

## ЭФФЕКТИВНЫЙ АЛГОРИТМ ПРИБЛИЖЕННОГО РЕШЕНИЯ МЕТРИЧЕСКОЙ ЗАДАЧИ КОММИВОЯЖЕРА

*Ю. Л. Костюк, С. А. Жихарев*

Предлагается новый алгоритм приближенного решения задачи коммивояжера. Алгоритм развивает известную идею построения маршрута по остовному дереву минимального веса. Дерево дополняется ребрами, соответствующими вторым ближайшим расстояниям для висячих вершин. Затем циклы в графе последовательно соединяются в единый маршрут. На промежуточных шагах используется локальный поиск. Рассмотрены два варианта реализации алгоритма — для двумерной евклидовой метрики за время  $O(n \log n)$  и для произвольной метрики за время  $O(n^2)$ , где  $n$  — число вершин графа. Приведены результаты численного эксперимента, подтверждающие хорошее качество получаемых решений.

### Введение

Известные алгоритмы приближенного решения задачи коммивояжера в метрическом пространстве можно разделить на две основные группы: 1) алгоритмы с относительной погрешностью 2 и временем работы  $O(n^2)$ , где  $n$  — число вершин графа; 2) алгоритмы с относительной погрешностью 1,5 и временем работы  $O(n^3)$ .

К первой группе относится алгоритм «ближайшего города» и алгоритм дерева [3]. В случае двумерной евклидовой метрики алгоритм дерева можно реализовать за время  $O(n \log n)$  [4]. В [2] нами предложен такой модифицированный алгоритм дерева, который строит в точности маршрут ближайшего города, причем в случае двумерной евклидовой метрики на это требуется время  $O(n \log n)$ .

Ко второй группе относится алгоритм Кристофидеса [3, 4], в котором самым трудоемким этапом, определяющим общее время работы, является построение паросочетания минимального веса. Для случая двумерной евклидовой метрики такое паросочетание можно найти за время  $O(n^{2,5} \log^4 n)$  [5].

Рассмотренный в [1] алгоритм использует решение задачи о назначениях, которое находится за время  $O(n^3)$  в общем случае и за время  $O(n^{2.5} \log n)$  в случае двумерной евклидовой метрики [5]. Как показано в [1], при некоторых предположениях длина маршрута, построенного этим алгоритмом, с вероятностью 1 стремится к длине оптимального маршрута. Однако для метрического случая (в том числе для двумерной евклидовой метрики) условия соответствующей теоремы заведомо не выполняются и, как показал эксперимент, алгоритм дает решение не лучшее чем алгоритм дерева.

Следует отметить, что статистическая погрешность алгоритмов оказывается меньше теоретической. В случае двумерной евклидовой метрики отношение  $\varphi$  длины приближенного маршрута к нижней оценке оптимума (сумме длин минимального остовного дерева и его максимального ребра) приблизительно равно 1,5 для алгоритма дерева и 1,25 для алгоритма Кристофидеса [2]. Здесь и далее в тексте статьи численные оценки погрешности алгоритмов получены на задачах коммивояжера при равномерном распределении  $n = 300$  точек плоскости в единичном квадрате.

Вычисленный каким-либо алгоритмом приближенный маршрут коммивояжера можно улучшить локальным поиском [3]. Для этого рассматривается скользящее окно в маршруте, состоящее из вершин с номерами  $i + 1, \dots, i + k$ , и для всех  $i$  последовательно выбираются минимизирующие маршрут перестановки. Так как величина  $k$  является константой, то общее время работы алгоритма при этом возрастает на  $O(n)$ . Локальный поиск дает заметное улучшение маршрута. Так, согласно [2] для алгоритма дерева отношение  $\varphi$  уменьшается до 1,35, а для алгоритма Кристофидеса — до 1,18 при размере окна  $k = 11$ .

