготовке научных публикаций) в случае, когда у работника отсутствуют как доступ к высокопроизводительным вычислительным мощностям в своей организации, так и достаточно сложные задачи, для решения которых требуются такие мощности (далеко не каждый численный эксперимент требует наличия суперкомпьютера, многие из рассмотренных задач вполне могут быть решены на достаточно производительном персональном компьютере с многоядерным процессором в сжатые сроки). При этом, безусловно, стоит подчеркнуть, что существует множество вычислительно сложных исследовательских задач, когда описанного подхода совершенно недостаточно, и требуется использование совсем иной вычислительной техники (суперкомпьютеров и вычислительных кластеров).

Литература

1. Harris P., Cox M. On a Monte Carlo method for measurement uncertainty evaluation and its implementation // Metrologia. 2014. Vol. 51, no. 4. P. S176–S182. DOI: 10.1088/0026-1394/51/4/S176.
2. van der Veen A., Cox M. Getting started with uncertainty evaluation using the Monte Carlo method in R // Accreditation and Quality Assurance. 2021. Vol. 26. P. 129–141. DOI: 10.1007/s00769-021-01469-5.
3. Crowder S., Delker C., Forrest E., Martin N. Monte Carlo Methods for the Propagation of Uncertainties // Introduction to Statistics in Metrology. Springer, 2020. P. 153–180. DOI: 10.1007/978-3-030-53329-8_8.

4. Zhang J. Modern Monte Carlo methods for efficient uncertainty quantification and propagation: A survey // WIREs Computational Statistics. 2020. Vol. 15, no. 5. DOI: 10.1002/wics.1539.
5. Evaluation of measurement data – Supplement 1 to the «Guide to the expression of uncertainty in measurement» – Propagation of distributions using a Monte Carlo method / Joint Committee for Guides in Metrology, JCGM 101:2008.
6. ГОСТ 34100.3.1 Неопределенность измерения. Часть 3. Руководство по выражению неопределенности измерения. Дополнение 1: Трансформирование распределений с использованием метода Монте-Карло. Стандартиформ, 2018.
7. Stepanov A.V., Chunovkina A.G. On testing of the homogeneity of variances for two-sided power distribution family // Accreditation and Quality Assurance. 2023. Vol. 28. P. 129–137. DOI: 10.1007/s00769-022-01525-8.
8. J. Rosenthal. Parallel computing and Monte Carlo algorithms // Far East Journal of Theoretical Statistics. 2000. Vol. 4, no. 2. P. 207–236.
9. Maigne L., Hill D., Calvat P. et al. Parallelization of monte Carlo simulations and submission to a grid environment // Parallel Processing Letters. 2004. Vol. 14, no. 2. P. 177–196. DOI: 10.1142/S0129626404001829.
10. Некрасов К.А., Поташников С.И., Боярченков А.С., Купряжкин А.Я. Метод Монте-Карло на графических процессорах: учебное пособие. Екатеринбург: Изд-во Урал. ун-та, 2016. 60 с.
11. Stepanov A.V., Chunovkina A.G. On choosing two-sided power distribution for measurement data // Advanced Mathematical and Computational Tools in Metrology and Testing XIII Series on Advances in Mathematics for Applied Sciences, 2024. Vol. 94. P. 268–277. DOI: 10.1142/9789819800674_0025.
12. Степанов А.В., Чуновкина А.Г. Об одном методе подбора закона распределения из семейства TSP и его свойствах // Материалы международной Воронежской зимней математической школы. Воронеж, ВГУ, 2024. С. 238–241.
13. Степанов А.В., Чуновкина А.Г. О применении TSP-распределения при анализе многократных неравноточных результатов измерений // Уфимская осенняя математическая школа - 2024: материалы международной научной конференции, 2024. Т. 2. С. 308–309.
14. Степанов А.В., Чуновкина А.Г. Об оценке параметров асимметричного TSP распределения и его применении // Вероятностные методы в дискретной математике. Тезисы докладов XI международной Петрозаводской конференции, 2024. Петрозаводск: КарНЦ РАН, 2024. С. 106–108.
15. Van Dorp J.R., Kotz S. The standard two-sided power distribution and its properties // The American Statistician. 2002. Vol. 56, no. 2. P. 90–99.
16. Kotz S., van Dorp J.R. Beyond Beta: Other Continuous Families of Distributions with Bounded Support and Applications. World Scientific Publishing, 2004.
17. Stoudt S., Pintar A., Possolo A. Coverage Intervals // Journal of Research of the National Institute of Standards and Technology. 2021. Vol. 126. DOI: 10.6028/jres.126.004.
18. Тырсин А.Н. Метод подбора наилучшего закона распределения непрерывной случайной величины на основе обратного отображения // Вестник ЮУрГУ. Серия: Математика. Механика. Физика. 2017. Т. 9, № 1. С. 31–38.
19. https://github.com/stepanov17/pavt_examples

On Some Parallel Algorithm for NFA Beterminization*

M.E. Abramyan^{1,2}, N.I. Krainukov¹

Faculty of Computational Mathematics and Cybernetics, Shenzhen MSU-BIT University,
Shenzhen, China,¹
Algebra and Discrete Mathematics Department, Southern Federal University, Rostov-on-Don,
Russian Federation²

The many problems are in theory of automata, one from this problem that is the problem of determinization of nondeterministic finite automata (NFA) [1].

The every letter of alphabet Σ produces the transformation of states of DFA A , but for NFA every letter produces the transformation of power set 2^Q of states Q . The subset construction converts an NFA to DFA. Every states of correspondent DFA is the subset of states NFA.

The main idea of algorithm of parallels determinization NFA is like the idea breadth-first search (BFS) in theory algorithms on graphs [2, 3].

For subset construction on parallel determinization NFA, three hash tables are used: **Visited**, **Csstates** (current subsets) and **Nsstates** (next subsets). The Hash table keys are sets of states of the NFA, and Hash table values are unique integers ($key \rightarrow int$).

For saving a transition table DFA is also used a **TransitionList** of triples (KeyIntCsstates, Letter, KeyIntNsstates). This list consists of already discovered part of transition of the equivalent DFA.

Algorithm parallel determinization NFA is showed in Fig. 1:

```
1: void ParallelDeterminization(Node r)
2: Visited=Csstates= $\emptyset$ ; Nsstates=r; /*V,Current, and Next*/
3: r.lev=level=0;
4: for a in Alphabet do
5: repeat
6: Csstates=Nstates;
7: for State s  $\in$  Csstates do /*in parallel */
8: for State n  $\in$  Transition(s)(a) do /*in parallel */
9: Add(TransitionList,(s, a, n))
10: if (n $\notin$  Vizited) then
11: Nsstates = Nsstates  $\cup$  n ; Vizited = Vizited  $\cup$  n
12: n.lev=level+1
13: level++
14: until Nsstates== $\emptyset$ 
15: od; /* for a in Alphabet */
```

Fig. 1. Algorithm parallel determinization NFA

This paper studied parallelizing determinization algorithm of NFA in two different approaches of HPCGAP and OpenMP experimentally. Our approaches show that implementation the different models for parallelization may produce approximately equal speed if it is executed in the same conditions.

References

1. Lallement G. Semigroups and Combinatorial Applications. NJ, Wiley & Sons, Inc., 1979.
2. Cormen T., Leiserson C. Introduction to Algorithms. MIT Press and McGraw-Hill, 2001.
3. Roosta S.H. Parallel Processing and Parallel Algorithms. Springer, 2000. DOI: 10.1007/978-1-4612-1220-1.

* This work is supported by a grant from the research program of Chinese universities “Higher Education Stability Support Program” (Section “Shenzhen 2022 – Science, Technology and Innovation Commission of Shenzhen Municipality”) – 深圳市 2022 年高等院校定支持划助目.

Hybrid tractography analysis in patients with brain lesions^{*}

A. Minnigulova¹, O. Dragoy^{1,2}

Center for Language and Brain, HSE University¹,
Institute of Linguistics, Russian Academy of Sciences²

For non-invasive studies of the white matter of the brain in vivo, data from diffusion-weighted magnetic resonance imaging (MRI) are processed to reconstruct fiber pathways and extract quantitative metrics. During these processes, we encounter several challenges related to fiber crossing, partial volume effects, and inherent ambiguities as an inverse problem [1]. A novel hybrid approach based on both deterministic and probabilistic tractography has been proposed as a potential solution to improve segmentation [1]. Since this method is built on reinforcement learning and our research samples are quite large, a high-performance computational tool is essential. Thus, we utilized the HSE University HPC Cluster "cHARISMa" to reconstruct 171 bundles for each of the 267 patients at two time points: before and after surgery. Importantly, the HPC cluster facilitates parallel computations, specifically, NVIDIA CUDA through the use of Nvidia GPUs, allowing us to run multiple processes simultaneously to optimize both human and technical resources.

In the current research, we are interested in language-related tracts, including three segments of the arcuate fasciculus, the frontal aslant tract, the inferior longitudinal fasciculus, the inferior fronto-occipital fasciculus, and the uncinate fasciculus bilaterally (see Fig. 1). To quantitatively investigate the differences in fiber characteristics before and after resection and their implications for language impairments, we extract diffusion and volumetric metrics for each of the tracts, followed by subsequent statistical analysis.

To date, we have compared the selected tracts and observed a decrease in both volume and number of fibers across all bundles. The next steps include investigating associations between tractography metrics and language impairments assessed by the Russian Aphasia Test (RAT) [2]. The RAT allows us to comprehensively assess language abilities [2]. Thus, after statistical analysis, we will be able to determine the causality of specific language impairments in patients following left-hemisphere glioma or epileptogenic lesion surgery.

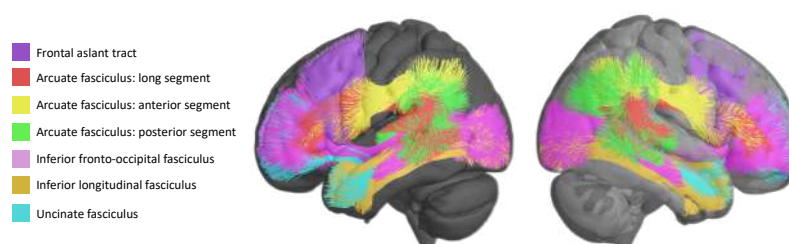


Figure 1. Associative tracts included into the analysis

References

1. Cabeen R.P., Toga A.W. Reinforcement Tractography: A Hybrid Approach for Robust Segmentation of Complex Fiber Bundles // Proceedings of the 17th International Symposium on Biomedical Imaging, ISBI 2020, April 3-7, 2020, Iowa City, Iowa, USA. IEEE, 2020. P. 999–1003. DOI: 10.1109/ISBI45749.2020.9098371.
2. Ivanova M.V., Akinina Yu.S., Soloukhina O.A. et al. The Russian Aphasia Test: The first comprehensive, quantitative, standardized, and computerized aphasia language battery in Russian // Plos one. 2021. Vol. 16, no. 11. P. e0258946. DOI: 10.1371/journal.pone.0258946.

^{*} This article is an output of a research project implemented as part of the Basic Research Program at the National Research University Higher School of Economics (HSE University).

Cache Oblivious Fast 2D Fourier Transform*

G.Y. Zavyalov, A.S. Levin, M.E. Beketov

HSE University

Fourier transform (FT) is a fundamental operation, providing basis for numerous methods in pure and applied mathematics. The celebrated Fast Fourier Transform (FFT) algorithm of Cooley and Tukey [1] enables efficient computation of discrete FT (DFT) of 1-dimensional signals. When the dimension is higher, in practice DFT is typically done by running FFT along each dimension (axis) of the signal array. While in fact one can explicitly generalize the idea behind FFT to 2+ dimensions: this is known as vector-radix FFT algorithm [2, 3], recently rediscovered [4]. Despite the 25% reduction of required arithmetic operations (complex multiplications), practical implementations of this algorithm are lacking in popular software libraries specializing in FTs.

In present work we study space complexity and cache behavior of said algorithm (which we refer to as F2FT for Fast 2D Fourier Transform), and show that it is in fact cache-oblivious [5], a remarkable property meaning that the number of cache misses is independent of the cache memory architecture. We provide our own C++ implementation of F2FT, which utilizes a custom layout of the processed arrays in memory (based on bit-reversal techniques). We benchmark our implementation versus that found in PocketFFT C++ library (a development over the FFTPack library [7]), as used in SciPy [6] of version 1.13.1. In PocketFFT, 2D DFT is achieved by separately doing FFT along two axes. We find a noticeable advantage of our implementation in terms of runtime and (write) cache misses – run on CPUs of both ARM64 and x86_64 architectures (in case when all arrays fit into RAM).

References

1. Cooley J., Tukey J. An algorithm for the machine calculation of complex Fourier series // Mathematics of computation. 1965. Vol. 19, no. 90. P. 297–301. DOI: 10.2307/2003354.
2. Rivard G. Direct fast Fourier transform of bivariate functions // IEEE Transactions on Acoustics, Speech, and Signal Processing. 1977. Vol. 25, no. 3. P. 250–252. DOI: 10.1109/TASSP.1977.1162951.
3. Harris D., McClellan J., Chan D. et al. Vector radix fast Fourier transform // ICASSP'77. IEEE International Conference on Acoustics, Speech, and Signal Processing. 1977. Vol. 2. P. 548–551. DOI: 10.1109/ICASSP.1977.1170349.
4. Tutatchikov V. Two-dimensional fast Fourier transform: Butterfly in analog of Cooley-Tukey algorithm // 11th International Forum on Strategic Technology (IFOST). 2016. P. 495–498. DOI: 10.1109/IFOST.2016.7884163.
5. Frigo M., Leiserson C., Prokop H. Cache-oblivious algorithms // 40th Annual Symposium on Foundations of Computer Science. 1999. P. 285–297. DOI: 10.1109/SFPCS.1999.814600.
6. Virtanen P., Gommers R., Oliphant T. et al. SciPy 1.0: fundamental algorithms for scientific computing in Python // Nature methods. 2020. Vol. 17., no. 3. P. 261–272. DOI: 10.1038/s41592-019-0686-2.
7. Swarztrauber P. Vectorizing the FFTs // Parallel computations. 1982. P. 51–83. DOI: 10.1016/B978-0-12-592101-5.50007-5.

*The work of M. Beketov was supported by HSE University Basic Research Program.

Multi-Agent Intelligent Decision Support System

E.I. Zaytsev, E.V. Nurmatova

MIREA - Russian Technological University

The development of the Multi-Agent Intelligent Decision Support System (MAIDSS) consisting of software modules that independently solve their private problems and combine their local capabilities to solve a common problem, extends the scope of automation of information processes to poorly formalized areas in which to make informed decisions it is necessary to perform logical inference, analyze large amounts of data and adapt to the conditions of uncertainty and non-stationarity of the environment.

According to the agent-based approach in the MAIDSS a complex problem is reduced into sub-tasks, which are solved by intelligent (applied) software agents. The cognitive data structures, logical inference methods, and artificial neural networks included in the applied software agents allow agents to act rationally and adapt to the environment. The Knowledge Representation and Processing Subsystem (KRPS) and Multi-Agent Reinforcement Learning (MARL) allow applied software agents to accumulate experience through interaction with each other and with the environment [1–8].

To create a highly efficient MAIDSS, it is necessary to develop a subsystem for managing a Distributed Knowledge Base (DKB), which should synthesize the optimal logical structure of the DKB or adapt the existing structure to the changing conditions of non-stationary environment, as well as timely and adequately respond to situations arising from the deficit of computational or time resources.

In order to increase manageability and accessibility of the DKB, the division of stored knowledge base objects into batches – partitions with different physical storage parameters is used. Some fragments of the DKB are of uncertain nature. Automating the processing of such fragments allows modeling complex causal relationships between concepts of some problem domain and increasing the range of sampling by query.

References

1. Zaytsev E.I., Khalabiya R.F., Stepanova I.V., Bunina L.V. Multi-Agent System of Knowledge Representation and Processing // Proceedings of the Fourth International Scientific Conference “Intelligent Information Technologies for Industry” (IITI'19). Vol. 1156. Springer, 2020. P. 131–141. DOI: 10.1007/978-3-030-50097-9_14.
2. Zaytsev E.I., Nurmatova E.V. Approach to knowledge management and the development of a multi-agent knowledge representation and processing system // Russian Technological Journal. 2023. Vol. 11, no. 4. P. 16–25. DOI: 10.32362/2500-316X-2023-11-4-16-25.
3. Nurmatova E.V., Zaitsev E.I. Modeling of objects state estimation on the basis of queries with fuzzy conditions // Engineering Bulletin of Don. 2024. No. 1 (109). (in Russian)
4. Sutton R.S., Barto A.G. Reinforcement Learning: An Introduction. Second Edition. MIT Press, Cambridge, MA, 2018. 552 p.
5. Barreto A., Dabney W., Munos R. et al. Successor features for transfer in reinforcement learning // Proceedings of the 31st International Conference on Neural Information Processing Systems. Curran Associates Inc., 2017. P. 4058–4068. DOI: 10.5555/3294996.3295161.
6. Jurgenson T., Avner O., Groshev E., Tamar A. Sub-goal trees: a framework for goal-based reinforcement learning // Proceedings of the 37th International Conference on Machine Learning. JMLR.org, 2020. Article 466. P. 5020–5030. DOI: 10.5555/3524938.3525404.
7. Zhao R., Sun X., Tresp V. Maximum entropy-regularized multi-goal reinforcement learning // Proceedings of the 36th International Conference on Machine Learning. Vol. 97. PMLR, 2019. P. 7553–7562.
8. Pong V., Gu S., Dalal M., Levine S. Temporal difference models: Model-free deep RL for model-based control // International Conference on Learning Representations, 2018.

Разработка архитектуры классификатора для оценки состояния объектов инфраструктуры с применением нейронных сетей

И.А. Абрамов, А.Ю. Камакин, Н.Д. Моисеев

Национальный исследовательский университет «Высшая школа экономики»

Геопространственный анализ – процесс изучения данных, содержащих географические компоненты, для выявления и изучения пространственных закономерностей. В наше время особое внимание уделяется его применению при планировании инфраструктуры городов [1]. Качественный анализ геопространственных данных позволяет выявлять проблемные зоны отдельных улиц и районов, что способствует эффективному распределению ресурсов при реализации проектов, направленных на улучшение качества жизни горожан.

