сообщения
объединенного
института
ядерных
исследований
дубна

10-88-591

З.И.Коженкова, В.Л.Пахомов

СРАВНЕНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА

1988

Задача коммивояжера эквивалентна нахождению кратчайшего гамильтонова цикла в полном графе, каждому ребру которого сопоставлен вес (длина). При этом городам соответствуют вершины, а дорогам — ребра графа.

Задача коммивояжера является одной из комбинаторных оптимизационных задач, для точного решения которых эффективные алгоритмы неизвестны. В полном графе порядка N существует $(N - 1)!/2$ гамильтоновых циклов. Для таких трудных с вычислительной точки зрения задач один из подходов состоит в том, чтобы ослабить требование глобальной оптимальности результата и требовать, чтобы результат был разумно близок к оптимальному. При этом ожидается, что проигрыш в стоимости найденных (квазиоптимальных) решений будет умеренным.

Исходя из этого факта, мы решили определить наиболее эффективный эвристический алгоритм из известных. Такая постановка задачи диктовалась необходимостью объективного выбора алгоритма оптимизации работы программно управляемого оборудования, используемого при производстве печатных плат. В случае N городов все исследуемые алгоритмы требуют времени порядка $O(N^2)$, можно показать, что они дают маршруты, длина которых не слишком отличается от длины оптимального маршрута, если матрица расстояний симметрична и удовлетворяет неравенству треугольника[1]. Такие приближенные алгоритмы имеют большую практическую ценность для задач, в которых время, необходимое для нахождения оптимального решения, быстро растет с увеличением размерности задачи.

Пусть в графе порядка N L — длина кратчайшего дерева, а C — длина кратчайшего маршрута коммивояжера, тогда справедлива оценка: $L < C \leq 2L$.

Эти значения получаются, например, в случае, если вершины графа являются вершинами правильного многоугольника, или если все вершины расположены на одной прямой. В общем случае для произвольного графа верхнюю оценку дает контур, порождаемый двойным обходом кратчайшего дерева.

С целью исследования алгоритмов был проведен вычислительный эксперимент для задачи коммивояжера со 100 городами. В качестве исходных данных взяты точки со случайно полученными координатами, приведенными в работе[2]. При реализации всех алгоритмов были вы брана одна структура данных, в которой для каждой вершины запоминаются ее координаты, указатель на следующую вершину в контуре и длина соответствующего ребра. Результатом работы всех алгоритмов является такая же структура данных. Это позволяет применять алгоритмы в произвольной последовательности. Вычислительный эксперимент был проведен на ЭВМ CDC-6500 и БЭСМ-6. Результаты выводились на графопостроитель для наглядной оценки работы алгоритмов. Эффективность алгоритмов оценивалась по длине полученного контура и времени работы на CDC-6500. Были реализованы следующие алгоритмы:

Алгоритм 1.1. Реализован "метод ближайшего соседа"/1/. В качестве начальной вершины выбирается первая в массиве данных. Среди всех еще не посещавшихся вершин в качестве следующей выбирается ближайшая к только что выбранной (т.е. к последней вершине, включенной в маршрут); если все вершины посещались, возвращаемся в исходный пункт (рис.1).

Время работы алгоритма T = 0,725 с.

Длина полученного контура L = 1394,705.

Все исследованные модификации этого алгоритма, отличающиеся методом выбора начальной вершины, к успеху не приводят. Для "метода ближайшего соседа" условием локальной оптимальности является то, что выбираемая на каждом шаге вершина является ближайшей к последней пройденной. Это условие не является условием глобальной оптимальности, т.к. небольшое отклонение в начале пути может дать большую экономию позже, что можно было наглядно увидеть в ходе исследования.

Алгоритм 1.2. После соединения двух первых вершин, далее, на каждом шаге мы уже имеем некоторую цепь. Наращивание этой цепи происходит с обоих концов, т.е. подключаем вершину, ближайшую к какому-либо концу цепи/1/.

Рис.1. Результат работы алгоритма 1.1.

