УДК 681.3

Шуть В.Н., Бычинский Д.И., Сахарчук М.Г.

ЭФФЕКТИВНЫЙ АЛГОРИТМ ПРИБЛИЖЕННОГО РЕШЕНИЯ МЕТРИЧЕСКОЙ ЗАДАЧИ КОММИВОЯЖЕРА

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].

Ко второй группе относится алгоритм Кристофидеса [3], 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 $O(n^{2.5}\log^4 n)$[4].

АЛГОРИТМ ДЕРЕВА

Для построения начального маршрута будем использовать новый вариант алгоритма дерева, практически не уступающий алгоритму Кристофидеса по качеству получаемых решений.

Алгоритм дерева [3] содержит следующие этапы:

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

Временная сложность алгоритма определяется первым этапом, 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 [3].

В предложенном в [2] модифицированном алгоритме дерева 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.

АЛГОРИТМ ЗАЦИКЛЕННОГО ДЕРЕВА

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 никла в графе Т и вклеивание 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.

Этап 3 выполняется за время $O(n)$ поиском в ширину, начиная от произвольной вершины. Будем расставлять пометки вершин и делать ссыпку 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 $O(n)$.

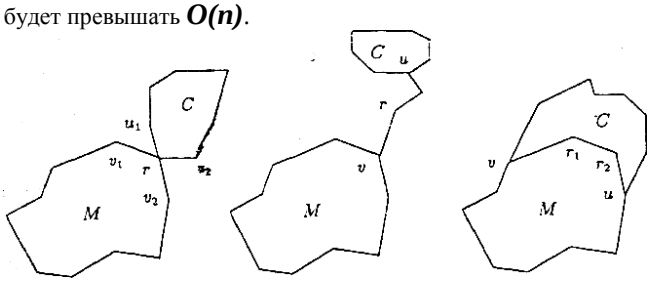

Рис. 1. Варианты взаимного расположения маршрута и цикла

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

Вариант a. Маршрут М и цикл С 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, г) заменяются на ребро (v_2, u_1);

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

Вариант b. Маршрут М и цикл С соединены цепью v,... ,u. На этой цепи выбирается произвольная вершина r. Вершины цепи от v до r последовательно включаются в

Шуть Василий Николаевич. *Доцент. каф. ЭВМ и С Брестского государственного технического университета.*
Бычинский Д.И*. Студент 5 курса специальности ЭВМ и С Брестского государственного технического университета.*
Сахарук М.Г. *Студент 5 курса специальности ЭВМ и С Брестского государственного технического университета*
Беларусь, БГТУ, 224017, г. Брест, ул. Московская, 267.

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, а). 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,... , r_1 последовательно включаются в объединенный маршрут аналогично тому, как это делается в варианте b. Таким же способом включаются в маршрут вершины цепи v,... ,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,... ,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%.

Для произвольного метрического пространства временная сложность алгоритма зацикленного 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.

Результаты расчетов показывают, что в случае равномерного распределения точек на плоскости алгоритм зацикленного дерева и алгоритм Кристофидеса находят маршруты, длины которых мало различаются.

После построения маршрута его длину можно уменьшить, проведя несколько итераций по оптимизации. Для улучшения маршрута будем использовать три типа оптимизации:

- перемещение вершин — исключение из маршрута какой-либо вершины и включение ее в другое ребро, если это приводит к уменьшению длины маршрута;
- обмен ребрами — разрыв связей между двумя парами вершин и связывание их в новом порядке, например, из связей AB и CD образуем AC и BD, если это приводит к уменьшению длины маршрута (это исключит пересекающиеся ребра, если они каким-либо образом окажутся в маршруте);
- исключение петель — петлей будем называть цепь из некоторого количества точек, соединенных последовательно, расстояние между крайними вершинами которой значительно меньше длины самой цепи.

Разумеется, для проведения итераций по оптимизации нет необходимости проверять все ребра и вершины, достаточно ограничится константным предельным расстоянием, которое будет радиусом области, содержащей участки маршрута, изменение которых может уменьшить общую длину контура. Так и для этапа исключения петель можно ограничится максимальным количеством узлов в петле, что увеличивает вероятность не обнаружения петли. Тут, как и везде впрочем, необходимо искать компромисс между временем вычислений и качеством результатов.

АЛГОРИТМ ОГРАНИЧЕННОЙ ОБЛАСТИ

Рассмотрим алгоритм включения вершин в замкнутый контур по принципу минимального увеличения общей длины контура. Изначально все вершины задаются в виде (x_i, y_i). Тогда расстояние между вершинами вычисляется по формуле:

$$d_{ij} = \sqrt{(x_i - x_i)^2 + (y_i - y_i)^2}$$

Шаги алгоритма:

1. Образование начального контура. Находим крайние вершины (если проводить аналогию с картой и городами: самый северный, самый южный, самый восточный и самый западный город) (рис. 2).

2. Получаем исходный замкнутый контур, соединяя полученные вершины (рис. 3).

3. Находим новые крайние вершины (рис.4).

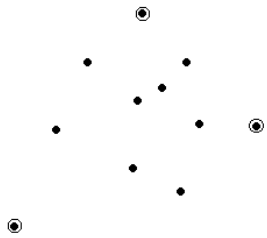

Рис. 2.

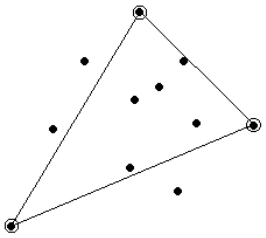

Рис. 3.

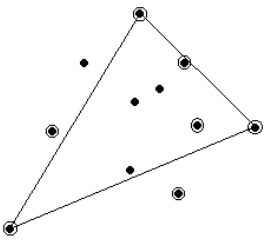

Рис. 4.