В [2] нами предложено локальный поиск в окне применять на промежуточных шагах работы модифицированного алгоритма дерева при добавлении очередной вершины в маршрут. (Будем называть этот вид локального поиска *поиском по внутреннему окну*, а локальный поиск в полностью построенном маршруте — *поиском по внешнему окну*. Здесь и далее под термином *маршрут* будем понимать либо полностью построенный маршрут коммивояжера, либо некоторый цикл в графе, что будет ясно из контекста.) Как показал эксперимент, поиск по внутреннему окну в модифицированном алгоритме дерева дает существенное уменьшение длины маршрута (отношение  $\varphi$  уменьшается до 1,22 при размере окна  $k = 11$ ). Время работы алгоритма дерева при этом возрастает в десятки раз, но остается значительно меньше, чем время работы алгоритма Кристофидеса.

В данной статье предлагается новый вариант алгоритма дерева, практически не уступающий алгоритму Кристофидеса по качеству получаемых решений.

### 1. Алгоритм зацикленного дерева

Алгоритм дерева [3] содержит следующие этапы:

- 1) построение минимального остовного дерева  $T$ ;
- 2) построение мультиграфа  $G$  путем удвоения ребер в  $T$ ;
- 3) нахождение маршрута коммивояжера по эйлерову циклу в  $G$ .

Временная сложность алгоритма определяется этапом 1, так как этапы 2 и 3 осуществляются за время  $O(n)$ . В общем метрическом случае дерево  $T$  можно построить алгоритмом Прима за время  $O(n^2)$ . В случае двумерной евклидовой метрики его можно построить алгоритмом Крускала за время  $O(n \log n)$ , используя триангуляцию Делоне [4].

В предложенном нами модифицированном алгоритме дерева [2] маршрут строится по дереву  $T$  другим способом. Пусть имеется некоторый маршрут обхода части вершин графа. (Вначале выбирается маршрут, состоящий из одной произвольной вершины.) Ребро дерева, инцидентное вершине маршрута, но не входящее в него, назовем *боковым*. Вершины текущего маршрута просматриваются последовательно. Если у очередной вершины маршрута нет боковых ребер, то делается переход к следующей вершине маршрута. Вершина, связанная с вершиной маршрута боковым ребром, включается в маршрут перед или после этой вершины (по критерию минимального удлинения маршрута). Затем производится локальный поиск в окне с центром в новой вершине. Алгоритм заканчивает работу, когда все вершины графа будут включены в маршрут.

Рассмотрим модификацию алгоритма дерева, которую назовем алгоритмом *зацикленного дерева*. Предлагаемый алгоритм включает следующие этапы:

- 1) построение минимального остовного дерева  $T$ ;
- 2) последовательный просмотр висячих вершин в  $T$  и включение в граф  $T$  кратчайшего из ребер, не входящих в  $T$  и инцидентных данной висячей вершине;
- 3) выделение какого-либо цикла — начального маршрута;
- 4) обнаружение связанного с маршрутом цикла в графе  $T$  и склеивание его с маршрутом, если есть вершины, не включенные в маршрут.

В случае произвольного метрического пространства этап 2 можно реализовать за время  $O(n^2)$ . Для двумерной евклидовой метрики этот этап осуществляется за время  $O(n)$ , если просматривать только ребра триангуляции Делоне.