С увеличением объема данных, содержащих географическую информацию, все больший приоритет отдается автоматизации их анализа, а с развитием глубокого обучения (deep learning), чаще встает вопрос о его применении в задачах геопространственного анализа. Геопространственный анализ активно использует методы автоматизированного сбора данных [2], что позволяет эффективно формировать датасеты для обучения моделей искусственного интеллекта.

В данной работе представлен программный комплекс, включающий в себя модуль сбора и подготовки изображений для формирования датасета, модуль классификаторов, а также модуль интерактивной веб-карты, демонстрирующей результат работы классификаторов. Карта предоставляет пользователю информацию о состоянии зданий, их назначении, а также иную информацию об особенностях различных районов города Москвы.

Модуль классификаторов состоит из комплексной архитектуры, состоящий из нескольких предобученных сверточных нейронных сетей, предназначенных для определения различных характеристик здания с изображения. Подход с использованием нескольких классификаторов позволяет получить более высокую точность при определении типа строения (жилое, коммерческое, промышленное и так далее) в условиях неравномерного распределения классов в датасете.

Помимо подготовки архитектуры, особое внимание уделяется исследованию возможности интеграции методов обработки естественного языка (NLP) и визуально-языковых моделей (VLM) для повышения точности классификации. Среди рассматриваемых моделей VLM, представлены PaliGemma 2, демонстрирующая высокую точность определения параметров объектов на изображении.

Исследование существующих архитектур классификаторов, также является одним из аспектов данной работы. Проведено исследование таких известных архитектур, как YOLO, RF-DETR, ResNet-50 и VGG19. Оценивались не только точность, в виде сравнения метрик Precision и Mean Average Precision (mAP), но и скорость обработки изображения. Данная оценка позволит выбрать наиболее подходящую модель, которая будет использоваться в нашем программном комплексе.

Литература

1. Kovacs-Györi A., Ristea A., Havas C., Mehaffy M., Hochmair H., Resch B., Juhasz L., Lehner A., Ramasubramanian L., Blaschke T. Opportunities and Challenges of Geospatial Analysis for Promoting Urban Livability in the Era of Big Data and Machine Learning // ISPRS International Journal of Geo-Information. 2020. Vol. 9, no. 12. P. 752. DOI: 10.3390/ijgi9120752.
2. Nekrasov H.A., Polivoda D.E., Prokofyeva E.N. The Geospatial Data Mining Concept Using Scrapping Technology // 2020 IEEE Moscow Workshop on Electronic and Networking Technologies, MWENT 2020, March 1-6, 2020, Moscow, Russia. IEEE, 2020. DOI: 10.1109/mwent47943.2020.9067406.

Улучшенный вывод типов в Python с использованием глубокого обучения сходству на основе графового представления кода*

Г.А. Арутюнов, С.М. Авдошин, А.М. Литвиненко
НИУ ВШЭ, ул. Мясницкая, д. 20, г. Москва, Россия

Целью данной работы стало исследование альтернативных подходов к представлению кода, таких как графы абстрактного синтаксического дерева и потока данных. Для этого была выбрана задача определения аннотаций типов в исходном коде программ на языке Python.

В предыдущей работе была разработана модель GraphTyper [1]. Модель является чистым Трансформером без каких-либо изменений в исходную архитектуру. Исходный код в графовом представлении переводится в последовательность благодаря вычислению вложений от собственных векторов и значений матрицы смежности, которые позволяют закодировать структуру графа [2]. Тем не менее, у первой версии модели был ограничен размер словаря. Увеличение размера словаря привело бы к увеличению количества параметров, а значит и веса модели.

Для решения проблемы во второй версии модели был использован метод глубокого обучения сходству [3]. Для обучения модели был собран датасет из 100 репозиторий на языке Python. Исходный код был преобразован в граф посредством метода, описанного в работе Typilus [3]. Далее, граф был преобразован в формат, который может быть обработан моделью TokenGT [2]. Кроме того, была адаптирована триплетная функция потерь из работы Typilus. Наконец, был имплементирован метод предсказания аннотаций типов из полученных векторных вложений с использованием k ближайших соседей. Улучшенная модель GraphTyper продемонстрировала точность 67.8% для типов с параметрами и 81.2% при отбрасывании параметров типов против 50.7% и 64.58% соответственно для первой версии модели.

Улучшения модели GraphTyper позволили разработать универсальную архитектуру модели, которая может обучаться на графовом представлении кода. Данное достижение позволит перейти в будущих исследованиях к более сложным задачам, а именно к применению данной модели для поиска и классификации уязвимостей в исходном коде программ и смарт-контрактов.

Литература

1. Arutyunov G.A., Avdoshin S.M. GraphTyper: Neural Types Inference from Code Represented as Graph // Proceedings of the Institute for System Programming of the RAS. 2024. Vol. 36, no. 4. P. 69–80. DOI: 10.15514/ispras-2024-36(4)-6.
2. Kim J., Nguyen T.D., Min S. et al. Pure Transformers are Powerful Graph Learners // CoRR. 2022. Vol. abs/2207.02505. arXiv: 2207.02505 [cs.LG]. DOI: 10.48550/arXiv.2207.02505.
3. Allamanis M., Barr E.T., Ducousso S., Gao Z. Typilus: Neural Type Hints // Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation. ACM, 2020. P. 91–105. DOI: 10.1145/3385412.3385997.

*Исследование выполнено при поддержке вычислительных ресурсов HPC кластера НИУ ВШЭ.

Подготовка ИТ-специалистов в области квантовых вычислений и квантовой теории информации

С.В. Борзунов, С.Д. Кургалин

ФГБОУ ВО «Воронежский государственный университет»

Решение самых актуальных задач требует в настоящее время применения как высокопроизводительных, так и перспективных квантовых вычислений. На кафедре цифровых технологий факультета компьютерных наук Воронежского государственного университета подготовка специалистов в области квантовых вычислений и квантовой теории информации ведется в течение более 20 лет. Накоплен большой опыт, выработаны оптимальные методики преподавания дисциплин, сформированы их связи с другими читаемыми математическими и физическими курсами. Система подготовки включает в себя предметы: квантовая теория, квантовые вычисления, квантовые информационные системы, квантовая теория информации, языки программирования квантовых компьютеров, линейная алгебра в квантовых вычислениях, алгоритмы коррекции ошибок, выполняются курсовые и выпускные квалификационные работы по этой тематике. В рамках практико-ориентированного подхода на базе ресурсов кафедральной Лаборатории суперкомпьютерных технологий и больших данных создан специализированный учебно-методический комплекс, охватывающий широкий спектр как фундаментальных основ квантовых вычислений и квантовой теории информации, так и особенности их практической реализации [1–4]. Он обеспечивает формирование компетенций ИТ-специалистов в области применения квантовых информационных систем для решения ресурсоемких задач, а также в области новейших методов передачи информации.



Рис. 1. Комплекс учебных пособий по квантовым вычислениям

Литература

1. Kurgalin S., Borzunov S. Concise Guide to Quantum Computing: Algorithms, Exercises, and Implementations. Springer, 2021. (Series: Texts in Computer Science). 122 p. DOI: 10.1007/978-3-030-65052-0.
2. Борзунов С.В., Кургалин С.Д. Основы квантовых вычислений и квантовой теории информации. СПб.: БХВ-Петербург, 2025. 248 с.
3. Борзунов С.В., Кургалин С.Д. Квантовые вычисления. СПб.: БХВ-Петербург, 2022. 144 с.
4. Борзунов С.В., Кургалин С.Д. Основы квантовых вычислений (для программистов). Воронеж: Издательский дом ВГУ, 2020. 100 с.

Эволюционные алгоритмы с параллельными операторами для задачи составления расписаний выполнения заказов клиентов*

П.А. Борисовский, Ю.В. Захарова, А.О. Захаров

Омский филиал Института математики им. С.Л. Соболева СО РАН

Рассматривается NP-трудная задача составления расписаний выполнения заказов n клиентов [1, 2]. Заказы клиентов включают в себя часть или все m продуктов. Задана длительность производства продукта j в заказе i , обозначаемая через $p_{ij} \geq 0$, $i \in I$, $j \in J$. Критерием оптимизации выступает сумма моментов завершения выполнения заказов.

Исследуется два варианта задачи: когда для производства всех продуктов используется одна производственная единица с учетом переналадки оборудования и когда каждый продукт производится на специализированном устройстве.

Для решения задачи предлагается два эволюционных алгоритма. Одним из таких алгоритмов выступает генетический алгоритм с оптимальной рекомбинацией [3]. Здесь в операторе кроссинговера решается задача оптимальной рекомбинации, которая формулируется с учетом свойства передачи генов и направлена на построение такого потомка, который является лучшим по целевой функции и наследует свойства родительских особей. Для решения задачи оптимальной рекомбинации (как точного, так и приближенного) используется распараллеленная на CPU версия алгоритма перебора совершенных паросочетаний в специальном двудольном графе. В операторе мутации используется $(1+\lambda)$ [4] эволюционная стратегия. Второй алгоритм предназначен для выполнения на GPU и представляет собой параллельную реализацию вероятностного итеративного локального поиска с использованием схемы «иди с победителями» [4] для отбора решений.

Генетический алгоритм тестировался на сервере AMD EPYC 7502 CPU, а итеративный локальный поиск – на компьютере Tesla V100 GPU. Результаты экспериментального исследования на сериях примеров из известных библиотек [1, 2] показали, что генетический алгоритм дает конкурентоспособные результаты по сравнению с известными в литературе алгоритмами [1, 2], а итеративный локальный поиск с распараллеливанием на GPU демонстрирует статистически значимое преимущество и позволил найти новые рекордные решения для задач с $n = 100, 200$ из библиотеки [2].

Литература

1. Hazir O., Gunalay Y., Erel E. Customer order scheduling problem: a comparative metaheuristics study // The International Journal of Advanced Manufacturing Technology. 2008. Vol. 37. P. 589–598. DOI: 10.1007/s00170-007-0998-8.
2. Shi Zh., Ma H., Ren M., Wu T., Yu A.J. A learning-based two-stage optimization method for customer order scheduling // Comput. Oper. Res. 2021. Vol. 136. 18 p.
3. Ereemeev A., Kovalenko Y. Optimal recombination in genetic algorithms for combinatorial optimization problems: part II // Yugoslav. J. Oper. Res. 2014. Vol. 24, no. 2. P. 165–186. DOI: 10.2298/YJOR131030041E.
4. Borisovsky P.A. A parallel “Go with the winners” algorithm for some scheduling problems // Jour. Appl. Indust. Math. 2023. Vol. 17, no. 4. P. 687–697. DOI: 10.1134/S1990478923040014.

*Исследование выполнено за счет гранта Российского научного фонда № 22-71-10015, <https://rscf.ru/project/22-71-10015/>.

NNMD: программный пакет на Python для обучения нейронных сетей Бехлера—Парринелло

А.С. Будников, А.В. Романов

Воронежский государственный университет

Применение нейронных сетей для исследования свойств различных материалов или молекул является распространенной практикой в задачах молекулярно-динамического моделирования, например, в материаловедении и биохимии. В частности, нейронные сети позволяют определить специальную модели межатомных потенциалов – машинно-обучаемые потенциалы, или нейронную сеть Бехлера—Парринелло [1], которые могут быть обучены на экспериментальных данных без серьезных временных затрат на построение таких моделей, а затем реиспользованы возможностью дообучения под новые данные.

Нейронная сеть Бехлера—Парринелло (Behler—Parrinello Neural Network) является нейронной сетью высокой размерности, т.е. она состоит из подсетей, называемых атомными НС (AtomicNN), которые обучаются совместно. Как дескрипторы используются функции Бехлера—Парринелло, которые являются функциями от межатомного расстояния, что позволяет запоминать для каждого атома системы его окружение в пределах сферы с радиусом отсечки.

Для атомов одного химического элемента подсеть будет одинаковой, поэтому количество подсетей определяется количеством химических элементов в рассматриваемой системе. Нейронная сеть такой структуры способна вычислять энергию системы через алгоритм прямое распространения, силу через алгоритм обратного распространения ошибки, так как сила – градиент энергии. В качестве функции ошибки выступает взвешенная функция среднеквадратичной ошибки, где веса регулируют вклад ошибки по энергии и силам.

NNMD (Neural Networks for Molecular Dynamics) [2, 3] реализован с помощью языка программирования Python, а также фреймворка машинного обучения PyTorch, который также используется при вычислении значений атомно-центрированных симметричных функций Бехлера—Парринелло из-за возможности векторизации алгоритма вычисления дескрипторов. В пакете реализован интерфейс для калькулятора ASE под молекулярно-динамические эксперименты с возможностью использования предобученной сети при работе симулятора.

Программный пакет разбит на несколько модулей: io (input/output), nn (нейронные сети и связанные с ними классы), features (для вычисления дескрипторов), md (имплементация калькулятора для ASE). Благодаря модульной структуре пользователь может использовать только необходимые инструменты без необходимости взаимодействовать со всеми компонентами пакета, а также расширять и адаптировать код под конкретные нужды и задачи.

В дальнейшем планируется доработать проект таким образом, чтобы пакет смог отрабатывать весь цикл обучения и применения сетей в реальных экспериментах с возможностью масштабирования результатов. Также при невозможности повышения производительности и точности через операции PyTorch вычисления будут перенесены на язык C++ с интеграцией кода на нем в пакет посредством биндинга Python/C++, что может значительно упростить вычислительную часть задачи, а также потенциально ускорить за счет применения параллельных технологий, например, CUDA.

Литература

1. Behler J. Atom-centered symmetry functions for constructing high-dimensional neural network potentials // The Journal of Chemical Physics. 2011. Vol. 134, no. 3. DOI: 10.1063/1.3553717.
2. Проект NNMD. URL: <https://github.com/ded-otshel'nik/nnm-d> (дата обращения: 01.04.2025)
3. Будников А.С., Романов А.В. GPU-реализация машинно-обучаемых потенциалов для молекулярной динамики // Труды молодых ученых факультета компьютерных наук Воронежского государственного университета. Т. 4. Воронеж: 2024. С. 26–33.

Метод онлайн-визуализации параллельных программ

П.А. Васёв

Институт математики и механики им. Н.Н. Красовского УрО РАН, Екатеринбург

Онлайн-визуализация параллельных программ подразумевает: 1) возможность наблюдения за ходом вычисления, и 2) возможность влияния на ход вычисления. Такая визуализация несет существенные выгоды [1], но сопряжена с определенными техническими сложностями:

а) в параллельном программировании до сих пор не предложено стандартных средств формирования визуальных образов для визуализации по ходу вычисления;

б) при больших вычислениях рендеринг визуальных образов необходимо осуществлять на узлах, где происходит вычисление [2], что обычно требует особых компетенций.

Осмысление пункта (а) приводит к наблюдению, что в то же время в параллельных программах доступен оператор `print`, который крайне удобен и успешно выводит информацию в лог-файл программы или файлы. Но лог-записи и файлы медленны, и по своей природе не визуальны. В то же время существует подход *CinemaScience* [3], который частично решает задачу онлайн-визуализации и использует `print`-подобный подход. Возникает закономерный вопрос, почему `print` удобен и доступен, и ему до сих пор не найдено достойных альтернатив?

В настоящей работе предлагается следующий подход. Пусть параллельная программа это клиент графической среды *Grafix* [4], по аналогии с клиентом среды *Gradio*, используемой при обучении нейросетей. Параллельная программа при запуске соединяется с заданным веб-сервером, который отвечает за визуализацию интерфейса программы. В ответ от веб-сервера программа получает веб-ссылку, по которой сервер будет готов отобразить визуализацию и управляющий интерфейс программы. Параллельная программа печатает эту ссылку в выходной поток, и далее производит вычисления как обычно.

При необходимости пользователь открывает поданную ему в потоке ссылку, и параллельная программа получает от веб-сервера сигнал *gui_attached*. В ответ на этот сигнал программа формирует графический интерфейс (в формате json, см. [4]) и отправляет его веб-серверу. Далее программа наполняет интерфейс актуальными данными о своем состоянии.

Пользователь наблюдает эти данные, и при необходимости формирует управляющие воздействия, которые определяются графическим интерфейсом программы. Сигналы об этих воздействиях поступают в параллельную программу, и она имеет возможность реагировать на них согласно своему устройству.

Следует отметить, что среда *Grafix* интегрирована со средой *Parallel Programming Kit* [5], но не требует ее использования. Любая параллельная программа может зарегистрироваться по заданным протоколам и представить свое состояние и варианты взаимодействия в предлагаемом веб-интерфейсе.

Литература

1. Vasev P. Analyzing an Ideas Used in Modern HPC Computation Steering // 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBREIT), Yekaterinburg, Russia, 2020. P. 1–4. DOI: 10.1109/USBREIT48449.2020.9117685.
2. Moreland K. The tensions of in situ visualization // IEEE Computer Graphics & Applications, March/April 2016. P. 5–9. DOI: 10.1109/MCG.2016.35.
3. Ahrens J. et al. An image-based approach to extreme scale in situ visualization and analysis // Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '14). IEEE, 2014. P. 424–434. DOI: 10.1109/SC.2014.40.
4. Васёв П.А. Сетевая технология программирования визуальных интерфейсов // GraphiCon 2024: Материалы 34-й Международной конференции по компьютерной графике и машинному зрению, Омск, 17-19 сентября 2024 года. Омск: Омский государственный технический университет, 2024. С. 354–360. DOI: 10.25206/978-5-8149-3873-2-2024-354-360.
5. Васёв П.А. Среда параллельного программирования *Parallel Programming Kit* // Тезисы докладов Национального суперкомпьютерного форума (НСКФ-2024), 26-29 ноября 2024, Институт программных систем им. А.К. Айламазяна РАН, Переславль-Залесский.

