«Задача коммивояжера»

Техническое задание

Выполнил: ст. гр. ПЗПИ-16-1, Асландуков Матвей

Задача коммивояжера – наиболее известная задача транспортной логистики, которая заключается в нахождении кратчайшего замкнутого маршрута, проходящего через каждый из *N* заданных городов ровно один раз. Таким образом, входными данными для этой задачи является число городов *N* и описание всех ребер графа, как правило заданное матрицей смежности *d: di,j* (0 <= *di,j*) означает расстояние между городами *i* и *j.*

Также существуют различные частные случаи данной задачи, которые накладывают различные ограничения на матрицу *d:*

* геометрическая задача коммивояжера – когда города представляют собой точки на плоскости, а расстояние между двумя различными городами вычисляется как эвклидово расстояние между соответствующими точками;
* метрическая задача коммивояжера – когда на матрицу смежности расстояний *d* накладывается неравенство треугольника: для любых различных *i, j, k* верно *di, j <= di, k + dk, j*;
* симметричная и ассиметричная задачи коммивояжера – когда матрица *d* является симметричной и ассиметричной соответственно.

Как известно, данная задача вместе со всеми описанными ее частными случаями является NP-полной, что говорит о невозможности нахождения точного решения за полиномиальное от *N* время. Самым наивным экспоненциальным решением является простой перебор всех *(N – 1)!* возможных маршрутов и последующий выбор маршрута, имеющего минимальную суммарную стоимость. Однако существует множество различных методов, работающих за полиномиальное от *N* время и дающих результат, гарантированно не сильно превосходящий оптимальный.

Целью данного проекта и будет изучение и сравнение между собой различных приближенных методов решения данной задачи.

В общем случае в решении данной задачи можно выделить два основных этапа:

1. нахождение «стартового» маршрута;
2. последовательное улучшение стартового маршрута методом локальных оптимизаций.

В ходе изучения различных материалов по данной задаче (см. перечень ссылок в конце), мы выяснили, что для нахождения «стартового» маршрута существует 5 наиболее распространенных метода:

1. Выбор случайного маршрута среди всех *(N – 1)!* возможных. Небольшой его модификацией является выбор лучшего маршрута среди *K* случайных, где параметр *K* можно варьировать, в зависимости от требуемого времени работы*;*
2. Метод ближайшего соседа – на каждом из (*N – 1)* шагов будем переходить в такую вершину, расстояние до которой от текущей минимально;
3. Нахождение минимального остовного дерева в описанном графе и дальнейшее выделение в нем эйлерова цикла;
4. Нахождение минимального остовного дерева в описанном графе и дальнейшее выделение в паросочетания;
5. Битонический коммивояжер – нахождение оптимального пути среди всех, которые идут сначала слева направо, потом справа налево.

Первый метод является самым простым в реализации и быстрым по времени работы, однако и самым малоэффективным в большинстве случаев, т.к. случайный маршрут, скорее всего, будет очень далек от оптимального.

Второй метод также довольно простой в реализации, зато работает гораздо лучше. Можно доказать, что он дает *log N*-приближение исходной задачи, а на случайных графах вообще показывает довольно хорошие результаты (*x-приближение* означает, что итоговая стоимость маршрута будет отличаться от оптимальной не более чем в *x* раз).

Третий метод является уже не таким тривиальным в плане реализации, но таким же быстрым по времени работы. Зато, в отличие от первого и второго методов, он уже гарантированно дает *2-приближение* исходной задачи в случае метрической задачи коммивояжера.

Четвертый метод является еще более трудным в плане реализации и куда более затратным по времени (т.к. поиск паросочетания требует асимптотически значительно больших затрат), зато даёт уже *1.5-приближение* исходной задачи, опять таки, в случае метрической задачи коммивояжера.

Пятый метод является довольно простым в реализации, и требует квадратичные затраты времени и памяти. Однако, стоит отметить, что не смотря на то, что он всегда находит лучший битонический маршрут, он все-равно может оказаться далеким от оптимального, т.к. оптимальный может быть совсем не битоническим.

