Подход к параллельной реализации решения псевдогеометрической версии задачи коммивояжёра геометрическим подходом

Б.Ф. Мельников[1], Бовень Лю[1], М.Э. Абрамян[1,2]

[1]Совместный университет МГУ–ППИ в Шэньчжэне,
[2]Южный 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 «начерно», 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 коммивояжёра: для заданной матрицы расстояний

$$\{a_{i,j} \mid i,j \in \{1,2,\ldots,n\}\}$$

[1]Пользуемся «школьной терминологией».

(диагональные элементы a_{ii} обычно не используются, в программных реализациях их удобно полагать равными ∞) требуется найти перестановку $(k_1, k_2, \ldots, k_n)$, минимизирующую значение суммы

$$a_{k_1k_2} + a_{k_2k_3} + \cdots + a_{k_{n-1}k_n} + a_{k_nk_1}\,; \tag{1}$$

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

$$\sqrt{\sum_{1\le i<j\le n} (a_{ij} - \overline{a_{ij}})^2}\,,$$

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 разработанной[1] 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

[1]И 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.)

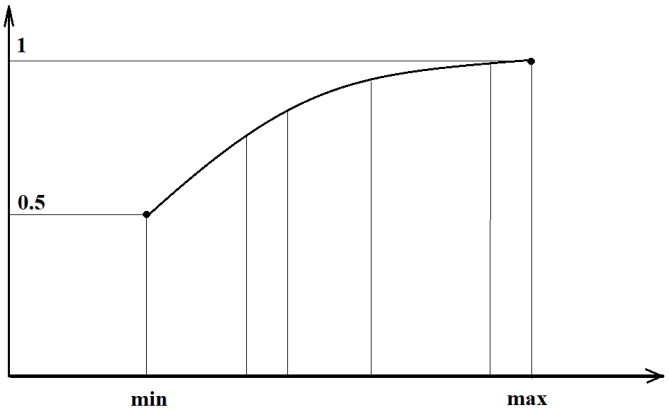

Рис. 1. Пример функции риска; бóльшие значения являются «плохими»; наихудший вариант наиболее важен

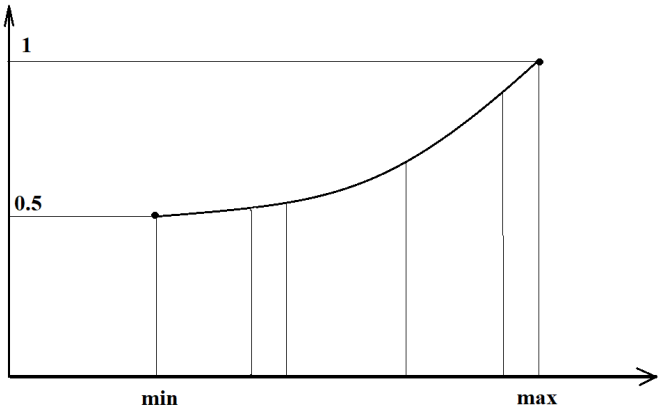

Рис. 2. Пример функции риска; бóльшие значения являются «плохими»; средний вариант наиболее важен

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.

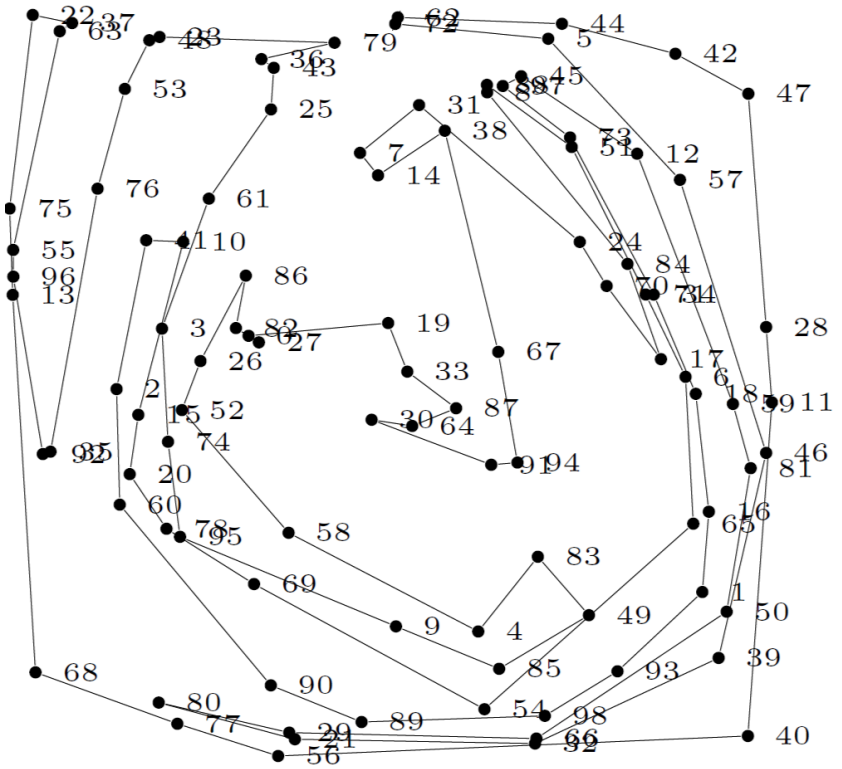

Рис. 3. Итоговое расположение точек для геометрического варианта ЗКВ

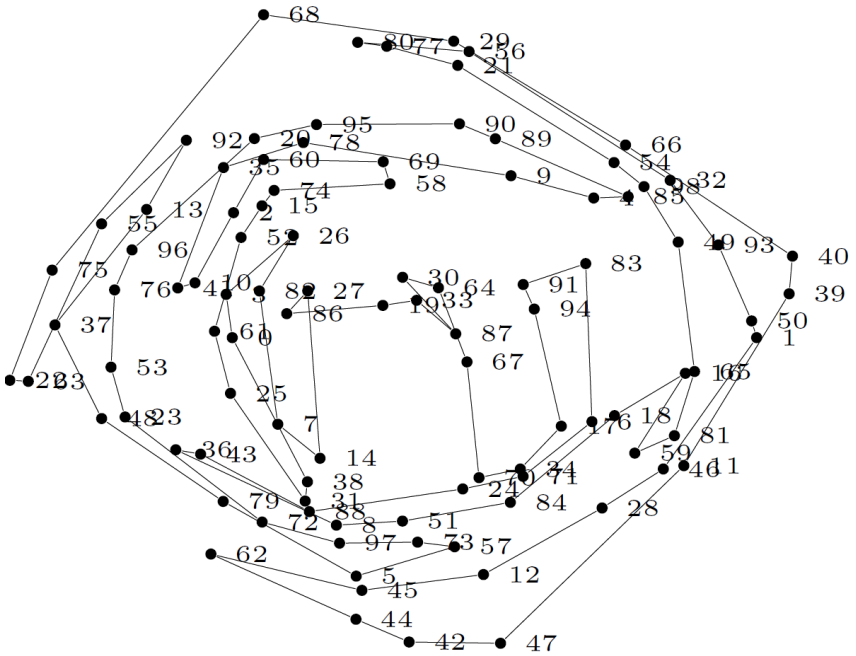

Рис. 4. Итоговое расположение точек для псевдогеометрического варианта ЗКВ

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.