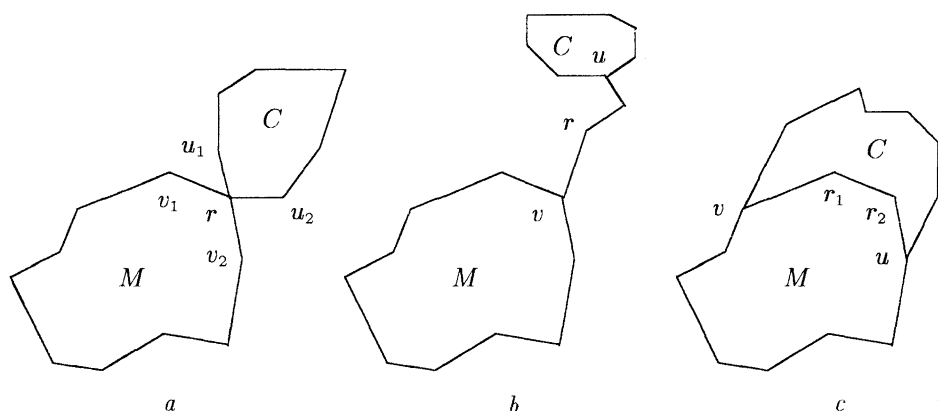


Рис. 1. Варианты взаимного расположения маршрута и цикла

Этап 3 выполняется за время  $O(n)$  поиском в ширину, начиная от произвольной вершины. Будем расставлять пометки вершин и делать ссылку из очередной просматриваемой вершины на предыдущую. Обнаружение ребра, соединяющего две помеченные вершины, означает выявление цикла — начального маршрута.

Этап 4 можно выполнить за время  $O(n)$ , если поиск цикла для склеивания проводить в глубину от такой вершины маршрута, которой инцидентно боковое ребро. Так же, как на этапе 3, расставляются метки и делаются ссылки на ранее просмотренные вершины. Так как ребра выявленного цикла после склейки удаляются из  $T$ , то общее время этапа 4 не будет превышать  $O(n)$ .

Алгоритм зацикленного дерева с рассмотренным вариантом реализации этапа 4 обозначим через А.

Возможные варианты взаимного расположения маршрута и цикла изображены на рис. 1. Ниже перечислены способы склеивания для каждого из них.

Вариант *a*. Маршрут  $M$  и цикл  $C$  сочленяются в одной вершине  $r$ . Возможны четыре способа склеивания:

- ребра  $(v_1, r)$  и  $(u_1, r)$  заменяются на ребро  $(v_1, u_1)$ ;
- ребра  $(v_2, r)$  и  $(u_2, r)$  заменяются на ребро  $(v_2, u_2)$ ;
- ребра  $(v_1, r)$  и  $(u_2, r)$  заменяются на ребро  $(v_1, u_2)$ ;
- ребра  $(v_2, r)$  и  $(u_1, r)$  заменяются на ребро  $(v_2, u_1)$ .

Среди этих способов выбирается тот, который дает минимальную длину объединенного маршрута.

Вариант *b*. Маршрут  $M$  и цикл  $C$  соединены цепью  $v, \dots, u$ . На этой цепи выбирается произвольная вершина  $r$ . Вершины цепи от  $v$  до  $r$  последовательно включаются в маршрут  $M$ , а вершины цепи от  $u$  до  $r$  — в цикл  $C$ . Каждая вершина включается либо перед, либо после

той вершины маршрута (цикла), с которой она связана ребром. Это определяется минимумом удлинения маршрута (цикла). В результате получаем описанный выше вариант (см. рис. 1, *a*). Эксперименты показали, что произвол в выборе вершины  $r$  практически не влияет на качество формируемого маршрута.

Вариант *c*. Маршрут  $M$  и цикл  $C$  имеют общую часть, вследствие чего вершины  $u$  и  $v$  оказываются соединенными между собой тремя цепями. Одна из этих цепей разрывается путем удаления какого-либо ребра  $(r_1, r_2)$ . После этого вершины цепи  $u, \dots, r_1$  последовательно включаются в объединенный маршрут аналогично тому, как это делается в варианте *b*. Таким же способом включаются в маршрут вершины цепи  $v, \dots, r_2$ . Эксперименты показали, что целесообразно из трех цепей выбирать ту, которая содержит минимальное число ребер, а удалять следует наиболее длинное ребро.

Во всех вариантах склеивание требует выполнения числа шагов, не превышающее длину цепи  $C$ . Таким образом, общее время всех операций склеивания для получения полного маршрута не превосходит  $O(n)$ .

В алгоритме зацикленного дерева можно применять локальный поиск с внутренним окном при каждом склеивании цикла с маршрутом. В случае склеивания по варианту *a* проводятся два локальных поиска: в окне с центром в вершине  $r$  и в окне с центром в той из вершин  $u_1$  или  $u_2$ , которая не соединена после склеивания с вершиной  $r$ . В случаях *b* и *c* локальный поиск проводится при каждом включении вершин в маршрут аналогично тому, как это делается в описанном выше модифицированном алгоритме дерева.