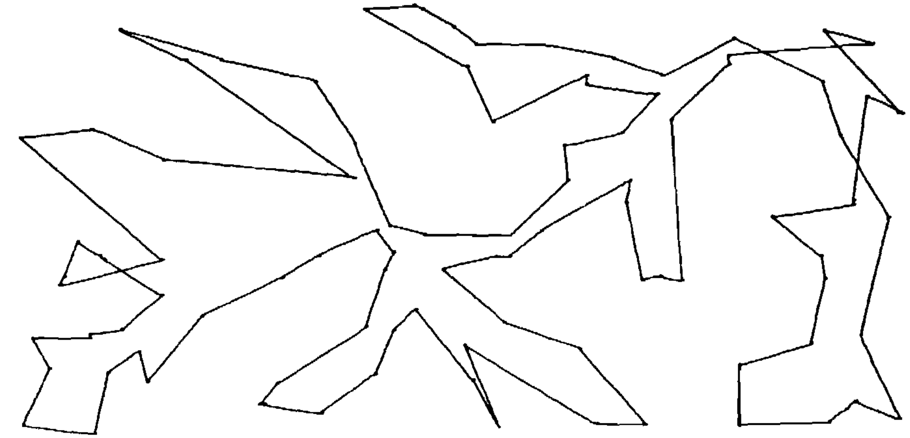


Рис.2. Результат работы алгоритма 2.

Время работы алгоритма T = 1,009 с.

Длина полученного контура L = 1410,887.

Алгоритм 2. Реализован метод "включения ближайшего города"/1/. Первая вершина выбирается произвольно, например, как первая в массиве данных. Новая вершина, добавляемая на каждом шаге, выбирается из вершин, еще не вошедших в маршрут, как ближайшая к некоторой вершине, уже вошедшей в маршрут (т.е. как ближайшая к маршруту), и включается в маршрут следующей после ближайшей к ней вершины из маршрута (рис.2).

Этот метод сложнее, чем "метод ближайшего соседа", но его также можно реализовать за время, пропорциональное N^2. Программа, реализующая данный алгоритм, получена модификацией программы поиска кратчайшего дерева.

Для этого метода доказана теорема/1/: если L – длина контура, полученного при работе этого метода, а C – длина оптимального контура, то справедливо следующее неравенство

$L \leq 2C$;

равенство выполняется только в случае, если города являются вершинами правильного многоугольника или лежат на одной прямой.

Время работы алгоритма T = 0,807 с.

Длина полученного контура L = 1468,575

Алгоритм 3.1. Реализован метод "самого дешевого включения"/1/.

Этот метод похож на предыдущий, но при включении в маршрут новой вершины выбирается не ближайшая, а обеспечивающая минимальный прирост длины контура, т.е. ищется такая вершина k, еще не включенная в маршрут, для которой справедливо следующее условие

$\min(L[ki] + L[kj] - L[ij])$,

где i, j — вершины, уже включенные в маршрут. Если в качестве начальной вершины берем первую в массиве данных, то имеем

время работы алгоритма T = 22,480 с,

длина полученного контура L = 1318,911.

Алгоритм 3.2. Если в качестве начального ребра выбрать минимальное из всех возможных C_n^2 ребер, то

время работы алгоритма T = 23,178 с,

длина полученного контура L = 1354,107.

Алгоритм 3.3. В качестве начальной вершины берем первую в массиве данных. Вершиной, добавляемой в маршрут на каждом шаге, является просто следующая из массива данных. Эта вершина добавляется по принципу "самого дешевого включения", при этом не происходит перебора оставшихся вершин (см.рис.3).

Время работы алгоритма T = 0,708 с.

Длина полученного контура L = 1227,185.

Все вышеперечисленные алгоритмы не проверяют наличие самопересечений полученного контура. Известно, что контур минимальной длины не должен иметь самопересечения.