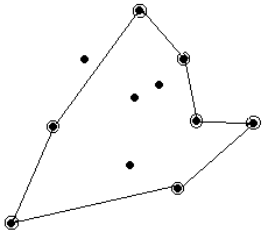

Рис. 5.

4. Включение этих вершин во внешний конур. Для этого для каждой выбранной вершины из пункта 3 находится такое ребро, при разрыве которого и включении точки в этот раз-

рыв приращение общей длины обводного контура будет минимальным. Для выполнения сего пункта последовательно берутся по две вершины из уже существующего контура. Затем находится сумма расстояний от включаемой вершины до двух вершин из контура и длина ребра между этими вершинами. Приращение контура будет разница между суммой расстояний до включаемой вершины и длиной ребра. Исследовав, таким образом, весь существующий контур находится минимальное приращение контура. После чего данное ребро разрывается и в него включается новая вершина.

Для первой итерации рис. 5.

6. Проверка на нахождение минимального маршрута среди всех заданных точек. Переходим к п. 3 до тех пор, пока все N вершин не будут включены в итоговый контур.

7. Вывод результатов. После решения задачи на терминал выводится найденный контур в виде последовательно соединенных вершин, а также длина этого контура.

К достоинствам данного алгоритма можно отнести не требовательность к вычислительной системе и высокое быстродействие.

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Численная проверка, алгоритмов проводилась на случайных, точках, распределенных равномерно и независимо. Результаты расчетов (таблица 1, рис. 6) показывают, что в случае равномерного распределения точек на плоскости алгоритм зацикленного дерева с последующей оптимизацией маршрута превосходит в большинстве случаев алгоритм граничной области, но существенно уступает ему по времени, что и не стало большой неожиданностью для разработчиков. С другой стороны не замерялось время работы алгоритма дерева, необходимое для построения маршрута, близкого по длине к результатам работы алгоритма ограниченной области.

Для принятия управленческих решений остро стоит вопрос соотношения максимальной эффективности решения оптимизационных задач и скоростью принятия решений. Поэтому оба алгоритма найдут соответствующее применение на практике.

Таблица 1 – Сравнение результатов работы описываемых алгоритмов

количество точек	Алгоритм дерева	Алгоритм области
10	**1620.8**	**1620.8**
	1577.7	**1577.7**
	1531.8	**1531.8**
	1350.3	**1350.3**
	1762.0	**1762.0**
50	**3266.8**	**3365.2**
	3051.6	**3108.7**
	3114.1	**3145.5**
	3254.6	**3237.1**
	3021.0	**3129.2**
80	**4071.6**	**4327.2**
	4212.6	**4182.2**
	3909.4	**4045.3**
	3750.3	**3910.6**
	3706.2	**3831.3**
100	**4676.5**	**4712.0**
	4474.8	**4570.5**
	4365.0	**4484.7**
	4255.0	**4372.9**
	4400.2	**4371.7**
1 000	**13260.1**	**14197.7**
	12504.4	**13177.7**
	12550.6	**13150.2**
	13623.3	**14425.8**
	13411.0	**13974.9**

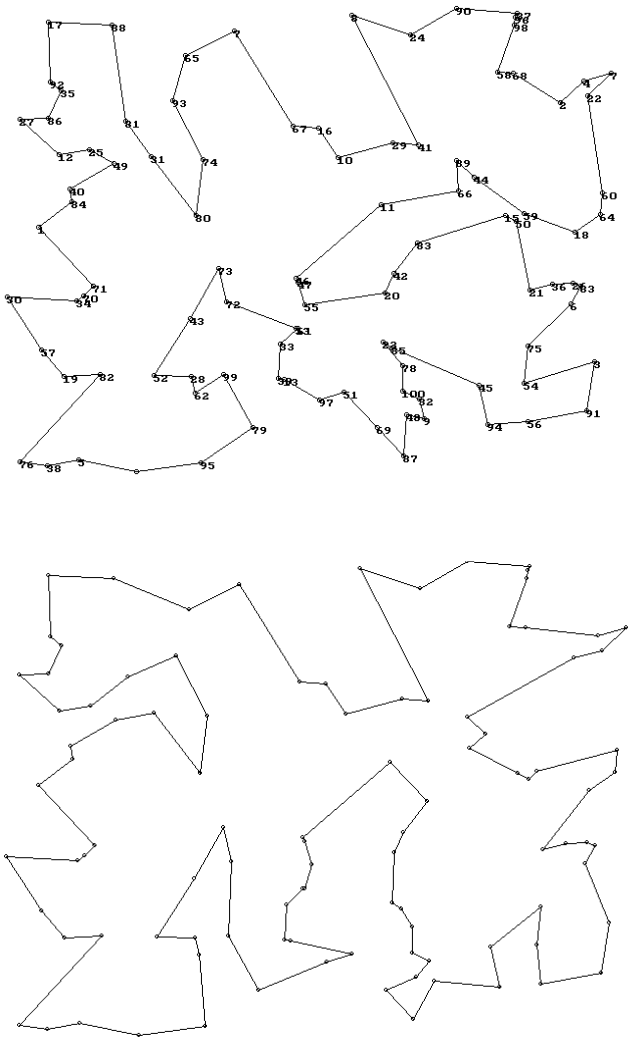

Рис. 6 – Примеры маршрутов построенных алгоритмами ограниченной области (сверху) и зацикленного дерева (снизу).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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., Стайглид К. Комбинаторная оптимизация: Алгоритмы и сложность. М.: Мир, 1985.
4. Дискретный анализ и исследование операций. Январь-июнь 2000. Серия 2. Том 7, №1.