Таким образом, в случае двумерной евклидовой метрики первый этап алгоритма зацикленного дерева можно выполнить за время  $O(n \log n)$ , а три последующих этапа — за время  $O(n)$ .

Возможно усовершенствование этапа 4 алгоритма при некотором увеличении времени его работы. Заметим, что в варианте *a* длина склеенного маршрута меньше суммы длин маршрута  $M$  и цикла  $C$ , а в вариантах *b* и *c* удлинение объединенного маршрута будет тем меньше, чем короче цепь  $v, \dots, u$ . Как показали эксперименты, с большей вероятностью получается более короткая цепь, если придерживаться следующей стратегии. Поиск в глубину от вершины, принадлежащей маршруту, следует вести в двух направлениях: по крайнему правому боковому ребру и по крайнему левому боковому ребру (это возможно в двумерном случае). После обнаружения двух циклов для каждого из них вычисляется удлинение маршрута и выбирается наилучший вариант.

В худшем случае такое усовершенствование может привести к квадратичному росту времени работы алгоритма. Однако на практике время

работы алгоритма увеличивается не более чем на 10%. Согласно эксперименту отношение  $\varphi$  не превосходит 1,185.

## 2. Поиск наилучшего цикла для склеивания

Для произвольного метрического пространства временная сложность алгоритма заикленного дерева составляет  $O(n^2)$ . Она определяется временем построения минимального остовного дерева и вычисления вторых кратчайших ребер, инцидентных висячим вершинам. В этом случае на этапе 4 обнаружения циклов и склеивания их с маршрутом целесообразно затратить дополнительное время  $O(n^2)$  для поиска таких циклов, которые дают минимальное удлинение склеенного маршрута. Рассмотренную ниже модификацию алгоритма заикленного дерева обозначим через В.

Пусть  $l$  — число ребер, добавленных в граф  $T$  на этапе 2 алгоритма. На этапе 4 до тех пор, пока еще не все вершины вошли в маршрут, повторяются следующие действия: последовательно обнаруживаются циклы и наилучший из них склеивается с маршрутом.

Поиск циклов будем вести в ширину, начиная с боковых ребер вершин текущего маршрута. В ходе поиска расставляются метки вершин и обратные ссылки. Ребро, соединяющее две помеченные вершины, определяет некоторый цикл. Это ребро назовем *особым*. При дальнейшем просмотре особое ребро временно исключается, поэтому всего будет обнаружено не более  $l - 1$  циклов (одно особое ребро войдет в начальный маршрут).

Каждый обнаруженный цикл оценивается в зависимости от того, насколько он может увеличить длину маршрута. Получив оценки для всех обнаруженных циклов, выбираем тот из них, который обладает наилучшей оценкой. Этот цикл склеиваем с маршрутом точно так же, как это было описано выше. Выделение цикла производится обратным ходом по проставленным ссылкам. Время выполнения этапа 4 будет не более  $O(n^2)$ , если обнаружение, оценивание и выделение всех циклов выполняются за время  $O(n)$ . Обнаруженный цикл располагается относительно маршрута по одному из трех вариантов (см. рис. 1). Вариант расположения можно выяснить за постоянное время, если в начале поиска в ширину каждой вершине, смежной с одним из боковых ребер маршрута, приписать вторую ссылку на саму себя и эту вторую ссылку копировать для всех вершин, в которые будет переход при последующем просмотре. При обнаружении цикла следует у вершин, соединенных особым ребром, сравнить вторые ссылки (пусть они указывают на вершины  $v_1$  и  $v_2$  соответственно). Возможны три варианта (рис. 2).