Сравнительный анализ многопоточной производительности CPU и GPU в JDFT-моделировании гетероструктур металл-графен*

Д.В. Гордиенко^{1,2}, С.В. Павлов¹, В.А. Кисленко¹, С.А. Кисленко¹

Объединенный институт высоких температур РАН¹,
Московский физико-технический институт²

В работе исследована производительность CPU и GPU на примере расчетов в рамках теории функционала электронной плотности в программе JDFTx [1, 2]. Произведены замеры скорости счета при варьировании числа OpenMP-потоков и MPI-процессов. Тестирование производилось в большом каноническом ансамбле при заданной величине химического потенциала электронов в гетероструктуре графен/золото с применением континуальной модели электролита CANDLE [3], симулирующей 0.1 М водный раствор KCl. Также выполнялось моделирование в каноническом ансамбле электронейтральной системы. Расчеты такого рода принципиально важны для реалистичного моделирования электрохимических систем и экспериментальных аналитических процедур, которые часто производятся в потенциостатическом режиме или при потенциале нулевого заряда в контакте с раствором электролита. Еще одной особенностью работы является осуществление тестирования на примере длительных расчетов до 24 часов, в отличие от часто встречающихся замеров на пробных системах длительностью в несколько минут.

Результаты демонстрируют существенный выигрыш от применения графических процессоров в рассматриваемом классе задач. Получены аналитические оценки необходимого объема оперативной памяти для проведения вычислений как в пределах одного MPI-процесса, так и для всей программы. Обнаружено, что при работе на GPU-системах следует запускать не более одного MPI-процесса на одной GPU, что эффективней как с точки зрения скорости вычислений, так и необходимого объема памяти, что существенно ограничивает применение графических процессоров на практике. При проведении моделирования на CPU-системах следует задавать как можно большее число MPI-процессов, но не больше числа неприводимых k-точек, внутри каждого MPI-процесса запускать наибольшее возможное число OpenMP-потоков.

Литература

1. Sundararaman R., Vigil-Fowler D., Schwarz K. Improving the accuracy of atomistic simulations of the electrochemical interface // Chemical reviews. 2022. Vol. 122, no. 12. P. 10651–10674. DOI: 10.1021/acs.chemrev.1c00800.
2. Sundararaman R., Letchworth-Weaver K., Schwarz K.A. et al. JDFTx: Software for joint density-functional theory // SoftwareX. 2017. Vol. 6. P. 278–284. DOI: 10.1016/j.softx.2017.10.006.
3. Sundararaman R., Goddard W.A. The charge-asymmetric nonlocally determined localelectric (CANDLE) solvation model // The Journal of chemical physics. 2015. Vol. 142, no. 6. DOI: 10.1063/1.4907731.

* Работа выполнена при поддержке Министерства науки и высшего образования Российской Федерации (Государственное задание № 075-00269-25-00).

Модификация и моделирование алгоритма поиска данных в служебной таблице маршрутизатора контент-ориентированных сетей

М.Н. Дегаев

ФГБОУ ВО Поволжский государственный технологический университет

За последнее десятилетие в связи с увеличением объемов данных возникла необходимость создания новых сетей с высокой пропускной способностью, использующих технологию коммутации пакетов. Быстрое развитие интернет-приложений и появление новых оптических средств связи также влияют на концепцию современных сетей передачи данных. Одним из перспективных решений в этом направлении являются сети, ориентированные на данные (Information-Centric Networking, ICN) [1].

Целью работы является модификация существующих алгоритмов поиска данных в хеш-таблицах с помощью индексирования элементов данных для выявления потенциальных возможностей уменьшения задержек (латентности) при обращении к служебным таблицам маршрутизатора ICN-сетей.

Объектами исследования являются контент-ориентированные сети и алгоритмы доставки данных пользователю. Предметом исследования является архитектура кэша в маршрутизаторах контента ICN-сетей.

В настоящее время существует 58 схем организации служебных таблиц ICN-сетей, из которых для таблиц Content Store, PIT и FIB предложено 7, 8 и 39 схем соответственно [2]. Кроме того, структура данных в этих трех таблицах представлена по-разному. Для FIB широко используется префиксное дерево, чтобы уменьшить потребление памяти, поскольку FIB должен поддерживать поиск самого длинного совпадающего префикса имени. Между тем, фильтр Блюма чаще используется в PIT, поскольку PIT требует более высокой частоты доступа к памяти.

Существуют три перспективных типа структур данных для реализации таблицы FIB: 1) FIB на основе фильтра Блюма, 2) FIB на основе Trie (префиксной таблицы) и 3) FIB на основе хэш-таблицы.

В работе для ускорения поиска данных в кэше маршрутизатора предложена новая структура таблицы FIB с использованием хеш-таблиц, состоящих из хеш-блоков и фильтра Блюма с поддержкой проверки коллизий.

Также разработаны модифицированные алгоритмы по генерации (вставки) хеш-значений и поиска в модифицированной таблице FIB. Алгоритмы используют хэш *mtmtr* [3], который обладает высокой сбалансированностью и низким уровнем коллизий при обработке большого массива данных для завершения процессов поиска хэшей.

Проведен анализ предлагаемого модифицированного метода с оценкой сложности алгоритмов, по результатам которого можно говорить о снижении вычислительной сложности и уменьшения задержек при поиске информации в кэше маршрутизатора.

Литература

1. Васяева Н.С., Дегаев М.Н. Исследование способов коммутации пакетов для коммутаторов Cisco // Инженерные кадры – будущее инновационной экономики России: Материалы VI Всероссийской студенческой конференции. Йошкар-Ола: ПГТУ, 2021. С. 28–35.
2. Li Z., Xu Y., Zhang B., Yan L., Liu K. Packet Forwarding in Named Data Networking: Requirements and Survey of Solutions // IEEE Communications Surveys & Tutorials. 2018. Vol. 21, no. 2. P. 1950–1987. DOI: 10.1109/COMST.2018.2880444.
3. Hash Functions all the way down. Aras' Blog. URL: <https://aras-p.info/blog/2016/08/02/Hash-Functions-all-the-way-down> (дата обращения: 20.01.2025).

О возможности применения на GPU предобусловливателя AIPS в качестве альтернативы ILU(0) в задаче гидродинамического моделирования нефтегазовых месторождений

А.С. Добровольцев, М.А. Сохатский, А.В. Юлдашев

Уфимский университет науки и технологий

Трехмерное гидродинамическое моделирование нефтегазовых месторождений – ресурсоемкая задача, вычислительным ядром которой является решение систем линейных алгебраических уравнений (СЛАУ). В целях ускорения расчетов в промышленные гидродинамические симуляторы внедряются оптимизированные методы решения СЛАУ, а также параллельные вычислительные технологии. Например, гидродинамический симулятор «РН-КИМ» [1], используемый в университете в целях обучения и научных исследований, имеет многопоточную версию для рабочих станций с многоядерными процессорами (CPU), а также адаптирован для работы на кластерных системах. Одним из новшеств является возможность ускорения гидродинамических расчетов за счет использования графических процессоров (GPU) на этапе решения СЛАУ.

Ранее проведенные исследования [2] показали возможность применения предобусловливателя AIPS, аппроксимирующего обратную матрицу на основе степенного разложения в ряд Неймана, в качестве альтернативы классическому алгоритму алгебраического многосеточного метода на первой ступени двухступенчатого предобусловливателя CPR [3]. Благодаря нетрудоёмкой процедуре построения, а также ориентированному на архитектуру CUDA параллельному алгоритму пакетного решения линейных систем с трехдиагональными матрицами, метод AIPS обеспечивает приемлемую масштабируемость процесса итерационного решения СЛАУ с двухступенчатым предобусловливателем на вычислительной системе с несколькими GPU.

На текущем этапе исследований рассматривается возможность применения предобусловливателя AIPS в качестве альтернативы методу ILU(0) – неполного LU-разложения без заполнения, который используется как одноступенчатый предобусловливатель либо на второй ступени CPR. Предпосылкой является то, что ILU(0) имеет низкий ресурс параллелизма и, несмотря на наличие модификаций, в том числе реализованных в оптимизированных библиотеках, например, cuSPARSE от NVIDIA, быстродействие ILU(0) на GPU зачастую «проседает» относительно CPU.

Проведена адаптация алгоритмов построения и применения предобусловливателя AIPS с учетом наличия мелкозернистой блочности (2x2, 3x3 или более в зависимости от используемой модели фильтрации) в структуре матриц, характерных для задачи гидродинамического моделирования нефтегазовых месторождений. В параллельном алгоритме пакетного решения линейных систем с трехдиагональными матрицами задействован метод блочной прогонки.

В итоге удалось достичь существенного ускорения при решении набора СЛАУ с размерностью от 2 до 18 миллионов на GPU NVIDIA A100 (CUDA Toolkit 11.8). Применение AIPS обеспечило снижение времени решения СЛАУ до 3.6 раз относительно ILU(0) в режиме одноступенчатого предобусловливателя, а также до 1.5 раз в качестве альтернативы на второй ступени CPR.

Литература

1. Гидродинамический симулятор РН-КИМ. URL: <https://rn.digital/rnkim> (дата обращения: 01.02.2025).
2. Юлдашев А.В., Репин Н.В., Спеле В.В. Параллельный предобусловливатель на основе степенного разложения обратной матрицы для решения разреженных линейных систем на графических процессорах // Вычислительные методы и программирование. 2019. Т. 20, № 4. С. 444–456. DOI: 10.26089/NumMet.v20r439.
3. Wallis J.R., Kendall R.P., Little T.E. Constrained residual acceleration of conjugate residual methods // Proc. SPE Reservoir Simulation Symposium. 1985. Article SPE-13536-MS. P. 415–428. DOI: 10.2118/13536-MS.

Об одном способе организации параллельной генерации сеток на основе декомпозиции расчетной области

А.Г. Дронов

ФГБОУ ВО МГТУ «СТАНКИН»

При математическом моделировании процессов дифференциальные уравнения в областях сложной геометрической формы, которые в основном формируются с использованием САПР, зачастую решаются с помощью численных методов с использованием расчетных сеток, представляющих дискретизацию исходной области в конечномерном пространстве.

Существует достаточно большое количество подходов и алгоритмов для формирования сеточных моделей. В то же время, широкую популярность получили вычислительные расчеты с помощью параллельных алгоритмов на системах с распределенной памятью. Анализ имеющихся решений [1] показал, что существует недостаток средств генерации сеток с применением параллельных вычислений.

В технологической цепочке «подготовка данных – расчет – анализ результатов» построение сеток относят к первому этапу. Использование здесь последовательных алгоритмов делает этап «узким местом», ухудшая эффективность распараллеливания согласно закону Амдала, а в некоторых случаях расчет делается и вовсе невозможным [2]. В связи с этим повышенный интерес представляет идея построения сеток с помощью параллельных или распределенных алгоритмов.

В статье делается упор на построение симплициальных сеток с использованием алгоритма движущегося фронта с входными данными в формате граничного представления [1, 2]. Расчетная сетка состоит из узлов, ребер, граней и ячеек. Метод состоит в дискретизации узлов и ребер и дальнейшем построении триангуляции поверхности и тетраэдризации пространства.

Алгоритм движущегося фронта формирует треугольные грани и тетраэдральные ячейки результирующей сетки, отступая от границ исходного объекта в направлении к его центру. На этапе триангуляции поверхности во фронт помещаются сформированные ребра, на этапе тетраэдризации – грани. На текущей итерации выбирается наиболее подходящая по критерию сущность, на которой строится грань или ячейка (в зависимости от этапа), не пересекающая текущий фронт. При добавлении нового узла также проверяется его близость к узлам фронта. Процесс продолжается до тех пор, пока существуют еще не пройденные ребра или грани.

Основная идея применения параллельности заключается в декомпозиции пространства на несколько областей с использованием направляющего вектора разбиения, каждая из которых отдается отдельному процессу. Для согласования построений на границах разбиения используется понятие общей зоны [1].

Если ребро при дискретизации не попадает в рабочее пространство процесса, то для него оно игнорируется. Если ни одна из вершин не принадлежит рабочей области процесса, происходит ожидание получения какой-то вершины от другого процесса. Далее строится дискретизация от имеющейся вершины ребра в области до попадания в общую зону между процессами, когда данные о посчитанной вершине, вместе с управлением, передаются соседнему процессу.

При триангуляции и тетраэдризации нужно определить, куда попадает новый узел. Если вершина попадает в область текущего процесса, коммуникация необходима лишь для проверки пересечений с текущим фронтом. Если узел лежит внутри процесса-соседа, ребро передается ему. Иначе выполняются действия в зависимости от того, какая область находится ближе.

Литература

1. Дронов А.Г. Об одном алгоритме генерации поверхностных сеток // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и Технические Науки. 2024. № 06. С. 74–79. DOI: 10.37882/2223-2966.2024.06.16.
2. Дронов А.Г., Болдарев А.С. О разработке параллельных алгоритмов построения расчетных сеток на системах с распределенной памятью // Перспективы науки. 2024. № 6 (177). С. 27–32.

Использование нейросетевого подхода для оценки влияния ионопровода на восстановление начальной координаты взаимодействий*

М.А. Жаров, К.В. Размыслов, Ф.Ф. Валиев, В.В. Монахов

Санкт-Петербургский государственный университет

Целью работы являлось решение с помощью нейросетевого подхода задачи определения координат столкновений частиц с использованием кольцевых детекторов на микроканальных пластинах и изучение влияния на результат различных конфигураций ионопровода коллайдера NICA.

Для анализа были выбраны столкновения Au-Au при энергии 7 ГэВ в системе центра масс. Для моделирования взаимодействий использовался генератор событий UrQMD 3.4 и программный пакет Geant4. Области детектирования располагались на расстоянии 4 метра от точки взаимодействия и представляли собой два кольцевых детектора с внутренним диаметром $d = 100$ мм и внешним $D = 600$ мм. Эта конфигурация была выбрана с учетом возможностей технической реализации системы для коллайдера NICA. Для решения задачи был разработан подход с применением нейронных сетей на основе работ [1, 2]. Полученные результаты отображены в табл. 1. Приведенные значения $\sqrt{MSE}$ соответствуют значениям для полного набора данных, усредненным по 10 моделированиям.

Таблица 1. Зависимость ошибки $\sqrt{MSE}$ в определении начальных координат и ее среднего разброса $\Delta\sqrt{MSE}$ от материала и толщины стенки ионопровода. Временное разрешение 200 пс, 64 пада

Материал	Железо		Углерод	
Толщина, мм	$\sqrt{MSE}$, см	$\Delta\sqrt{MSE}$, см	$\sqrt{MSE}$, см	$\Delta\sqrt{MSE}$, см
0	3.81	0.22	3.81	0.22
1.2	4.23	0.21	3.93	0.19
3	4.95	0.28	4.68	0.23
5	5.22	0.26	4.30	0.22
8	5.96	0.30	4.82	0.32

Использование ионопровода со стенками небольшой толщины (1–3 мм) не оказывает значительного влияния на результат. Однако при увеличении толщины стенки ионопровода эффект от прохождения частиц через материал стенки становится более выраженным, что ухудшает точность решения в 1.2–1.4 раза в зависимости от материала.

Литература

1. Сандул В.С., Феофилов Г.А., Валиев Ф.Ф. Исследование возможностей быстрого мониторинга столкновений пучков с использованием моделирования методом Монте-Карло и методов машинного обучения // Физика элементарных частиц и атомного ядра. 2023. Т. 54, № 4. С. 848–854.
2. Galaktionov K.A., Roudnev V.A., Valiev F.F. Artificial Neural Networks Application in Estimating the Impact Parameter in Heavy Ion Collisions Using the Microchannel Plate Detector Data // Physics of Atomic Nuclei. 2023. Vol. 86, no. 6. P. 1426–1432. DOI: 10.1134/S10637788230602.

* Работа выполнена при поддержке СПбГУ, шифр проекта 103821868.

Решение задачи регрессии булевых данных с помощью параллельного алгоритма генетического программирования*

А.О. Захаров

Омский филиал Института математики им. С.Л. Соболева СО РАН

Задача регрессии булевых данных состоит в восстановлении зависимости на основании данных входа и выхода. На вход подаются m булевых векторов $x^1, x^2, \dots, x^m$, $x^i \in \{0, 1\}^n$, на выходе известно булево значение $y^i \in \{0, 1\}$, $i \in \{1, \dots, m\}$. Необходимо найти функциональную зависимость $f(x)$ выхода от входа, которая минимизирует значение целевой функции на обучающей выборке $\sum_{i=1}^m (y^i - f(x^i))^2$. Рассматривается генетическое программирование [1] для решения задачи регрессии, где особями являются булевы функциональные деревья. Задается набор базовых функций $F = \{f_1, \dots, f_k\}$, которые располагаются в узлах деревьев. Листам соответствуют переменные $x_1, x_2, \dots, x_n$ или константы 0, 1. Искомая зависимость $f(x)$ представляет собой суперпозицию функций в узлах, считая от корня.

Как правило в генетическом программировании используются рандомизированные операторы скрещивания на деревьях [2]. Целью данной работы является исследование и внедрение оптимальной рекомбинации в генетическое программирование (для начала на примере булевых данных), о чем автором уже были предприняты попытки в [3]. Однако, в оптимизированных операторах остро стоит вопрос трудоемкости, так как на каждом шаге алгоритма происходит перебор допустимых потомков с целью найти оптимального потомка в смысле значения целевой функции. В данной работе исследуется распараллеливание вычисления целевой функции при генерации начальной популяции, в локальном поиске и в оптимизированном операторе скрещивания. Обучающая выборка делится на непересекающиеся подмножества $M_1, M_2, \dots, M_s$, и сумма $\sum_{i \in M_j} (y^i - f(x^i))^2$ считается в своем потоке j , $j = 1, \dots, s$. Потоки выполняются параллельно. В качестве экспериментальных данных использовались полные таблицы истинности булевых функций четности и мультиплексора, которые широко используются в литературе. Вычислительный эксперимент показал преимущество использования распараллеливания с ростом длины входа задачи: для функции четности – начиная с длины 8, для мультиплексора – начиная с длины 11. Параллельные и последовательные запуски выполнялись для одной и той же псевдослучайной последовательности и на одно и то же число итераций, что обеспечивало эквивалентность качества полученных решений. Эксперимент проводился на сервере AMD EPYC 7502 CPU.

Литература

1. Koza J.R., Keane M.A., Streeter M.J., Mydlowec W., Yu J., Lanza G. Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Springer, 2003.
2. Poli R., Page J. Solving high-order boolean parity problems with smooth uniform crossover, sub-machine code GP and demes // Genet. Program. Evolvable. Mach. 2000. Vol. 1. P. 37–56. DOI: 10.1023/A:1010068314282.
3. Zakharov A. Optimal recombination problem in genetic programming for Boolean functions // Numerical Computations: Theory and Algorithms, NUMTA 2023. Vol. 14476 / eds. by Y.D. Sergeyev, D.E. Kvasov, A. Astorino. Springer, Cham, 2025. P. 226–240. Lecture Notes in Computer Science. DOI: 10.1007/978-3-031-81241-5_16.