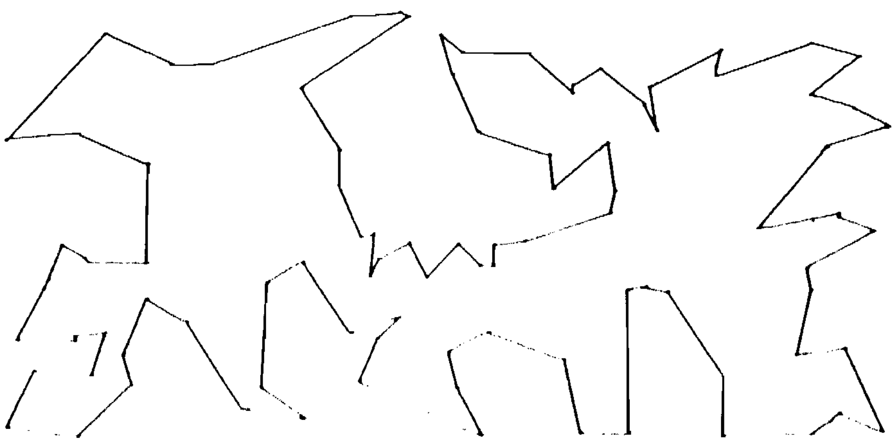
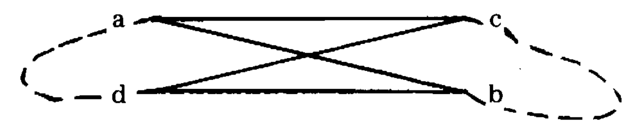


Рис 3. Результат работы алгоритма 3.3

Алгоритм 4. Этот алгоритм основан на следующей теореме. Есть контур: ...a,b,...,c,d,..., пусть L[ab] — длина ребра ab и L[cd] — длина ребра cd в этом контуре. Если ребра ab и cd пересекаются, то справедливо следующее неравенство

$L[ab] + L[cd] \geqslant L[ca] + L[db]$,

где L[ca] — длина ребра ca, а L[db] — длина ребра db.



Этот критерий можно использовать для ликвидации пересечений ребер в контуре.

Если в контуре ...a,b,...,c,d,...

$L[ab] + L[cd] > L[ca] + L[db]$,

то ребра ab и cd заменяются на ребра ac и bd. Этот процесс продолжается до тех пор, пока просмотр всех пар ребер дает уменьшение длины контура/4/.

Время работы алгоритма T = 12,982 с.

Длина полученного контура L = 1244,220.

В заключение следует отметить, что упорядочение исходного массива по координате X или Y не всегда ведет к улучшению результата, т.е. к уменьшению длины контура, отсюда можно сделать вывод, что рассмотренные методы довольно эффективны и не сильно зависят от начальной конфигурации. Комбинирование алгоритмов дает незначительное улучшение результата, но при этом возрастает время работы программы. Применение алгоритма 4 к результатам работы любого другого алгоритма, как правило, всегда приводит к незначительному уменьшению длины контура.

Наименьшая длина контура, которую удалось получить (алг.3.3 + + алг.4), равна 1161,937, за время 5,205 с.

Длина минимального дерева 988,081 и время его поиска 0,617 с/5/.

Применение всех перечисленных алгоритмов к различным наборам данных показало устойчивое преимущество алгоритма 3.3 по длине получаемого контура и времени счета

В заключение авторы благодарят Н.Н.Говоруна за поддержку этой работы.

ЛИТЕРАТУРА

1. *Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. М.: Мир, 1980*

2. *Пахомов В.Л., Пырвулеску К. ОИЯИ, Р11-10634, Дубна, 1977*

3. *Пахомов В.Л. ОИЯИ, Р11-85-910, Дубна, 1985.*
4. *Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. М.: Мир, 1985.*
5. *Ососков Г.А., Пахомов В.Л., Коженкова З.И.-В кн.: Труды V Международного совещания по проблемам математического моделирования, программированию и математическим методам решения физических задач. ОИЯИ, 11-84-818, Дубна, 1985.*

Рукопись поступила в издательский отдел
2 августа 1988 года