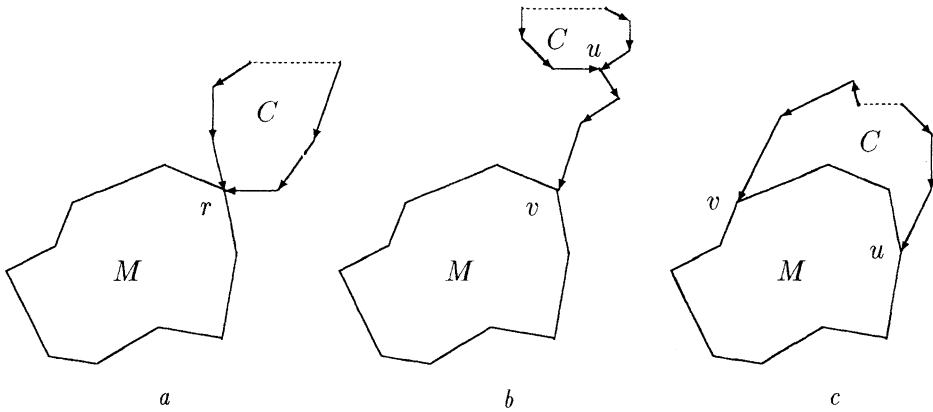


Рис. 2. Определение варианта расположения маршрута и цикла

Вариант *a*.  $v_1 \neq v_2$  и первые ссылки в вершинах  $v_1$  и  $v_2$  указывают на одну и ту же вершину.

Вариант *b*.  $v_1 = v_2$ .

Вариант *c*.  $v_1 \neq v_2$  и первые ссылки в вершинах  $v_1$  и  $v_2$  указывают на разные вершины.

Вариант *a* оценивается разностью между длиной маршрута  $M$  и цикла  $C$  до склеивания и длиной маршрута после склеивания (со знаком минус).

Вариант *b* оценивается расстоянием между конечными вершинами цепи  $v, \dots, u$ . Для получения оценки необходимо выделить цикл, чтобы найти вершину  $u$ . Однако для частично перекрывающихся циклов возможно многократное отслеживание по одним и тем же ребрам, вследствие чего выделение всех циклов на очередном шаге этапа 4 алгоритма приведет к времени  $O(n^2)$  выполнения шага и времени  $O(n^3)$  всего этапа. Во избежание этого зададим константу  $m$  ( $1 \leq m \leq l$ ), которая ограничивает число обратных проходов по одному и тому же ребру. Если  $m = l$ , то все обнаруженные циклы будут выделены и, следовательно, оценены. Если  $m < l$ , то при выделении некоторого цикла можем обнаружить ребро, уже пройденное  $m$  раз. В этом случае выделение прекращается и цикл оценивается бесконечностью. Сложность этапа оценивания всех циклов типа *b* на одном шаге этапа 4 составит не более  $O(mn)$ . В экспериментах число  $m$  выбиралось равным 10, поскольку даже при больших  $n$  редко возникала необходимость отслеживания по одним и тем же ребрам более 10 раз.

Вариант *c* оценивается расстоянием между вершинами  $v$  и  $u$ , входящими в маршрут  $M$ .

Эксперименты показали, что для алгоритма В отношение  $\varphi$  не превосходит 1,183. Это почти не уступает алгоритму Кристофидеса с локальным поиском в окне размера 11.

### 3. Результаты численного эксперимента

Численная проверка алгоритмов проводилась на случайных точках, равномерно и независимо распределенных в единичном квадрате. В табл. 1–5 приведены средние длины маршрутов, деленные на  $\sqrt{n}$ . При моделировании генерировалось такое число наборов случайных точек, которое было достаточно для вычисления средних с погрешностью менее 0,01 при доверительной вероятности 0,95. Все алгоритмы обрабатывали одни и те же наборы случайных точек. В качестве нижней оценки длины оптимального маршрута вычислялась величина, равная сумме длин минимального остовного дерева и его максимального ребра. Средние этих оценок (также деленные на  $\sqrt{n}$ ) приведены в табл. 6.

Результаты расчетов показывают, что в случае равномерного распределения точек на плоскости алгоритм заикленного дерева и алгоритм Кристофидеса находят маршруты, длины которых мало различаются. Алгоритм А (с временем работы  $O(n \log n)$ ) дает результат, более близкий к результату алгоритма Кристофидеса, чем к результату алгоритма модифицированного дерева.