*Исследование выполнено за счет гранта Российского научного фонда № 22-71-10015, <https://rscf.ru/project/22-71-10015/>.

Высокопроизводительное моделирование процессов роста оксигидроксидных структур алюминия

Б.Д. Зиннуров, Г.С. Смирнов, Е.В. Тарарушкин

Национальный исследовательский университет «Высшая школа экономики»

Более 100 лет назад были получены высокопористые структуры алюминия путем окисления жидкого сплава Hg(Al) во влажной среде [1]. Второй раз материал привлекает внимание ученых уже к концу 20го века [2] и в начале 21го [2]. В результате его изучения они приходят к выводу, что по физико-химическим свойствам структура близка к классическим аэрогелям, в связи с чем находит применение в качестве мембран [4], сенсоров [5] и защитных покрытий [6] в экспериментальных установках разных сфер применения. Перед авторами же данной работы стоит задача в построении модели роста одномерных высокопористых (т.н. нанофибрилл) оксигидроксидов алюминия, промежуточный результат которого и будет представлен.

Компьютерное моделирование на основе методов квантовой молекулярной динамики проводится на кластере суперкомпьютера МИЭМ НИУ ВШЭ «сHARISMa». На первом этапе рассматривают ряд соединений Al-O-H при заранее заданной температуре на протяжении многого количества шагов. Полученные на выходе конфигурации атомов, силы и давления используются для создания межатомного потенциала на основе машинного обучения, который необходим в построении новой модели Al-O-H. Модель строится на теории функционала электронной плотности. Машинное обучение помогает компенсировать главное ограничение теории – в прошлом ее вычислительная сложность позволяла проводить преимущественно статические расчеты [7].

Литература

1. Wislicenus H. Zeitschrift für Chemie und Industrie der Kolloide // Steinkopff & Springer, 1908. Vol. 2, XI-XX.
2. Vignes J-L., Mazerolle L., Michel D. A Novel Method for Preparing Porous Alumina Objects // Key Engineering Materials, 1997. P. 132–136, 432–435. DOI: 10.4028/www.scientific.net/KEM.132-136.432.
3. Мартынов П., Асхадуллин Р., Юдинцев П., Ходан А. Новый метод синтеза наноматериалов на основе контролируемого селективного окисления жидкометаллических фаз // Новые промышленные технологии. 2008. Т. 4. С. 48–52.
4. Yin X. Effect of boehmite sol on the performance of alumina microfiltration membranes // Ceramics International. 2019. Vol. 45, no. 13. P. 16173–16179. DOI: 10.1016/j.ceramint.2019.05.138.
5. Mohan P. Boehmite nanorod/gold nanoparticle nanocomposite film for an easy-to-use optical humidity sensor // Sensor. Actuator. B. 2012. Vol. 168. P. 429–435. DOI: 10.1016/j.snb.2012.04.055.
6. Chen Q. The effect of AlOOH boehmite nanorods on mechanical property of hybrid composite coatings // Thin Solid Films. 2009. Vol. 517, no. 17. P. 4871–4874. DOI: 10.1016/j.tsf.2009.03.097.
7. Iijima S. One-dimensional nanowires of pseudoboehmite (aluminum oxyhydroxide γ -AlOOH) // Proceedings of the National Academy of Sciences of the United States of America. 2016. Vol. 113, no. 42. P. 11759–11764. DOI: 10.1073/pnas.1614059113.

Разработка методов искусственного интеллекта для управления внешними устройствами на основе ЭЭГ в замкнутом контуре биологической обратной связи

Е.П. Ивличев, С.Г. Сидоров

Ивановский государственный энергетический университет им. В.И. Ленина

Настоящая работа посвящена разработке и исследованию методов искусственного интеллекта для управления внешними устройствами на основе данных ЭЭГ (электроэнцефалограммы) в контуре биологической обратной связи. В контексте нейроинтерфейсов снижение вычислительной сложности и размера моделей критически важно для обеспечения работы в реальном времени и использования в портативных системах. Цель данной работы – предложить методы сокращения количества слоев и параметров в нейронных сетях, сохраняя при этом высокую точность распознавания команд управления. Результаты показывают перспективность применения данной технологии в системах нейроинтерфейсов.

Преимуществом разработанной архитектуры является разделение обработки простых и сложных признаков ЭЭГ. Сложные признаки обрабатываются с применением многослойной бинарной нейронной сетей, обеспечивая их более точное представление, тогда как простые признаки обрабатываются в однослойной БНС. Классификация реализована с использованием последовательности условных операторов, оценивающих параметры признака, и передающих входные данные в соответствующую ветвь вычислительного графа. Такой подход позволяет повысить эффективность обучения и увеличить точность предсказаний.

Экспериментальная часть включала регистрацию сигналов ЭЭГ от 10 участников при выполнении задач по управлению внешними устройствами. ЭЭГ записывались системой Нейрон-Спектр-СМ с частотой дискретизации 16кГц. Данные ЭЭГ подвергались фильтрации и выделению признаков, после чего классифицировались при помощи разработанной архитектуры БНС. Полученные результаты показали, что точность классификации достигает 81.5%, при этом количество параметров сети снижено на порядок по сравнению с сверточными и рекуррентными нейронными сетями без классификации признаков [1–3].

В данной работе предложен новый подход к созданию легковесных и детерминированных систем управления на основе ЭЭГ, использующих бинарные нейронные сети и метод классификации признаков по сложности. Полученные результаты указывают на перспективность применения БНС для построения эффективных, быстро обучаемых и детерминированных систем мозг-компьютер. Результаты демонстрируют сокращение времени обучения и повышение точности классификации при использовании предложенных методов. В будущих исследованиях планируется дальнейшее изучение и оптимизация архитектур БНС, применение их в замкнутом контуре биологической обратной связи, а также изучение робастности систем к шумам и артефактам в данных ЭЭГ.

Литература

1. Schirrmeister R.T., Springenberg J.T., Fiederer L.D.J. et al. Deep learning with convolutional neural networks for EEG decoding and visualization // Human Brain Mapping. 2017. Vol. 38, no. 11. P. 5391–5420. DOI: 10.1002/hbm.23730.
2. Srinivasan G., Khosla P. Binary neural networks for efficient brain-computer interfaces // 2019 IEEE Biomedical Circuits and Systems Conference (BioCAS). IEEE, 2019. P. 1–4.
3. Alvarez F., López J.A. Fast and energy-efficient EEG classification with Binarized Neural Networks // Sensors. 2020. Vol. 20, no. 21. Article 6155.

Модель с подстройкой параметров для модулярных вычислений в больших компьютерных диапазонах

С.А. Инютин, В.О. Иванов

Московский авиационный институт (национальный исследовательский университет)

Современные вычислительные задачи в области теории чисел часто требуют обработки числовых значений, значительно превышающих возможности стандартных компьютерных архитектур. Особую сложность представляют операции с числами специального вида и тестирование на простоту, где традиционные методы сталкиваются с фундаментальными ограничениями. Модулярная арифметика предлагает принципиально иной подход к решению этих задач, обеспечивая не только работу с числами произвольной величины, но и естественный параллелизм вычислений [1].

Предлагаемая модель основана на тщательном подборе параметров модулярной системы, включая выбор оптимальных оснований. Ключевым аспектом является минимизация бивалентного эффекта, который возникает при отображении модулярных вычетов в двоичных регистрах. Для этого предлагается использовать пары простых чисел, симметрично расположенных относительно степеней двойки (2^{31} , 2^{32} , 2^{64}). Такой подход позволяет максимально эффективно использовать стандартные 32- и 64-разрядные регистры современных процессоров [2, 3].

Для практической реализации модели были разработаны три специализированных алгоритма на языке Python. Особое внимание уделено оптимизации производительности: применение библиотеки Numba с компилятором LLVM позволило ускорить вычисления в 40–50 раз. Для работы с особо большими числами (2^{64}) была дополнительно использована высокопроизводительная программа primesieve, написанная на C++. Эти решения обеспечили эффективный поиск симметричных простых чисел даже в экстремально больших диапазонах. Алгоритмы дают одинаковый результат при их выполнении, что подтверждает их взаимную правильность.

Разработанная модель имеет значительный потенциал для применения в различных областях, включая криптографию, обработку сигналов и вычислительную теорию чисел. Особую ценность представляют возможности параллельной обработки данных и устойчивость к ошибкам, что делает модулярные вычисления перспективным направлением для создания высокопроизводительных вычислительных систем. Дальнейшие исследования могут быть направлены на адаптацию модели для работы с еще большими числовыми диапазонами и специализированными аппаратными архитектурами.

Литература

1. Инютин С.А. Модулярная алгоритмика многоразрядных вычислений. М.: Изд-во МАИ, 2020. 160 с.
2. Инютин С.А. Модулярные процессоры – оценки, история борьбы и победы над бивалентным дефектом // Развитие вычислительной техники в России и странах бывшего СССР: история и перспективы. SoRuCom-2017: сб. тр. IV Междунар. конф., Зеленоград, 3–5 октября 2017 / под ред. д.ф.-м.н. А.Н. Томилина. М.: РЭУ им. Г.В. Плеханова, 2017. С. 72–77.
3. Эрдниева Н.С. Использование специальных модулей системы остаточных классов для избыточного представления // Вестник АГТУ. Сер. Управление, вычислительная техника и информатика. 2013. № 2. С. 75–84.

Сравнительный анализ методов тензорной ренормализационной группы для моделирования двумерных конденсированных систем*

А.В. Карпова, С.С. Акименко

ФГАОУ ВО «Омский государственный технический университет»

Исследование двумерных конденсированных систем, учитывающих молекулярную структуру и сложные взаимодействия, является одной из ключевых задач современной науки о материалах. Это обусловлено потребностью в создании новых материалов с улучшенными электрическими, оптическими или каталитическими свойствами [1]. Ввиду экспериментальных трудностей и ограничений традиционных методов, применение новых подходов, в частности метода тензорной ренормализационной группы (ТРГ), относится к перспективным направлениям. Особый интерес представляет недавно предложенная авторами из Японии модификация метода ТРГ со взвешенными связями (ТРГВС) [2], где за счет учета влияния окружения системы обеспечивается сопоставимая точность результатов по сравнению с другими подходами при аналогичных вычислительных затратах. Однако оценка эффективности ТРГВС, представленная авторами, ограничена простыми спиновыми моделями, что не позволяет сделать однозначные выводы о ее применимости к более сложным адсорбционным системам.

Программная реализация разработана на языке программирования python 3.12.3. Распараллеливание расчетов осуществлялось с применением библиотек numpy 1.26.4 и scipy 1.13.0. Вычисления проводились на рабочей станции с процессором AMD Ryzen Threadripper 3970X и 256 Гб оперативной памяти.

Алгоритм ТРГВС верифицирован на примере модели Изинга на квадратной решетке при $\chi = 24$ (число оставляемых сингулярных значений). Полученные результаты согласуются с литературными данными [2]. Сравнение подходов ТРГ и ТРГВС проводилось на треугольной решетке на примере модели жестких гексагонов (модели моноцентриковой адсорбции с запретом на ближайшее соседство). Использование этой модели, имеющей аналитическое решение и некратную двум структуру, обеспечивает объективную оценку. Анализ эффективности распараллеливания разработанной реализации позволил установить, что оптимальное число потоков растет с увеличением параметра χ (рис. 1). При этом ТРГВС демонстрирует минимальную относительную ошибку и наименьшие вычислительные затраты при любом χ .

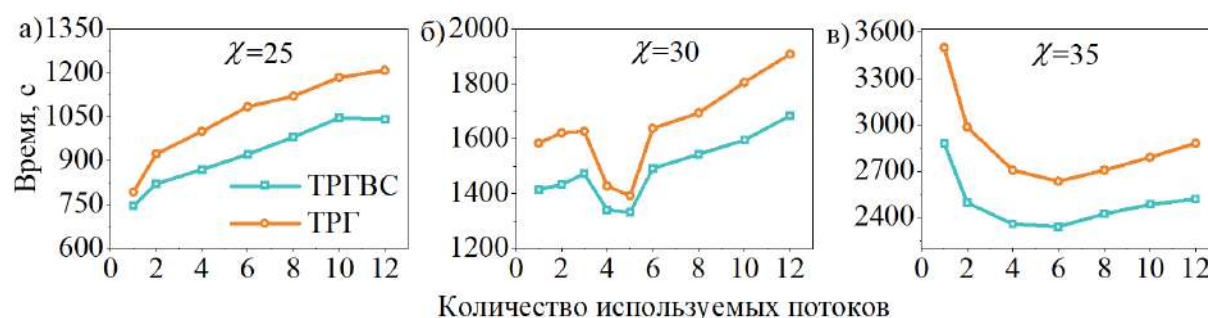


Рис. 1. Сравнение эффективности распараллеливания алгоритмов при различном параметре χ

Литература

1. Zhang L., Wang N., Li Y. Design, synthesis, and application of some two-dimensional materials // Chem. Sci. 2023. Vol. 14, no. 20. P. 5266–5290. DOI: 10.1039/D3SC00487B.
2. Adachi D., Okubo T., Todo S. Bond-weighted tensor renormalization group // Phys. Rev. B. 2022. Vol. 105, no. 6. P. L060402. DOI: 10.1103/PhysRevB.105.L060402.

* Исследование выполнено за счет гранта Российского научного фонда (проект № 22-71-10040).

Разработка средств анализа исполнения LuNA-программ на основе факторов SLOU*

С.Е. Киреев^{1,2}, Н.С. Ляпич²

Институт вычислительной математики и математической геофизики СО РАН¹,
Новосибирский государственный университет²

В работе рассматривается проблема анализа производительности и оптимизации параллельных программ на основе параллелизма задач. Предлагается количественно оценить степень влияния факторов, из которых складывается время работы программы: нехватка работы (Starvation), задержки на передаче данных (Latency), накладные расходы (Overhead) и полезная нагрузка (Useful work). Значения соответствующих характеристик позволят оценить свойства параллельных программ для нахождения их узких мест и дальнейшей оптимизации.

Целью работы является реализация алгоритма получения характеристик SLOU [1] для актуальной версии системы фрагментированного программирования LuNA [2]. Описание самого подхода и его реализации для более ранней версии системы LuNA см. в [1]. О технологии фрагментированного программирования, системе LuNA и разнице в ее версиях см. в [2].

В результате работы в новую версию системы LuNA был встроен компонент для сбора характеристик SLOU. Был проведен ряд синтетических тестов (аналогичных представленным в [1]), направленных на проверку правильности получаемых характеристик. Суть тестов заключается в том, чтобы исследовать характеристики SLOU для LuNA-программ с заранее известными свойствами. Тестирование показало, что при изменении параметров LuNA-программы получаемые значения характеристик SLOU изменяются ожидаемым образом (см. рис. 1). Следовательно, данная реализация алгоритма получения характеристик SLOU применима к анализу LuNA-программ.

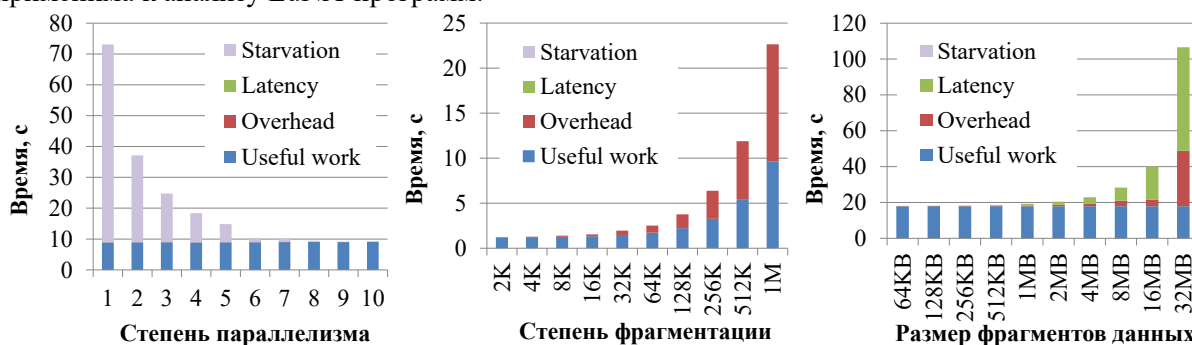


Рис. 1. Зависимость характеристик SLOU от параметров LuNA-программы

Представленный в [1] и реализованный в настоящей работе алгоритм вычисления характеристик SLOU учитывает только зависимости по данным. В ближайших планах – расширить алгоритм поддержкой зависимостей по управлению. Далее планируется разработать методику оптимизации фрагментированных программ на основе характеристик SLOU.

Литература

1. Киреев С.Е., Литвинов В.С. Анализ исполнения фрагментированных программ на основе факторов SLOW // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 2. С. 77–96. DOI: 10.14529/cmse240205.
2. Малышкин В.Э., Перепелкин В.А. Мультиагентный подход к повышению эффективности исполнения фрагментированных программ в системе LuNA // Проблемы информатики. 2023. № 3. С. 55–67. DOI: 10.24412/2073-0667-2023-3-55-67.

* Работа выполнена в рамках государственного задания ИВМиМГ СО РАН FWNM-2025-0005.

Полиэдральный кодизайн тайлинга циклов в MLIR

А.В. Левченко

Суперкомпьютерный центр СПбПУ

Актуальность работы обусловлена необходимостью разработки промежуточного языка планирования для тайлинга циклов. Тайлинг — это метод разделения вычислительных операций алгоритма на подмножества (тайлы) для реорганизации порядка вычислений, применяемый в целях оптимизации локальности и параллелизма программ в области HPC.