Также стоит отметить, что хоть четвертый и третий методы дают гарантированные неплохие приближения, это не значит, что четвертый метод всегда лучше третьего, третий лучше второго и т.д. Все эти методы имеют абсолютно разную структуру, и на различных графах лучшим может оказаться любой из них. Например, при удачном стечении обстоятельств, первый алгоритм может «случайно» сразу выбрать оптимальный маршрут.

Для дальнейшего улучшения «стартового» маршрута будут использоваться различные методы локальных оптимизаций:

1. 2-opt;
2. k-opt.

Подробно об этих методах будет расписано в третьем этапе данного проекта (описание предлагаемого решения).

**После анализа материалов, были выделены такие алгоритмы и структуры данных:**

* алгоритм *Прима* для нахождения минимального остовного дерева;
* *dfs* для выделения эйлерова цикла;
* *динамическое программирование по маскам* для решения задачи при маленьких ограничениях;
* *алгоритм Джона Бентли* для поиска кратчайшего битонического пути;
* *stack* для быстрого нахождения пары (тройки) вершин, в которых можно изменить последовательность ребер в ходе одной итерации метода локальных оптимизаций.

**Будет реализовано:**

В первую очередь отметим, что будет рассмотрен геометрический случай задачи коммивояжера, т.к. он является самым естественным при визуализации данной задачи (например, поиск кратчайшего туристического маршрута по карте мира).

Будут реализованы все пять методов поиска «стартового» маршрута, оба метода локальных оптимизаций, а также все десять возможных комбинаций нахождения итогового маршрута.

Также для доказательства заявленных *x-приближений* некоторых описанных выше методов поиска «стартового» маршрута будет реализован экспоненциальный алгоритм динамического программирования «по маскам», позволяющий за разумное время для небольших ограничений (*1 <= N <= 20*) находить гарантированно оптимальный маршрут. Этот алгоритм будет использоваться для сравнения всех остальных методов с оптимальным ответом на небольших графах.

Все десть описанных методов будут реализованы в виде консольной утилиты, которой на вход подается граф, заданный координатами городов, а на выходе создается файл с найденным маршрутом. У этой утилиты также будет присутствовать функциональность визуализации заданной карты городов и найденного маршрута;

Помимо консольной утилиты будет реализовано интерактивное приложение, которое позволит визуализировать работу конкретного метода. С помощью него можно будет визуализировать:

* метод ближайшего соседа;
* метод остовного дерева с дальнейшим выделением эйлерова цикла;
* локальный поиск 2-opt.

В этом приложении также можно будет создавать карту городов с помощью пользовательского интерфейса.

**Тестирование**

Будет проведено два набора тестов, каждый из которых будет состоять из нескольких тестов, а полученные результаты будут усредняться:

1. *1 <= N <= 20.* На каждом тесте этого набора будет запущено экспоненциальное решение, которое гарантированно находит оптимальный ответ. Это поможет показать погрешность, которую показывают различные приближенные решения;
2. *100 <= N <= 1000.* На тестах этого набора экспоненциальное решение, которое гарантированно ищет оптимальное решение, уже никак не сможет успеть отработать за разумное время, поэтому оно здесь использоваться не будет. Однако будут сравнены между собой все 10 приближенных решений по найденной длине маршрута и временных затрат на поиск.

Для тестирования будут использоваться различные случайные графы, полученные путем последовательной генерации каждого города с помощью равномерного генератора псевдослучайных чисел.

**Источники**:

1. [Постановка задач и ее частные случаи](http://synset.com/ai/ru/tsp/Salesman_Intro.html);
2. [Описание решений, дающих *2* и *1,5* приближения исходной задачи](http://rain.ifmo.ru/cat/data/theory/unsorted/approx-algo-2010/article.pdf);
3. [Методы локальных оптимизаций улучшения стартовых решений](http://www.math.nsc.ru/LBRT/k5/TPR_MMF-2013(2)/TSPr.pdf)