Т а б л и ц а 1

Алгоритм дерева с внешним окном размера  $k$ 

$n$	$k = 1$	$k = 3$	$k = 7$	$k = 11$
50	1,043	0,993	0,929	0,884
100	1,033	0,987	0,928	0,895
150	1,030	0,986	0,932	0,899
300	1,017	0,972	0,920	0,894

Т а б л и ц а 2

Модифицированный алгоритм дерева  
с внутренним и внешним окнами размера  $k$ 

$n$	$k = 1$	$k = 3$	$k = 7$	$k = 11$
50	0,978	0,937	0,864	0,839
100	0,969	0,925	0,857	0,830
150	0,967	0,924	0,851	0,825
300	0,954	0,912	0,839	0,813
1000	0,940	0,899	0,828	0,803
5000	0,932	0,892	0,823	0,798

Т а б л и ц а 3

Алгоритм Кристофидеса  
с внешним окном размера  $k$ 

$n$	$k = 1$	$k = 3$	$k = 7$	$k = 11$
50	0,885	0,860	0,840	0,829
100	0,859	0,835	0,820	0,812
150	0,848	0,826	0,812	0,804
300	0,832	0,811	0,797	0,791

Т а б л и ц а 4

Алгоритм А зацикленного дерева  
с внутренним и внешним окнами размера  $k$

$n$	$k = 1$	$k = 3$	$k = 7$	$k = 11$
50	0,880	0,864	0,842	0,830
100	0,866	0,849	0,824	0,814
150	0,864	0,848	0,821	0,810
300	0,855	0,835	0,806	0,794
1000	0,849	0,833	0,803	0,791
5000	0,844	0,822	0,794	0,782

Т а б л и ц а 5

Алгоритм В зацикленного дерева  
с внутренним и внешним окнами размера  $k$

$n$	$k = 1$	$k = 3$	$k = 7$	$k = 11$
50	0,882	0,864	0,843	0,830
100	0,860	0,843	0,822	0,812
150	0,857	0,840	0,816	0,804
300	0,845	0,829	0,804	0,793
1000	0,838	0,820	0,794	0,783
5000	0,833	0,813	0,788	0,776

Т а б л и ц а 6

Нижняя оценка длины оптимального маршрута коммивояжера

$n$	50	100	150	300	1000	5000
Нижняя оценка	0,712	0,693	0,684	0,670	0,659	0,652

Алгоритм В (с временем работы  $O(n^2)$ ) дает результат, еще более близкий к результату алгоритма Кристофидеса. Впрочем, дополнительное улучшение, составляющее доли процента, при больших размерностях может не оправдываться существенным увеличением времени счета. Поэтому для плоского случая алгоритм А более предпочтителен. Алгоритм В целесообразно использовать для пространств большей размерности или для неевклидовых метрик, когда этап 1 алгоритма (вычисление минимального остовного дерева) все равно требует времени  $O(n^2)$ .

## ЛИТЕРАТУРА

1. Гимади Э. Х., Глебов Н. И., Сердюков А. И. Алгоритм для приближенного решения задачи коммивояжера и его вероятностный анализ // Сиб. журн. исслед. операций. 1994. Т. 1, № 2. С. 8–17.
2. Жихарев С. А., Костюк Ю. Л. Локальный поиск в метрической задаче коммивояжера // Геоинформатика: Теория и практика. Томск: Изд-во Том. ун-та, 1998. Вып. 1. С. 84–95.
3. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация: Алгоритмы и сложность. М.: Мир, 1985.

4. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение. М.: Мир, 1989.
5. Vaidya P. M. Geometry helps in matching // SIAM J. Comput. 1989. V. 18, N 6. P. 1201–1225.

Адрес авторов:

Томский  
государственный университет,  
факультет информатики,  
пр. Ленина, 36,  
634050 Томск, Россия.  
E-mail: kost@inf.tsu.ru

Статья поступила

6 января 2000 г.