Научная новизна. В работе предложен метод совместного проектирования тайлинга и слияния циклов, называемый полиэдральным кодизайном (рис. 1). Метод решает проблему согласованности тайлинга и слияния циклов в рамках полиэдральной компиляции [1]. *Входными данными* метода являются циклы вычислительной программы на языке многоуровневого промежуточного представления (Multi-Level Intermediate Representation, MLIR) библиотеки LLVM, а также размеры тайлов, полученные посредством алгоритма Tile Size Selection (TSS). Совместное проектирование тайлинга и слияния циклов выполняется посредством полиэдральной компиляции с использованием диалекта MLIR/Affine. Далее реализуется описание трансформирующих операций на языке планирования (Tiling Intermediate Representation, TIR) на основе диалекта MLIR/Transform. Язык TIR применяется при метапрограммировании диалектов для тайлинга посредством диалекта MLIR/IRDLL. Наконец, полученные варианты трансформаций применяются к IR целевой программы. *Выходными данными* метода являются трансформированные циклы на языке MLIR.

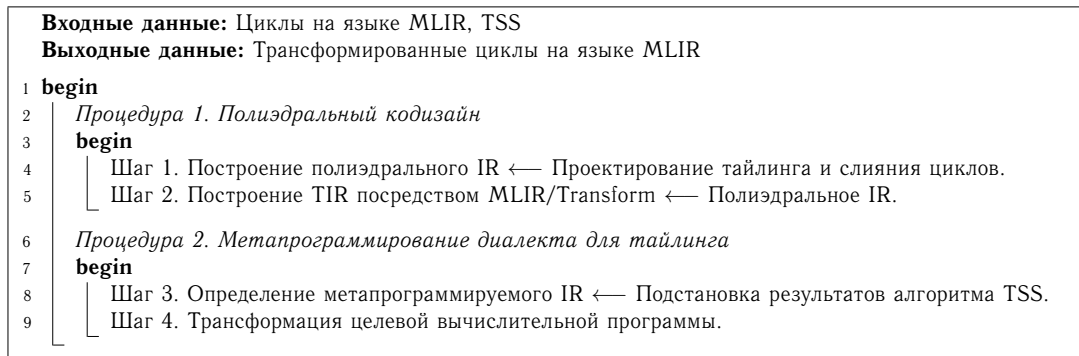


Рис. 1. Процедура полиэдрального кодизайна в конвейере тайлинга циклов MLIR

Экспериментальные результаты получены для полиэдральных версий тестовых программ класса трафаретных вычислений (stencils) FDTD-1D/2D, Jacobi-1D/2D и Seidel-2D при выполнении на системе с AMD EPYC 7763. Определен диалект для тайлинга циклов в IR используемых тестовых программ, позволяющий включить результаты алгоритма TSS в качестве параметров размеров тайлов для операций Transform IR. В результате полиэдральный кодизайн тайлинга и слияния циклов, реализованный посредством языка планирования MLIR/Transform с использованием значений TSS, позволил получить более чем двукратное ускорение тайлового кода относительно функции планирования для базового тайлинга.

Литература

1. Zhao J., Xu J., Di P. et al. Modeling the Interplay between Loop Tiling and Fusion in Optimizing Compilers Using Affine Relations // ACM Transactions on Computer Systems. January, 2024. Vol. 41, no. 1–4. Article 5. DOI: 10.1145/3635305

Реализация пакетирования с использованием параллельных вычислений в системе логического проектирования

И.П. Логинова

Объединенный институт проблем информатики НАН Беларуси

Важным этапом в автоматизированных системах проектирования цифровых интегральных схем является этап, в котором проводится синтез логических схем, реализующих функциональные комбинационные блоки в различных технологических базисах. Первый и особенно трудоемкий этап синтеза схемы представляет технологически независимая оптимизация, результат которой в дальнейшем и определяет такие характеристики логической схемы: площадь (число транзисторов), быстродействие, энергопотребление. Получение эффективных решений, в отдельных случаях, предлагает поиск такого решения для большого числа проектов, проведение большого числа экспериментов с разными методиками и алгоритмами оптимизации. Разбиение описания проекта на сеть объектов преследует цели: понижение размерности объекта оптимизации, получение оптимального решения для каждого элемента сети и, в итоге, получение оптимального решения задачи в целом. Расширение системы логической оптимизации функционально-структурных описаний цифровых устройств FLC-2 путем включения проектных процедур, которые выполняются с использованием параллельных вычислений [1], а также использование механизма продукций и стратегий [2], позволяет сократить число однотипных проектных действий, сократить время проектирования, повысить размерность решаемых задач синтеза. В основе подхода, реализующего такие проектные процедуры, лежит запуск параллельных процессов, в каждом из которых выполняется программа (например, проводится оптимизация) с отдельными частями проекта.

Следует отметить, что в системе FLC-2 [1, 2] на всех этапах проектирования проводятся преобразования только с одним проектом, который представлен описанием одной логической схемы, а также отсутствует возможность одновременного проведения проектных операций с несколькими описаниями схем. Поэтому появился запрос на разработку программной среды, которая позволяет проводить вычислительные эксперименты с разными проектами, с разными программами, представляющими разные методы оптимизации. Для реализации пакетирования была разработана и интегрирована с системой FLC-2 подсистема, которая позволяет объединять в пакет разные проекты, формирует скрипты для запуска параллельных процессов с указанием типа проектной процедуры, назначенной для каждого объекта в проекте, а затем запускает параллельное выполнение проектных процедур для всех объектов пакета. Возможности системы пакетирования позволяют: организовать автоматизированный подбор оптимальных значений атрибутов и параметров проектных операций; проводить сериализацию и формирование таблиц базы данных в формате SQLite [2] с информацией о проведенных экспериментах; проводить визуальный анализ результатов параллельных процессов, как в экспериментах с одним проектом, так и с проектами, собранными в пакет. В состав подсистемы пакетирования включено программное средство, которое визуализирует трассу выполнения параллельной программы [1], проводит оценку времени выполнения оптимизации в каждом параллельном процессе и оценку распределения нагрузки по ядрам процессора.

Литература

1. Логинова И.П. Построение и визуальный анализ параллельных решений в системе логической оптимизации функционально-структурных описаний дискретных схем // Компьютерное проектирование в электронике (EDA Conference 2024): сб. трудов Междунар. науч.-практ. конф., Республика Беларусь, Минск, 28 ноября 2024. Минск: БГУИР, 2024. С. 108–111.
2. Бибилло П.Н., Романов В.И. Логическое проектирование дискретных устройств с использованием продукционно-фреймовой модели представления знаний. Минск: Беларуская навука, 2011. 279 с.

Библиотека параллельных реализаций регуляризирующих алгоритмов решения линейных некорректно поставленных обратных задач

Д.В. Лукьяненко, В.Д. Шинкарев, С.С. Торбин, Д.О. Карпачев,
Р.А. Лоос, Н.А. Наволоцкий, Д.Н. Прямиков

МГУ имени М.В. Ломоносова, физический факультет, кафедра математики

При решении прикладных обратных задач многие исследователи, не специализирующиеся на разработке численных методов и их программной реализации, обычно используют готовые программные решения, которые доступны в широко известных прикладных пакетах программ и реализуют какие-либо известные методы, востребованные для решения рассматриваемых задач. При выборе таких программных решений исследователи чаще всего руководствуются популярностью соответствующих методов, предполагая, что популярный метод даст наилучший результат. Однако популярность многих методов зачастую связана со следующей особенностью. Исследователи обычно используют те методы, программная реализация которых доступна, а затем ссылаются на их применение в своих научных работах. Ссылки повышают известность этих методов и привлекают внимание других ученых. Возникает замкнутый круг, в котором вероятность выбора тех или иных методов зачастую определяется не их практическими возможностями, а известностью. В случае решения обратных задач, многие из которых относятся к классу некорректно поставленных, важной практической особенностью, определяющей качество решения таких задач, является возможность использования продвинутых регуляризирующих алгоритмов. При этом многие актуальные методы решения обратных задач, основанные на использовании достаточно эффективных регуляризирующих алгоритмов, остаются невостребованными, так как не имеют доступных высокопроизводительных программных реализаций, которые позволили бы задействовать все имеющиеся вычислительные ресурсы (все ядра многоядерных процессоров и/или графические ускорители, расположенные на доступных для расчетов вычислительных узлах).

Настоящее исследование направлено на частичное закрытие соответствующего пробела — на разработку эффективных программных реализаций параллельных версий некоторых актуальных регуляризирующих алгоритмов решения линейных обратных задач. Предполагается распараллелить некоторые регуляризирующие алгоритмы, которые зарекомендовали себя среди специалистов в области решения прикладных обратных задач, но которые не имеют общедоступных программных реализаций для использования на многопроцессорных системах.

Результатом исследования станут эффективные параллельные реализации актуальных регуляризирующих алгоритмов с открытым исходным кодом на одном из популярных языков научного программирования Python/C++/Fortran. В качестве основной технологии параллельного программирования будет использоваться технология параллельного программирования MPI, наиболее часто используемая для расчетов на вычислительных системах с распределенной памятью.

Анализ и визуализация данных в системе мониторинга эффективности задач HPC TaskMaster суперкомпьютерного комплекса

И.А. Мазаев, П.С. Костенецкий

Национальный исследовательский университет «Высшая школа экономики»

Данная работа посвящена созданию подсистемы визуализации данных о вычислительных задачах пользователей для системы HPC TaskMaster [1], разработанной Отделом суперкомпьютерного моделирования НИУ ВШЭ для суперкомпьютера sCHARISMa. Основная цель этой подсистемы – предоставить пользователям и администраторам удобный инструмент для исследования большого массива данных о задачах. В качестве платформы визуализации выбрана система Grafana, уже используемая для мониторинга нагрузки на кластер в TaskMaster. Ключевым элементом данной работы является разработка интерактивного трехмерного scatter plot для визуализации задач в пространстве их характеристик. Пользователь имеет возможность выбора параметров для каждой из осей X, Y, Z из списка доступных характеристик задачи, например: продолжительность выполнения задачи, среднее или максимальное использование CPU или GPU, использование диска. Предусмотрена возможность фильтрации отображаемых задач по конкретному пользователю или по временному интервалу.

На рис. 1 представлен один из вариантов визуализации статистического отчета о задачах пользователя, распределенных по времени. В этом примере по оси X время запуска задачи, по Y – количество GPU, по Z – среднее использование GPU, цвет – среднее использование CPU от фиолетового (0%) к желтому (100%). Рис. 1 позволяет провести анализ работы пользователя суперкомпьютера, выполняющего эксперименты по масштабированию параллельных программ. Пользователь изменяет количество GPU и находит компромисс между количеством GPU и утилизацией GPU.

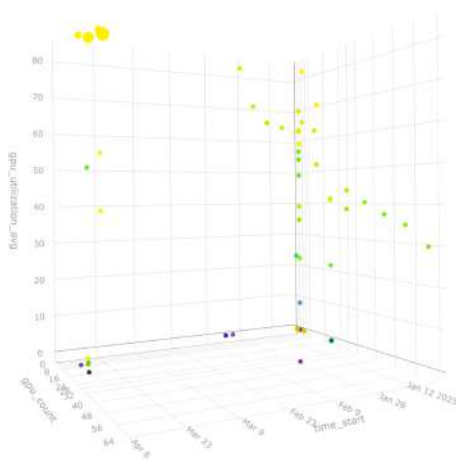


Рис. 1. Пример отчета о задачах пользователя суперкомпьютера

Литература

1. Kostenetskiy P., Shamsutdinov A., Chulkevich R. et al. HPC TaskMaster - Task Efficiency Monitoring System for the Supercomputer Center // Parallel Computational Technologies (PCT'2022). Vol. 1618. Springer, 2022. P. 17–29. Communications in Computer and Information Science. DOI: 10.1007/978-3-031-11623-0_2.

Численное решение двумерных параболических уравнений гемодинамики на многоядерных процессорах*

Э.Е. Малиев¹, Г.Ф. Громыко¹, Н.А. Лиходед², А.А. Толстиков²

Институт математики НАН Беларуси¹, Белорусский государственный университет²

Работа посвящена разработке и программной реализации на многоядерных процессорах алгоритмов численного моделирования течения крови в кровеносных сосудах. Рассматривается двумерная физико-математическая модель течения крови в сосуде, для дискретизации уравнений движения крови используется метод контрольных объемов. Вычисления по алгоритмам сводятся к решению блочно-трехдиагональных систем линейных алгебраических уравнений (СЛАУ) с вложенной структурой блоков с использованием варианта блочного метода Гаусса—Зейделя в неявной форме, обладающего естественным параллелизмом.

Течение крови в системе кровообращения в общем случае описывается трехмерными нестационарными уравнениями для вязкой жидкости совместно с уравнениями динамики эластичных оболочек сосудов. Это связано с необходимостью учета реальных свойств крови, пространственной геометрии сосудов, влияния вязкости, взаимного влияния гидродинамики в сосуде и их деформации. В этой работе рассматривается упрощенная, но по многим показателям адекватная, двумерная модель движения крови в прямом сосуде без дефектов при следующих допущениях: течение двумерное, ламинарное, жидкость несжимаемая, неньютоновская, стенки сосудов твердые. Предполагается, что область течения представляет собой прямоугольник с открытыми противоположными границами.

В основу алгоритма моделирования положены численные схемы, построенные с использованием метода контрольных объемов. Необходимо построить и численно решить СЛАУ для двух компонент скорости и давления. В работе [1] разработан алгоритм численного решения на каждом временном слое блочно-трехдиагональных СЛАУ для получения поперечной компоненты скорости; аналогично можно получить и алгоритм для случая продольной компоненты. В этой работе построен алгоритм численного решения на каждом временном слое блочно-трехдиагональных систем линейных алгебраических уравнений с вложенной структурой блоков для одновременного получения поперечной и продольной компонентов скорости. Кроме того, на каждом временном слое решается блочно-трехдиагональная СЛАУ для получения давления.

Алгоритмы программно реализованы с использованием технологии параллельного программирования OpenMP. Разработанные алгоритмы и программы могут быть использованы как задел для построения алгоритмов и программ для OpenMP- и CUDA-реализаций двумерных и трехмерных параболических уравнений гемодинамики. Результаты исследований позволят построить и численно реализовать более адекватную, по сравнению с известной [2], модель кровотока в условиях наличия дефектов в полости сосуда, например, тромбоза сосудов.

Литература

1. Баханович С.В., Громыко Г.Ф., Лиходед Н.А., Толстиков А.А. Параллельная реализация алгоритмов численного решения двумерных параболических уравнений гемодинамики // Параллельные вычислительные технологии (ПаВТ'2024): Труды международной научной конференции (Челябинск, 2–4 апреля 2024 г.). Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2024. С. 52–64.
2. Конюхов В.М., Храменков М.Г., Конюхов И.В. Фильтрационная модель и параллельные вычисления характеристик кровотока в системе кровообращения при наличии закупоривающего кольцевого тромба // Программные продукты и системы. 2022. Т. 35, № 3. С. 340–347. DOI: 10.15827/0236-235X.139.340-347.

* Работа выполнена в рамках государственной программы научных исследований Республики Беларусь «Конвергенция-2025», подпрограмма «Математические модели и методы».

Подход к параллельной реализации проверки бинарного отношения покрытия для заданной пары конечных языков

Б.Ф. Мельников¹, Линцянь Мэн²

Совместный университет МГУ-ППИ в Шэньчжэне¹,
Московский государственный университет им. М.В. Ломоносова²

Работа над рассматриваемой тематикой началась с публикации [1], где предлагалось разделение множества конечных формальных языков на специально определенные классы эквивалентности. Для исследования данной эквивалентности требуются алгоритмы для отношения покрытия. В статьях [2, 3] авторы предложили варианты автоматов на основе функции проверки отношения покрытия. При простом покрываемом языке небольшого размера сложность автомата PRI «в худшем» случае растет экспоненциально к длине входа. Нужно параллельный подход к реализации проверки отношения покрытия.

В данной работе рассмотрели параллельные подходы для реализации программы построения автомата PRI, используя технологию OpenMP и MPI. Конкретная реализация параллельного алгоритма равносильна разработке специальной эвристики выбора состояний, декомпозирующей связь между предыдущим и последующим состоянием в переходе.

Рассмотрели два основных подхода. Первый подход разработан для построения полного автомата PRI с известным множеством состояний, позволяющее равномерно распределить вычисление сбалансированной загрузки. Второй подход реализует построение сокращенного автомата тем путем, что в каждом раунде из суммированных результатов «главный» процесс выбирает новые состояния и распределяет нагрузку максимально равномерно.

Основные результаты экспериментов показывает, что нужно оптимизировать эвристики для декомпозиции задачи: первый подход теряет время из-за избыточности вычисления недостижимых состояний; второй подход крайне неэффективен для небольших автоматов из-за времени коммуникации между процессами. Одновременно важный открытый вопрос состоит в непредсказуемости достижимых состояний автомата по форме языков. При небольшом изменении буквы в языке количество состояний может колебаться от самых тривиальных двух (начальное и финальное состояния) до десятков. Эти основные подходы не лишены практической ценности.

Итак, рассматриваемая нами задача достаточно сложна – несмотря на простые ее условия. Можно сделать вывод о необходимости применения при создании алгоритмов решения этой задачи как различных эвристик, так и возможных вариантов распараллеливания. Применение связано с ранее описанными авторами гипотезами теории формальных языков, а также с построением различных алгоритмов автоматизации построения компиляторов.

Литература

1. Melnikov B. The equality condition for infinite catenations of two sets of finite words // International Journal of Foundations of Computer Science. 1993. Vol. 4, no. 3. P. 267–273. DOI: 10.1142/S0129054193000171.
2. Melnikov B., Meng L. Eight variants of finite automata for checking the fulfillment of the iteration coverage ratio of two finite languages. Part I // International Journal of Open Information Technologies. 2023. Vol. 11, no. 11. P. 1–9. (in Russian)
3. Melnikov B., Meng L. Eight variants of finite automata for checking the fulfillment of the iteration coverage ratio of two finite languages. Part II // International Journal of Open Information Technologies. 2024. Vol. 12, no. 2. P. 1–11. (in Russian)

Моделирование потока задач вычислительного кластера НИУ ВШЭ с использованием SLURM Simulator

Р.М. Мишенин, П.С. Костенецкий

Высшая школа экономики

Задача эффективного распределения ресурсов вычислительной системы широко известна. Она становится критически важной в многопользовательских многопроцессорных системах, таких как суперкомпьютеры. В НИУ ВШЭ функционирует высокопроизводительный вычислительный кластер «сHARISMA», состоящий из 48 вычислительных узлов шести типов. Основными характеристиками узлов являются наличие графических ускорителей и их модели, типы центральных процессоров и объем оперативной памяти. На суперкомпьютере используется планировщик задач SLURM версии 23.02.4.

При подборе параметров планировщика необходимо заранее определить, как они повлияют на поток задач вычислительного кластера. Одним из способов решения данной проблемы является моделирование вычислительного кластера с учетом его топологии и многообразия выполняемых задач. В настоящее время существует множество симуляторов высокопроизводительных кластеров, таких как AccaSim, GridSim, CloudSim, SimGrid. В результате сравнения было принято решение использовать SLURM Simulator [1].

Для проведения моделирования был подготовлен комплект конфигурационных файлов и трасс событий. В набор входят конфигурационные файлы SLURM, файл настроек менеджера учетных записей, файл, содержащий перечень аккаунтов и пользователей, зарегистрированных в системе управления заданиями. Каждое событие содержит временную метку постановки задания в очередь, количество запрошенных ресурсов, класс обслуживания и другие стандартные атрибуты задачи, необходимые для дальнейшего анализа и моделирования. В качестве источника данных для формирования файла с трассами событий был использован поток задач, выполненных за год на вычислительном кластере «сHARISMA».

Под модельным временем выполнения потока задач будем понимать время, необходимое для выполнения заданного набора задач на моделируемой конфигурации вычислительного комплекса. Задача сводится к минимизации модельного времени путем оптимизации конфигурации моделируемого суперкомпьютера. Ключевыми параметрами, существенно влияющими на работу алгоритма планирования, являются тип планировщика (builtin или backfill), а также весовые коэффициенты, варьирующиеся от 0 до 2 147 483 645 и определяющие приоритет задач. При помощи моделирования будет подобран оптимальный комплект вычислительного оборудования для апгрейда суперкомпьютера в 2026 г., который наиболее эффективно снизит время ожидания реальных вычислительных задач в очереди.

В качестве проверки модели было рассчитано влияние небольшого апгрейда, выполненного в 2024 г. В кластер были добавлены два вычислительные узла на базе GPU Nvidia H100. Среднее модельное время ожидания задач в очереди уменьшилось на 2.1 % относительно базовой конфигурации, что подтвердилось реальными статистическими значениями, полученных на суперкомпьютере – 1.95 %.

Дальнейшими задачами исследований является подстройка алгоритмов планировщика задач на моделируемой копии суперкомпьютера «сHARISMA». В результате моделирования будут определены изменения в параметрах конфигурации SLURM, внедрение которых в планировщик реального вычислительного кластера позволит снизить среднее время ожидания задач в очереди и дополнительно повысить эффективность использования суперкомпьютера.

Литература

1. Simakov N. et al. Slurm Simulator: Implementation and Parametric Analysis // High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation. PMBS 2017. Vol. 10724 / eds. by S. Jarvis, S. Wright, S. Hammond. Springer, Cham, 2017. P. 197–217. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-72971-8_10.

Молекулярно-динамическое моделирование поведения воды в условиях ограниченной геометрии кремниевых нанопор*

Е.Г. Одинцова^{1,3}, Д.Л. Гурина^{1,3}, Ю.А. Будков^{1,2,3}

Лаборатория вычислительной физики НИУ ВШЭ¹,

Департамент прикладной математики НИУ ВШЭ²,

НИО «Мультимасштабное моделирование молекулярных жидкостей и растворов»

Института химии растворов им. Г.А. Крестова РАН³

Понимание влияния межфазных эффектов на перенос жидкости через нанопоры имеет огромное значение для различных областей науки и техники, таких как проектирование нанофлюидных устройств, процессы адсорбции и разделения, а также прогнозирование и управление переносом жидкости. Одним из ключевых аспектов является изучение структурных и транспортных свойств воды в нанопорах, что стало возможным благодаря использованию методов компьютерного моделирования.

Целью данного исследования является анализ влияния ширины щелевидных кремниевых пор и функционализации их поверхности на структуру и динамику воды в условиях ограниченного пространства. С использованием метода полноатомного молекулярно-динамического моделирования исследовано поведение воды в кремниевых порах шириной 0.6–3.0 нм с негидроксильной и полностью гидроксильной поверхностью. Моделирование выполнено для поверхностей β -кristобалита (111), воспроизводящих свойства аморфного кремнезема с характерными силанольными группами (Si-OH и Si-(OH)₂).

Моделирование проводилось с использованием вычислительных ресурсов суперкомпьютера sCHARISMa НИУ ВШЭ в программном пакете Gromacs 2022.2. Использование гибридного параллелизма (MPI + CUDA) для эффективного распределения вычислений между CPU и GPU, позволило ускорить расчеты в десятки раз по сравнению с чисто CPU-реализацией. Такой подход открывает новые возможности для моделирования жидкостей в нанопорах, позволяя проводить расчеты с ранее недоступной точностью и скоростью. Молекулы воды описывались моделью SPC/E. Моделирование исследуемых систем выполнялось при температуре 298 К в NVT-ансамбле (термостат Нозе-Гувера, $\tau = 0.1$ пс) с периодическими граничными условиями во всех трех направлениях.

Для исследования структурных и динамических свойств воды в нанопорах были проведены расчеты распределения плотности, функций радиального распределения, среднего числа и времени жизни водородных связей, ориентационной упорядоченности молекул воды (угол между вектором дипольного момента и осью Z, перпендикулярной стенкам поры), транспортных характеристик (среднеквадратичные смещения в параллельном и перпендикулярном направлениях относительно поверхности поры).

В гидроксильрованных порах шириной 0.6–1.0 нм вода формирует высокоупорядоченные слои с аномально высокой локальной плотностью (до 1.5 г/см³). При этом подвижность молекул вдоль поверхности значительно выше, чем в негидроксильрованных порах. С увеличением ширины пор (1.5–3.0 нм) в гидроксильрованных системах сохраняется четко выраженное слоистое распределение плотности, тогда как в негидроксильрованных порах плотность изменяется плавно. Гидроксильрование поверхности приводит к улучшению смачиваемости за счет образования прочных водородных связей между молекулами воды и поверхностными силанольными группами, а также к формированию упорядоченных водных слоев с аномальными транспортными свойствами (в отличие от хаотичной адсорбции в негидроксильрованных порах). Кроме того, наблюдается усиление анизотропии диффузии: молекулы воды демонстрируют более высокую подвижность вдоль поверхности поры.

Таким образом, гидроксильрование может являться одним из инструментов для управления структурой и динамикой воды в наноразмерных порах, что имеет важное значение для современных нанотехнологий и материаловедения.

* Работа выполнена при финансовой поддержке Российского научного фонда (грант № 24-11-00096).

Архитектура среды моделирования когнитивных процессов

П.А. Панилов, М.А. Комардин, Р.М. Бритвин

МГТУ им. Н.Э. Баумана

Данная работа описывает разработку среды моделирования когнитивных процессов, основанной на концепции атрибутов, миров и состояний [1, 2]. Основная идея заключается в представлении каждого объекта в системе как набора динамически обновляемых атрибутов, унаследованных от родительского мира. Такая архитектура позволяет моделировать процессы обучения, принятия решений и адаптации в режиме реального времени.

В предложенной модели объекты группируются в «миры», определяющие их базовые свойства, а динамика изменений описывается через вложенные состояния (рис. 1).

Основными механизмами являются:

Наследование атрибутов. Каждый объект получает характеристики от своего родительского мира.

Динамическое обновление. Атрибуты корректируются в процессе взаимодействия между объектами.

Графовая модель связей. Использование графовых баз данных обеспечивает эффективное хранение и анализ взаимосвязей.

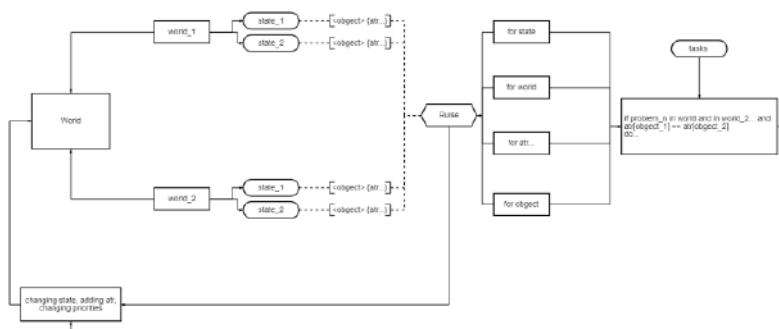


Рис. 1. Схема работы алгоритма

Математический аппарат включает методы теории множеств, гауссовское распределение приоритетов, байесовское обновление и Марковские процессы для оптимизации принятия решений. Эти методы позволяют корректировать распределение действий в зависимости от отклонений между ожидаемым и фактическим результатом.

В качестве демонстрации возможностей системы рассматривается моделирование образовательного процесса. Здесь объекты (ученики, школы, транспорт и социальное окружение) характеризуются набором атрибутов, отражающих как внутренние, так и внешние факторы. Механизм наследования и динамического обновления позволяет прогнозировать изменения в образовательных показателях и оптимизировать распределение ресурсов.

Новизна подхода заключается в интеграции динамически обновляемых атрибутов с графовыми структурами, что обеспечивает высокую адаптивность системы и приближение к реальным когнитивным механизмам

Литература

1. Дивеев А.И. Машинное обучение систем интеллектуального управления // Интеллектуальные системы. Теория и приложения. 2022. Т. 26, № 1. С. 35–43.
2. Суханов В.А., Цибизова Т.Ю. Архитектура системы развития и поддержки профессиональной интеллектуальной деятельности // Динамика сложных систем – XXI век. 2020. Т. 14, № 1. С. 23–31. DOI: 10.18127/j19997493-202001-02.

Метод автоматического вывода алгоритмов эффективного решения задач на вычислительных моделях с шаблонами

Д.Р. Парфенов

Новосибирский государственный университет

В работе рассматривается проблема автоматического конструирования применимых на практике параллельных программ. Программа конструируется по описанию предметной области в виде вычислительной модели и известным наборам входных и выходных переменных [1, 2]. В результате планирования выводится план – упорядоченный по информационным зависимостям набор операций, которые необходимо выполнить для решения прикладной задачи.

Задача выбора оптимального набора операций для построения наиболее эффективного плана в общем случае является переборной. Однако для того чтобы автоматически сконструированная по плану программа была практически применимой, ей как правило не обязательно быть оптимальной. В данной работе предлагается метод для конструирования таких достаточно эффективных прикладных программ, которые не всегда являются оптимальными.

Первой частью метода является автоматическая оценка нефункциональных свойств генерируемых программ по соответствующим свойствам используемых в них операций. Вторая часть метода – дополнение вычислительных моделей заготовками эффективных алгоритмов – шаблонами. Шаблоны ставят в соответствие множеству параметров планы для решения подзадач и позволяют связать перебор планов с перебором множества значений некоторых параметров прикладного алгоритма. Например, в прикладной задаче вычисления 50 автокорреляционных сверток сейсмических трасс [3] в качестве параметра использовалось количество трасс, которые будут обрабатываться на GPU. При этом остаток распределялся для исполнения на CPU.

Шаблоны позволяют перейти от перебора всех возможных планов только к тем, которые представляют интерес для решения прикладной задачи, а также делают возможным применение существующих методов решения оптимизационных задач, например градиентные методы, к выводу планов на вычислительных моделях.

Литература

1. Вальковский В.А., Малышкин В.Э. Синтез параллельных программ и систем на вычислительных моделях. Новосибирск: Наука, 1988. 129 с.
2. Malyshkin V. Active Knowledge, LuNA and Literacy for Oncoming Centuries // Programming Languages with Applications to Biology and Security. Vol. 9465. Springer, Cham, 2015. P. 292–303. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-25527-9_19.
3. Хайретдинов М.С., Воскобойникова Г.М., Седухина Г.С. Алгоритмы поточной свертки в задачах активного вибросейсмоакустического мониторинга // Геосибирь. 2017. Т. 1. С. 230–234.

Разработка клиент-серверной системы управления пользовательскими данными в рамках проекта Algo500

А.Г. Пионткевич, А.С. Антонов, Р.В. Майер

НИВЦ МГУ имени М.В. Ломоносова

Проект Algo500 [1, 2] направлен на создание масштабируемой цифровой платформы для совместного анализа свойств алгоритмов и особенностей архитектур суперкомпьютеров и предполагает решение проблемы оценки эффективности алгоритмов в контексте их применения на различных архитектурах суперкомпьютеров. Платформа Algo500 предназначена для того, чтобы исследовать и анализировать свойства реализации любого алгоритма в зависимости от характеристик конкретной вычислительной системы, что является важным для оптимизации вычислительных процессов и повышения производительности [3]. Одной из ключевых компонент платформы Algo500 является PerfData, предназначенная для эффективного управления пользовательскими данными, получаемыми в ходе выполнения суперкомпьютерных экспериментов [3, 4].

С увеличением объема файлов, представляющих программы и данные, относящиеся к условиям суперкомпьютерных экспериментов, возникает необходимость в создании удобного и эффективного клиент-серверного приложения, которое обеспечит быструю и надежную загрузку пользовательских файлов на удаленный сервер. Целью данного исследования является разработка подсистемы хранения файлов программ и данных, входящих в условия суперкомпьютерного эксперимента, интегрированной в компоненту PerfData проекта Algo500.

Разработанное нами приложение

- 1) обеспечивает безопасную загрузку на удаленный сервер по сети Internet файлов размером до нескольких десятков гигабайт и возможность возобновления загрузки файлов с места остановки для минимизации риска потери данных при разрыве соединения;
- 2) поддерживает авторизацию пользователей и разграничение прав пользователей при файловых операциях;
- 3) имеет интуитивно понятный web-интерфейс;
- 4) предусматривает интеграцию с проектом Algo500;
- 5) выполняется на всех современных web-браузерах;
- 6) использует открытые, бесплатные библиотеки.

Клиентская часть содержит интерфейс с реализованными функциями пользователя, при обработке которых происходит обращение к серверной части. Серверная часть, написанная в среде Node.js, содержит микросервис больших файлов, использующий библиотеку Resumable.js, и микросервис директорий. Данные пользователя для авторизации и необходимая информация о суперкомпьютерных экспериментах поступают из базы данных Algo500.

Литература

1. Antonov A.S., Nikitenko D.A., Voevodin V.V. Algo500 - A New Approach to the Joint Analysis of Algorithms and Computers // Lobachevskii J Math. 2020. Vol. 41, no. 6. P. 1435–1443. DOI: 10.1134/S1995080220080041.
2. Antonov A.S., Maier R.V. Development and Implementation of the Algo500 Scalable Digital Platform Architecture // Lobachevskii J Math. 2022. Vol. 43, no. 7. P. 837–847. DOI: 10.1134/S1995080222070058.
3. Антонов А.С., Майер Р.В. Онтологический анализ предметной области цифровой платформы Algo500 // Вычислительные методы и программирование. 2023. Т. 24, № 1. С. 89–114. DOI: 10.26089/NumMet.v24r107.
4. Antonov A.S., Maier R.V., Nikitenko D.A. Representation of Experimental Data in the Algo500 Project // Lobachevskii J Math. 2023. Vol. 44, no. 7. P. 3005–3015. DOI: 10.1134/S199508022308005X.

Разработка подсистемы динамического изменения приоритета задач в очереди суперкомпьютера

Г.С. Рыбаков, П.С. Костенецкий

НИУ ВШЭ

С ростом числа задач в различных областях науки, требующих высокопроизводительных вычислений, правильное распределение приоритетов задач и квот пользователей стало одним из важнейших факторов, влияющих на эффективную работу суперкомпьютерных центров. На суперкомпьютере НИУ ВШЭ «сHARISMa» настроено множество дополняющих друг друга алгоритмов планирования потока задач, а также ограничений, таких как проектные лимиты и пользовательские квоты. Алгоритмы и ограничения не позволяют одному или нескольким пользователям полностью занять ресурсы суперкомпьютера и таким образом парализовать вычислительные эксперименты других пользователей университета. Кроме того, для поиска и устранения некорректно запущенных и не выполняющих вычислений задач, в НИУ ВШЭ разработана система мониторинга эффективности HPC TaskMaster.

Набор перечисленных программных средств позволяет обеспечить комфортную работу на суперкомпьютере НИУ ВШЭ более 1500 пользователей. Суперкомпьютер работает на базе системы очередей SLURM. Так как пользователей довольно много, в очереди всегда есть задачи, ожидающие запуска. Например, когда один пользователь ставит в очередь сто вычислительных задач, первые из них запускаются немедленно, а остальные получают пониженный приоритет. Такой подход позволяет не блокировать работу остальных пользователей суперкомпьютера.

В то же время, существуют внешние факторы, влияющие на поток задач, например, наступление выходных или праздничных дней. В период пониженной активности пользователей можно смягчать ограничения для задач, ожидающих запуска. При этом можно добиться, чтобы большее, чем обычно, количество ресурсов, занятых несколькими активными в нерабочий день пользователями, не мешало остальным. Ручное изменение приоритетов задач, а также квот пользователей, позволяет подстраивать поток задач под внешние условия, но часто это сопряжено с задержками, связанными с человеческим фактором. Данная статья посвящена разработке подсистемы динамического изменения приоритетов задач и пользовательских квот, позволяющей более эффективно распоряжаться ресурсами суперкомпьютера.

Путем анализа графика загрузки суперкомпьютера за 4 года, определены факторы, влияющие на колебание загрузки, а также их периодичность и продолжительность. В частности, этим фактором является тип дней в неделе: будние и выходные. Таким образом, средняя разница между этими двумя типами дней составляет 6.2%. Исходя из этого фактора, можно определять, насколько можно изменить квоты пользователей для максимальной утилизации ресурсов суперкомпьютера. Для этого была предложена архитектура подсистемы TaskShift, состоящая из трех модулей. Первый модуль агрегирует данные. Второй – модуль прогнозирования временных рядов, который предсказывает загруженность CPU и GPU с помощью фреймворка NeuralProphet. Третий модуль – это планировщик, который на основе предсказанной средней загруженности и текущего статуса суперкомпьютера, включающего загруженность и очередь задач, принимает решение о запуске той или иной задачи в обход планировщика SLURM.

Следующим этапом является разработка системы, состоящей из этих трех компонентов, а также добавление в модуль планирования симулятора SLURM для проверки эффективности стратегии перед запуском задач. Определена сезонность загруженности суперкомпьютера, проанализирована зависимость выходных от будних, предложен подход по предсказанию загруженности и анализу задач.

Расчет вольт-амперных характеристик для упрощенной модели варистора*

К.К. Сабельфельд¹, Д.С. Абрамкин², С.Е. Киреев¹, А.Е. Киреева¹

Институт вычислительной математики и математической геофизики СО РАН¹,

Институт физики полупроводников им. А.В. Ржанова СО РАН²

Варистором называется нелинейный резистор, который используется для стабилизации и защиты электронного оборудования от перенапряжений. Его отличительной чертой является нелинейная вольт-амперная характеристика (ВАХ). Варистор состоит из высокопроводящих гранул (например, Ni, покрытых тонким слоем Al_2O_3), на стыке которых возникает эффект нелинейной проводимости. Исследование связи между электрическими характеристиками варистора и его микроструктурой является актуальной задачей, в частности, для усовершенствования технологии их производства. Целью работы является разработка и реализация алгоритма вычисления динамической ВАХ варистора с учетом его трехмерной микроструктуры, процессов туннелирования электронов и токов утечки. В настоящий момент выполнен первый этап работы: разработан и реализован алгоритм вычисления статической ВАХ для двумерной модели варистора. Варистор представлен в виде массива плотно упакованных шариков (гранул) одинакового радиуса. Заданная микроструктура варистора определяет его электрическую схему. К электродам варистора прикладывается напряжение U_{tot} , и вычисляется протекающий через него ток I_{tot} . Зависимость тока I от напряжения U для контакта между двумя гранулами описывается степенным законом: $I(U) = k \cdot U^\alpha$ [1]. Алгоритм вычисления статической ВАХ следующий. 1) С использованием законов Кирхгофа для электрической схемы варистора составляется система нелинейных уравнений для I , I_{tot} и U . 2) Система решается для заданного значения U_{tot} с помощью метода Ньютона. 3) Шаг 2 повторяется для необходимого множества значений U_{tot} . При реализации метода Ньютона для решения СЛАУ применяется функция LAPACK_dgelsd. Задача расчета ВАХ является трудоемкой, так как нелинейная система решается для многих значений U_{tot} , и для каждого U_{tot} требуется несколько раз решить СЛАУ. Например, расчет ВАХ варистора 21×21 гранул (размер СЛАУ 1254×1253) для $U_{\text{tot}} \in [0.1; 30]$ Вольт занимает 17 ч. Следовательно, необходима параллельная реализация алгоритма, которая выполняется путем распределения между MPI-процессами вычислений шага 2 для различных значений U_{tot} . Для расчетов использовался кластер НКС-1П, ЦКП «Сибирский суперкомпьютерный центр СО РАН», узел с Intel Xeon Phi 7290 (KNL). На рис. 1 а представлена эффективность распараллеливания программы для $k = 0.1$, $\alpha = 25$, $U_{\text{tot}} \in [0.1; 3000]$ Вольт для варистора 7×7 гранул (размер СЛАУ 127×126). Результаты моделирования (рис. 1 б, в) соответствуют физическим данным: степенному закону $I(U)$ и заданной электрической схеме варистора.

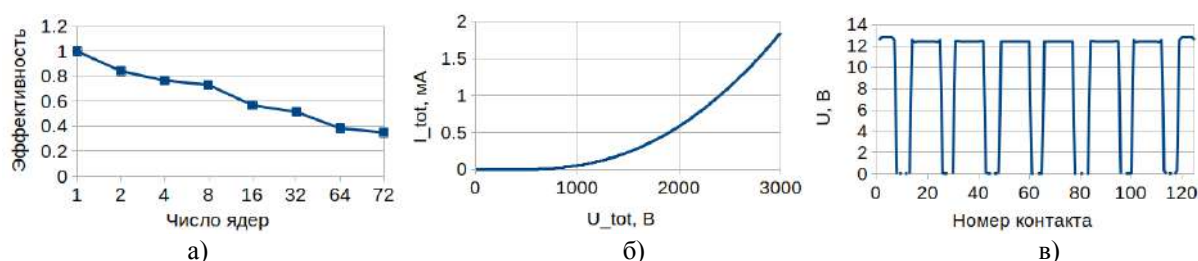


Рис. 1. Эффективность распараллеливания (а), ВАХ варистора (б), значения U на контактах (в)

Литература

1. Weimer M.A., Hakim L.F., King D.M. et. al. Ultrafast metal-insulator varistors based on tunable Al_2O_3 tunnel junctions // Applied Physics Letters. 2008. Vol. 92. P. 164101–164103. DOI: 10.1063/1.2913763.

* Работа выполнена при финансовой поддержке гранта Российского научного фонда № 24-11-00107.

Интеллектуальное прогнозирование времени выполнения вычислительных задач GROMACS для оптимизации работы планировщика задач суперкомпьютера "сHARISMa" НИУ ВШЭ

М.Т. Саликова, П.С. Костенецкий, Р.М. Мишенин
НИУ ВШЭ

Современные суперкомпьютеры являются незаменимыми инструментами для решения сложных научных и промышленных задач. Эффективное планирование потока вычислительных задач суперкомпьютера требует информации о времени, за которое может завершиться каждая из ожидающих запуска задач. Нередко пользователи устанавливают задаче временной лимит с большим запасом, что снижает эффективность работы планировщика. Разработка метода предсказания времени выполнения задач Gromacs на суперкомпьютере позволит более точно выстраивать поток задач, оптимально загружая вычислительные ресурсы суперкомпьютера.

Известные подходы предсказания длительности выполнения вычислительных задач обычно основываются на обобщенных метриках [1, 2], таких как число задействованных ядер и объем памяти, но не всегда учитывают специфические особенности отдельных приложений. Например, на время выполнения задач молекулярной динамики в пакете Gromacs влияют такие параметры, как количество атомов, временной шаг и общее количество шагов моделирования. Чем сложнее структура и выше плотность сетки, тем больше вычислительных ресурсов требуется, что влияет на общую длительность расчета.

Для осуществления исследования потребовалось агрегировать данные о задачах молекулярной динамики Gromacs, выполненных на суперкомпьютере *сHARISMa*. Датасет включает более двух тысяч записей, каждая из которых содержит такие параметры, как количество атомов, временной шаг, общее число шагов моделирования, число задействованных ядер, объем используемой памяти и фактическое время выполнения задачи. Для того чтобы увеличить пригодность для использования в прогнозных моделях, были учтены только данные о задачах, помеченных системой анализа *HPC TaskMaster* [3] как эффективные. Перед анализом датасет был подвергнут предварительной обработке, в том числе очистке от аномалий и нормализации, что позволило минимизировать влияние шумов и обеспечить корректность дальнейших вычислений.

На текущем этапе работы разработан алгоритм для статистического анализа, использующий ключевые параметры задач Gromacs для формирования прогноза времени выполнения. Следующей целью исследования является внедрение модели машинного обучения для повышения точности предсказания. После завершения этапа обучения модель будет протестирована и оценена ее эффективность в реальных условиях. В дальнейшем модель будет интегрирована в систему *HPC TaskMaster* для оптимизации потока задач суперкомпьютера.

Литература

1. Guo J. et al. Machine Learning Predictions for Underestimation of Job Runtime on HPC System // Supercomputing Frontiers, SCFA'18. Vol. 10776. Springer, 2018. P. 179–198. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-69953-0_11.
2. Gaussier E. et al. Improving backfilling by using machine learning to predict running times // Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2015. Article 64. P. 1–10. DOI: 10.1145/2807591.2807646.
3. Kostenetskiy P. et al. Enhancement of the Data Analysis Subsystem in the Task-Efficiency Monitoring System HPC TaskMaster for the сHARISMa Supercomputer Complex at HSE University // Parallel Computational Technologies, PCT'2024. Vol. 2241. Springer, 2024. P. 49–64. Communications in Computer and Information Science. DOI: 10.1007/978-3-031-73372-7_4.

Моделирование работы суперкомпьютерного кластера для оценки эффективности интеллектуальных алгоритмов в задачах диспетчеризации

В.Н. Сеннов, А.А. Лукашин, Е.П. Петухов

Санкт-Петербургский политехнический университет Петра Великого

Повышение эффективности использования суперкомпьютерных центров (СКЦ) является крайне важной задачей, особенно в условиях растущего спроса на высокопроизводительные вычисления и дефицита суперкомпьютерных ресурсов.

За последние годы большое количество работ было посвящено моделям оценки времени выполнения задач в СКЦ, в том числе с применением методов машинного обучения [1, 2]. Проведенные исследования показали, что пользователи суперкомпьютерных ресурсов, как правило, не используют параметр времени исполнения задачи или устанавливают его с большим запасом [3]. Данный параметр используется планировщиком ресурсов суперкомпьютерного кластера для постановки задач в очередь и формирование расписания.

Таким образом, разработка моделей для улучшения пользовательской оценки времени выполнения является важной и актуальной задачей. Однако метрики качества (обычно, средняя абсолютная или относительная ошибка), используемые при исследовании получившейся модели оценки не отвечают на вопрос, насколько эффективнее станет работать алгоритм диспетчеризации при использовании исследуемой модели.

Эффективность работы алгоритмов диспетчеризации оценивается при помощи таких метрик, как среднее количество простаивающих узлов или среднее время ожидания задачи в очереди. Исследование эффективности работы алгоритмов оценки времени исполнения задач и поиска наилучших настроек планировщика проводить на работающем кластере крайне сложно, поэтому необходимо иметь возможность провести моделирование работы суперкомпьютерного кластера с выбранным алгоритмом оценки времени и собрать метрики.

В данной работе предложен метод симуляции работы виртуального кластера на основе виртуализации служебных и вычислительных узлов кластера, а также показаны результаты симуляции на данных СКЦ «Политехнический».

Исследование выполнено при финансовой поддержке Минобрнауки России в рамках научно-исследовательского проекта «Разработка и исследование моделей машинного обучения для решения фундаментальных задач искусственного интеллекта в топливно-энергетическом комплексе» (FSEG-2024-0027). Результаты исследования получены с использованием вычислительных ресурсов суперкомпьютерного центра «Политехнический» (<https://scc.spbstu.ru>).

Литература

1. Баранов А.В., Николаев Д.С. Применение машинного обучения для прогнозирования времени выполнения суперкомпьютерных заданий // Программные продукты и системы. 2020. Т. 33, № 2. С. 218–220. DOI: 10.15827/0236-235X.130.218-228.
2. Muliukha V.A., Vostrov A.V., Motorin D.E. et al. Supercomputer resources management using machine learning methods under constraints // Computing, Telecommunications and Control. 2024. Vol. 17, no. 3. P. 32–41. DOI: 10.18721/JCSTCS.17303.
3. Malov S.V., Lukashin A.A. Count time series analysis of jobs scheduling in the hybrid supercomputer center // Computing, Telecommunications and Control. 2024. Vol. 17, no. 3. P. 42–53. DOI: 10.18721/JCSTCS.17304.

Применение принципов визуальных интерфейсов в больших языковых моделях как альтернатива текстовому взаимодействию

Е.А. Трофимова¹, Р.С. Савчук²

ВШЭ, Научно-учебная лаборатория методов анализа больших данных¹,
ВШЭ ФКН²

На данный момент, большая часть взаимодействия между пользователем и моделью происходит при помощи текстового интерфейса. Данный вид взаимодействия с LLM является наиболее интуитивным как и с точки зрения архитектуры модели, так и с точки зрения использования. В то же время, неформатированный вывод в текстовом формате является неудобным для восприятия комплексной информации. Язык разметки *Markdown*, использующийся повсеместно в чат-ботах, позволил минимально форматировать текст. Несмотря на успешный опыт внедрения методов форматирования текста, взаимодействие с LLM не получило существенного развития. В *SearchGPT* [1] были представлены виджеты, встроенные в текст, но наполнение данных виджетов генерируется без участия LLM. В рамках исследования предлагается изучить возможные подходы к взаимодействию с LLM, выявить их ограничения и предложить более интуитивные методы визуального взаимодействия.

Такие языки разметки как *Markdown* являются простыми для генерации, так как не имеют вложенности и поддерживают лишь примитивные *теги*, такие как заголовки и списки. Генерация более сложных языков разметки долгое время являлась затруднительной из-за множества ошибок в выводе, которые не решались увеличением числа параметров модели. Данная проблема была решена [2] путем ограничения возможных токенов в выводе, причем метод Grammar-Constrained Decoding применим для любых *бесконтекстных грамматик*. Данное решение было усовершенствовано [3] и добавлено в исходный код LLM пакетов, таких как *Llama.cpp* [4].

Предложенный подход: представление диалогов с пользователем в виде двунаправленного графа: вершины представляют собой логические блоки информации, а ребра – гиперссылки. Каждая вершина содержит набор виджетов наподобие HTML. Таким образом, данный граф возможно конвертировать в произвольный язык разметки. Для того, чтобы найти наилучший язык разметки для конкретной LLM, был разработан инструментарий для генерации и оценки бесконтекстных грамматик, в том числе SR(1) парсер-генератора.

Литература

1. OpenAI, Introducing ChatGPT Search.
URL: <https://openai.com/index/introducing-chatgpt-search/>.
2. Geng S., Josifoski M., Peyrard M., West R. Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning // Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023. Association for Computational Linguistics, 2023. P. 10932–10952.
DOI: 10.18653/V1/2023.EMNLP-MAIN.674.
3. Shorten C., Pierse C., Smith T.B. et al. StructuredRAG: JSON Response Formatting with Large Language Models // CoRR. 2024. Vol. abs/2408.11061. arXiv: 2408.11061. URL: <https://arxiv.org/abs/2408.11061>.
4. llama.cpp: LLM inference in C/C++. URL: <https://github.com/ggerganov/llama.cpp>.

Параллельный алгоритм послойного обучения нейронных сетей*

И.В. Шарун, А.В. Зыкина

Омский государственный технический университет

В настоящее время подходы к ускорению обучения глубоких нейронных сетей строятся на основе параллелизма данных, послойного последовательного обучения [1], либо на основе вертикального (конвейерного) параллелизма [2]. Данные подходы не лишены недостатков: либо происходит простой оборудования, либо возникает сложная система коммуникации между объектами оптимизации.

В рамках данного исследования рассматривается подход к ускорению обучения глубоких нейронных сетей за счет применения параллельного алгоритма послойного обучения. Суть данного метода заключается в декомпозиции общей задачи обучения сети на множество отдельных оптимизационных задач для каждого слоя. Каждый слой рассматривается как самостоятельный объект оптимизации, что позволяет организовать его обучение независимо от других слоев. После завершения шага оптимизации каждого слоя происходит агрегация полученных параметров, формируется общий набор весов, который используется в качестве начальной точки для следующей итерации.

Такая организация процесса обучения обеспечивает возможность использования параллельных и распределенных вычислений для одновременной работы над отдельными слоями или группами слоев, что значительно сокращает время обучения. Подход позволяет сократить время обучения, особенно для сетей с большим количеством слоев и параметров.

Для проверки эффективности подхода были проведены эксперименты с различными архитектурами нейронных сетей прямого распространения, включая сверточные нейронные сети. В ходе экспериментов использовались известные наборы данных, в том числе MNIST, CIFAR-10 и ImageNet, произведена оценка применимости метода для широкого спектра задач, связанных с классификацией изображений.

Результаты экспериментов подтвердили гипотезу о том, что параллельный алгоритм послойного обучения способен не только значительно сократить время обучения, но и сохранить уровень точности модели, сравнимый с результатами, полученными при использовании стандартных методов. При этом наблюдается связь между увеличением размера архитектуры сети и масштабируемостью предложенного подхода: чем больше слоев и параметров содержит сеть, тем более выраженным становится эффект ускорения.

Дальнейшие исследования направлены на анализ применимости метода для других видов нейронных сетей, в т.ч. рекуррентных, улучшение методов агрегации параметров для достижения еще большего ускорения без потери качества модели.

Литература

1. Karimi A., Kalhor A., Sadeghi Tabrizi M. Forward layer-wise learning of convolutional neural networks through separation index maximizing // Scientific Reports. 2024. Vol. 14. Article 8576. DOI: 10.1038/s41598-024-59176-3.
2. Núñez G., Romero-Sandí H., Rojas E., Meneses E. A Study of Pipeline Parallelism in Deep Neural Networks // Revista Colombiana de Computación. 2024. Vol. 25, no. 1. P. 48–59. DOI: 10.29375/25392115.5056.

*Работа выполнена в рамках Государственного задания № FSGF-2024-0006 «Разработка новых моделей и методов принятия решений в многоуровневых системах управления».

Обновляемость систем и их компонент в суперкомпьютерном рейтинге Топ50

Д.Г. Элишаков¹, Д.А. Никитенко²

МГУ имени М.В. Ломоносова¹, НИВЦ МГУ²

Топ50 — рейтинг суперкомпьютеров России, созданный НИВЦ МГУ и МСЦ РАН в декабре 2004 года, с ранжированием систем по производительности на тесте Linpack, как и в мировом рейтинге Топ500 [4, 5]. В настоящее время рейтинг включает в себя обширный набор данных и метрик. В разделе статистики доступны различные показатели: от количества процессоров, ускорителей и ядер в каждой системе до доли поставщиков компонент [1–3]. Однако отсутствует информация, связанная с обновляемостью суперкомпьютеров и их компонент. Основной задачей исследования является разработка методов анализа обновления рейтинга и входящих в него систем. Предложен метод анализа временной задержки между анонсом используемых компонент и их появлением в реальных системах, входящих в рейтинг.

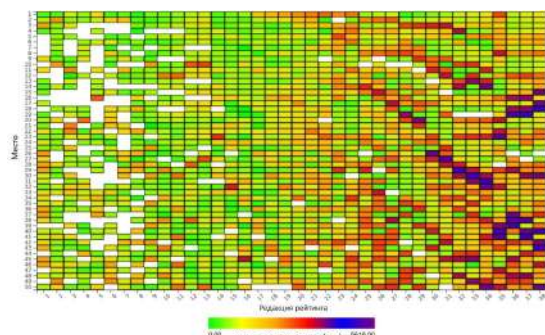


Рис. 1а) Тепловая карта задержки процессоров

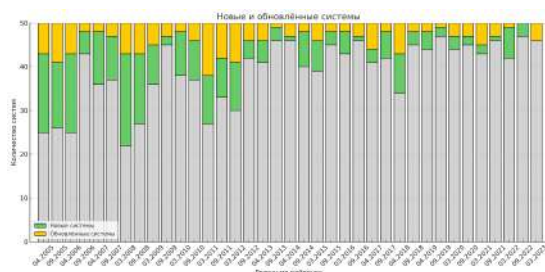


Рис. 1б) Количество новых и обновленных систем в редакциях списка Топ50

Отставание компоненты (в днях) рассчитывается как разница между датой публикации редакции рейтинга и минимумом из даты анонса и даты первого появления компоненты (так как некоторые компоненты появлялись в рейтинге раньше официального анонса) (Рис. 1а). Кроме того, разработан инструмент для оценки числа обновлений в редакциях рейтинга. Обновление системы включено в функционал рейтинга и определяется исходя из заявки (Рис. 1б). Видно, что количество новых и обновленных систем в целом уменьшается с течением времени. Рост в 2011 году связан с появлением графических ускорителей в НРС (Рис. 1б). Относительно компонент отчетливо наблюдается область в районе 2010 года (13–20 редакции), где преобладает зеленый цвет, что свидетельствует об использовании новых компонент в этот период, в том числе в связи с государственными программами поддержки установки высокопроизводительных систем в вузах (Рис. 1а). В дальнейшем планируется усовершенствовать и добавить полученную статистику на официальный сайт рейтинга.

Литература

1. Nikitenko D., Antonov A., Zheltkov A., Voevodin V. Describing HPC System Architecture for Understanding Its Capabilities // RuSCDays 2020. Vol. 1331. Springer, 2020. CCIS. P. 425–435. DOI: 10.1007/978-3-030-64616-5_37.
2. Nikitenko D., Zheltkov A. The Top50 list vivification in the evolution of HPC rankings // PCT 2017. Vol. 753. Springer, 2017. P. 14–26. CCIS. DOI: 10.1007/978-3-319-67035-5_2.
3. Nikitenko D. Hierarchical Model of Architecture of Supercomputer Systems for Comparison and Ranking // Вестник Южно-Уральского государственного университета. 2022. Т. 11, № 4. С. 5–18. DOI: 10.14529/cmse220401.
4. Kapridov A.A., Nikitenko D.A. The Top50 performance ranking statistical data processing and visualization methods // Труды международной научной конференции ПаВТ'2019. Челябинск: Издательский центр ЮУрГУ. С. 19–33.
5. Рейтинг суперкомпьютеров мира TOP500. URL: <https://www.top500.org/>

Индекс по фамилиям

А

Abramyan M.E., 297

В

Barablina Y.E., 8

Bartashevich E.V., 36

Beketov M.E., 299

Buryak D.Y., 44

С

Chuprov I., 16

Д

Derkach D., 16

Dragoy O., 298

Е

Efremenko D., 16

Efremov A., 56

И

Ilyasov Y., 56

К

Krainukov N.I., 297

Kychkin A., 16

Л

Levin A.S., 299

М

Makarova T.M., 36

Minnigulova A., 298

Н

Novikov D.A., 44

Nurmatova E.V., 300

Р

Pisarev V., 56

Promyslov G., 56

Р

Raudsep V.A., 8

С

Sabidaeva E.A., 90

Shchur L.N., 82

Sukhoverkhova D.D., 82

Surikova E.V., 90

Т

Timofeev A., 56

У

Virabyan A., 99

З

Zavyalov G.Y., 299

Zaytsev E.I., 300

А

Абрамкин Д.С., 330

Абрамов И.А., 301

Абрамян М.Э., 186

Авдошин С.М., 302

Акименко С.С., 316

Антонов А.С., 328

Арутюнов Г.А., 302

Б

Башара В.О., 193

Борзунов С.В., 303

Борисовский П.А., 304

Борщук О.С., 107

Бритвин Р.М., 326

Будков Ю.А., 325

Будников А.С., 305

В

Валиев Ф.Ф., 311

Васёв П.А., 116, 306

Васильев В.С., 151

Г

Гордиенко Д.В., 307

Горохов С.А., 235

Громыко Г.Ф., 322

Гудков В.А., 132

Гуленок А.А., 132

Гурина Д.Л., 325

Д

Дегаев М.Н., 308

Добровольцев А.С., 309

Дордопуло А.И., 132

Дронов А.Г., 310

Е

Емельяненко М.В., 193

Ж

Жаров М.А., 311

Жуков К.А., 172

З

Замятина Е.Б., 259

Захаров А.О., 304, 312

Захаров П.В., 218

Захарова Ю.В., 304

Зиннуров Б.Д., 313

Зонтов Ю.В., 142

Зыкина А.В., 334

И

Иванов А.А., 142

Иванов В.О., 315

Ивличев Е.П., 314

Инютин С.А., 315

К

Калягин В.А., 164

Камакин А.Ю., 301

Карабекян Д.С., 142

Карпачев Д.О., 320

Карпова А.В., 316

Киреев С.Е., 317, 330

Киреева А.Е., 330

Кисленко В.А., 307

Кисленко С.А., 307

Комардин М.А., 326

Косов П.В., 151

Костенецкий П.С., 321, 324, 329, 331

Костылев И.Д., 164

Краснопольский Б.И., 172

Куприй Р.М., 172

Кургалин С.Д., 303

Л

Левин И.И., 132

Левченко А.В., 318

Легалов А.И., 151

Литвиненко А.М., 302

Лиходед Н.А., 322

Логинова И.П., 319

Лоос Р.А., 320

Лукашин А.А., 332

Лукьяненко Д.В., 320

Лю Бовень, 186

Ляпич Н.С., 317

М

Мазаев И.А., 321

Майер Р.В., 328

Малиев Э.Е., 322

Мельников Б.Ф., 186, 323

Мишенин Р.М., 324, 331

Моисеев Н.Д., 301

Монахов В.В., 311

Морозов В.И., 193

Мустайкин М.С., 211

Мэн Линцянь, 323

Н

Наволоцкий Н.А., 320

Нарсеев А.С., 218

Никитенко Д.А., 335

О

Одинцова Е.Г., 325

П

Павлов С.В., 307

Панилов П.А., 326

Парфенов Д.Р., 327

Пескова Е.Е., 211

Петухов Е.П., 332

Пионткевич А.Г., 328

Поршнева С.В., 116

Потапов А.Г., 226

Прямыков Д.Н., 320

Р

Размыслов К.В., 311

Резванов А.А., 235

Резник А.А., 235

Романов А.В., 226, 305

Рыбаков Г.С., 329

С

Сабельфельд К.К., 330

Савчук Р.С., 333

Салибекян С.М., 247

Саликова М.Т., 331

Сальников А.Н., 270

Самойлова К.В., 259

Санников Т.В., 270

Сеннов В.Н., 332

Сидоров С.Г., 314

Смирнов Г.С., 313

Сохатский М.А., 309

Степанов А.В., 284

Т

Тарарушкин Е.В., 313

Толстиков А.А., 322
Торбин С.С., 320
Трофимова Е.А., 333

Ш

Шарун И.В., 334
Шинкарев В.Д., 320

Э

Элишаков Д.Г., 335

Ю

Юлдашев А.В., 309

Я

Якуба В.И., 142
Янковская У.И., 218

Содержание

Короткие статьи

Predicting Specific Heat Capacity of Nanofluids Using Machine Learning Methods: Applications in Thermophysics <i>Y.E. Barablina, V.A. Raudsep</i>	8
Application of Physics-Informed Neural Networks for Solving the Inverse Advection-Diffusion Problem to Localize Pollution Sources <i>I. Chuprov, D. Derkach, D. Efremenko, A. Kychkin</i>	16
Construction of Self-Healing PDMS Materials Models by Supercomputer MD Simulations <i>T.M. Makarova, E.V. Bartashevich</i>	36
Research and Optimization Pruning Method in Train Process: DPIREC <i>D.A. Novikov, D.Y. Buryak</i>	44
Efficiency of Machine Learning Tasks on HPC Devices <i>G. Promyslov, A. Efremov, Y. Ilyasov, V. Pisarev, A. Timofeev</i>	56
Extraction of Properties of Anisotropic Spin Model by Deep Transfer Learning Methods <i>D.D. Sukhoverkhova, L.N. Shchur</i>	82
Comparative Study of LoRA and Full Fine-Tuning in Large Language Models <i>E.V. Surikova, E.A. Sabidaeva</i>	90
Enhancing Admission Inquiry Responses with Fine-Tuned Models and Retrieval-Augmented Generation <i>A. Virabyan</i>	99
Эффективный предобуславливатель для решения систем линейных уравнений, возникающих в задаче фильтрации в деформируемой поровой среде <i>О.С. Борщук</i>	107
Метод схем для эффективного выполнения графов задач <i>П.А. Васёв, С.В. Поршнев</i>	116
Комплекс средств высокоуровневого синтеза «Тесей» для многокристальных реконфигурируемых вычислительных систем <i>А.И. Дордопуло, И.И. Левин, В.А. Гудков, А.А. Гуленок</i>	132
Оценка степени манипулируемости известных процедур агрегирования для большого числа альтернатив и участников <i>Ю.В. Зонтов, А.А. Иванов, Д.С. Карабекян, В.И. Якуба</i>	142
Эволюционная разработка многопоточных программ с применением процедурно-параметрической парадигмы программирования <i>П.В. Косов, А.И. Легалов, В.С. Васильев</i>	151

Численный анализ неопределенности алгоритмов идентификации графовых моделей <i>И.Д. Костылев, В.А. Калягин</i>	164
Оценка эффективности различных форматов представления разреженных матриц для вычислений на графических ускорителях <i>Р.М. Куприй, Б.И. Краснопольский, К.А. Жуков</i>	172
Подход к параллельной реализации решения псевдогеометрической версии задачи коммивояжера геометрическим подходом <i>Б.Ф. Мельников, Бовень Лю, М.Э. Абрамян</i>	186
Численная оптимизация проверочной матрицы LDPC-кода для применения в протоколе квантового распределения ключей с использованием высокопараллельных вычислений <i>В.И. Морозов, В.О. Башара, М.В. Емельяненко</i>	193
Параллельный алгоритм для численного исследования дозвуковых реагирующих течений в цилиндрической системе координат <i>М.С. Мустайкин, Е.Е. Пескова</i>	211
Механические свойства композита Al_2Ni_3 , армированного углеродной нанотрубкой, при одноосной деформации растяжения <i>А.С. Нарсеев, П.В. Захаров, У.И. Янковская</i>	218
Особенности выбора параметров для нейронной сети Бехлера—Парринелло на примере кристалла лития <i>А.Г. Потапов, А.В. Романов</i>	226
Квантово-механический расчет термодинамических свойств гексагонального и кубического рутения <i>А.А. Резник, С.А. Горохов, А.А. Резванов</i>	235
Разработка языка программирования для dataflow вычислений <i>С.М. Салибекян</i>	247
Использование сетей Петри для оптимизации параллельных бизнес-процессов <i>К.В. Самойлова, Е.Б. Замятина</i>	259
Обработка потока задач с зависимостями на нескольких вычислительных кластерах <i>Т.В. Санников, А.Н. Сальников</i>	270
О параллелизации метода Монте-Карло при решении некоторых метрологических задач <i>А.В. Степанов</i>	284
Плакаты	
On Some Parallel Algorithm for NFA Beterminization <i>М.Е. Abramyan, N.I. Krainikov</i>	297

Hybrid Tractography Analysis in Patients with Brain Lesions <i>A. Minnigulova, O. Dragoy</i>	298
Cache Oblivious Fast 2D Fourier Transform <i>G.Y. Zavyalov, A.S. Levin, M.E. Beketov</i>	299
Multi-Agent Intelligent Decision Support System <i>E.I. Zaytsev, E.V. Nurmatova</i>	300
Разработка архитектуры классификатора для оценки состояния объектов инфраструктуры с применением нейронных сетей <i>И.А. Абрамов, А.Ю. Камакин, Н.Д. Мусеев</i>	301
Улучшенный вывод типов в Python с использованием глубокого обучения сходству на основе графового представления кода <i>Г.А. Арутюнов, С.М. Авдошин, А.М. Литвиненко</i>	302
Подготовка ИТ-специалистов в области квантовых вычислений и квантовой теории информации <i>С.В. Борзунов, С.Д. Кургалин</i>	303
Эволюционные алгоритмы с параллельными операторами для задачи составления расписаний выполнения заказов клиентов <i>П.А. Борисовский, Ю.В. Захарова, А.О. Захаров</i>	304
NNMD: программный пакет на Python для обучения нейронных сетей Бехлера—Парринелло <i>А.С. Будников, А.В. Романов</i>	305
Метод онлайн-визуализации параллельных программ <i>П.А. Васёв</i>	306
Сравнительный анализ многопоточной производительности CPU и GPU в JDFT-моделировании гетероструктур металл-графен <i>Д.В. Гордиенко, С.В. Павлов, В.А. Кисленко, С.А. Кисленко</i>	307
Модификация и моделирование алгоритма поиска данных в служебной таблице маршрутизатора контент-ориентированных сетей <i>М.Н. Дегаев</i>	308
О возможности применения на GPU преобусловливателя AIPS в качестве альтернативы ILU(0) в задаче гидродинамического моделирования нефтегазовых месторождений <i>А.С. Добровольцев, М.А. Сохатский, А.В. Юлдашев</i>	309
Об одном способе организации параллельной генерации сеток на основе декомпозиции расчетной области <i>А.Г. Дронов</i>	310
Использование нейросетевого подхода для оценки влияния ионопровода на восстановление начальной координаты взаимодействий <i>М.А. Жаров, К.В. Размыслов, Ф.Ф. Валиев, В.В. Монахов</i>	311

Решение задачи регрессии булевых данных с помощью параллельного алгоритма генетического программирования <i>А.О. Захаров</i>	312
Высокопроизводительное моделирование процессов роста оксигидроксидных структур алюминия <i>Б.Д. Зиннуров, Г.С. Смирнов, Е.В. Тарарушкин</i>	313
Разработка методов искусственного интеллекта для управления внешними устройствами на основе ЭЭГ в замкнутом контуре биологической обратной связи <i>Е.П. Ивлишев, С.Г. Сидоров</i>	314
Модель с подстройкой параметров для модулярных вычислений в больших компьютерных диапазонах <i>С.А. Инютин, В.О. Иванов</i>	315
Сравнительный анализ методов тензорной ренормализационной группы для моделирования двумерных конденсированных систем <i>А.В. Карпова, С.С. Акименко</i>	316
Разработка средств анализа исполнения LuNA-программ на основе факторов SLOU <i>С.Е. Киреев, Н.С. Ляпич</i>	317
Полиэдральный кодизайн тайлинга циклов в MLIR <i>А.В. Левченко</i>	318
Реализация пакетирования с использованием параллельных вычислений в системе логического проектирования <i>И.П. Логинова</i>	319
Библиотека параллельных реализаций регуляризирующих алгоритмов решения линейных некорректно поставленных обратных задач <i>Д.В. Лукьяненко, В.Д. Шинкарев, С.С. Торбин, Д.О. Карпачев, Р.А. Лоос, Н.А. Наволоцкий, Д.Н. Пряников</i>	320
Анализ и визуализация данных в системе мониторинга эффективности задач HPC TaskMaster суперкомпьютерного комплекса <i>И.А. Мазаев, П.С. Костенецкий</i>	321
Численное решение двумерных параболических уравнений гемодинамики на многоядерных процессорах <i>Э.Е. Малиев, Г.Ф. Громыко, Н.А. Лиходед, А.А. Толстиков</i>	322
Подход к параллельной реализации проверки бинарного отношения покрытия для заданной пары конечных языков <i>Б.Ф. Мельников, Линцянь Мэн</i>	323
Моделирование потока задач вычислительного кластера НИУ ВШЭ с использованием SLURM Simulator <i>Р.М. Мишенин, П.С. Костенецкий</i>	324

Молекулярно-динамическое моделирование поведения воды в условиях ограниченной геометрии кремниевых нанопор <i>Е.Г. Одинцова, Д.Л. Гурина, Ю.А. Будков</i>	325
Архитектура среды моделирования когнитивных процессов <i>П.А. Панилов, М.А. Комардин, Р.М. Бритвин</i>	326
Метод автоматического вывода алгоритмов эффективного решения задач на вычислительных моделях с шаблонами <i>Д.Р. Парфенов</i>	327
Разработка клиент-серверной системы управления пользовательскими данными в рамках проекта Algo500 <i>А.Г. Пионткевич, А.С. Антонов, Р.В. Майер</i>	328
Разработка подсистемы динамического изменения приоритета задач в очереди суперкомпьютера <i>Г.С. Рыбаков, П.С. Костенецкий</i>	329
Расчет вольт-амперных характеристик для упрощенной модели варистора <i>К.К. Сабельфельд, Д.С. Абрамкин, С.Е. Киреев, А.Е. Киреева</i>	330
Интеллектуальное прогнозирование времени выполнения вычислительных задач GROMACS для оптимизации работы планировщика задач суперкомпьютера «сHARISMa» НИУ ВШЭ <i>М.Т. Саликова, П.С. Костенецкий, Р.М. Мишенин</i>	331
Моделирование работы суперкомпьютерного кластера для оценки эффективности интеллектуальных алгоритмов в задачах диспетчеризации <i>В.Н. Сеннов, А.А. Лукашин, Е.П. Петухов</i>	332
Применение принципов визуальных интерфейсов в больших языковых моделях как альтернатива текстовому взаимодействию <i>Е.А. Трофимова, Р.С. Савчук</i>	333
Параллельный алгоритм послойного обучения нейронных сетей <i>И.В. Шарун, А.В. Зыкина</i>	334
Обновляемость систем и их компонент в суперкомпьютерном рейтинге Top50 <i>Д.Г. Эмишаков, Д.А. Никитенко</i>	335

Электронное издание

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ТЕХНОЛОГИИ
(ПаВТ'2025)

Короткие статьи и описания плакатов
(г. Москва, 8–10 апреля 2025 г.)

Издательский центр Южно-Уральского государственного университета

Подписано в печать 23.05.2025. Формат 60 × 84 1/8.
Усл. печ. л. 39,99. Заказ 124.
