«Задача коммивояжера»

Анализ существующих алгоритмов, подходов к решению данной задачи

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

Т.к. задача коммивояжера является *NP*-трудной, то все методы ее решения можно разделить на два класса:

1. точные методы – которые гарантированно дают правильный ответ при достаточно небольших значениях *N;*
2. приближенные методы – которые находят определенное приближение исходной задачи (т.е. ответ, доказуемо хуже не более, чем в *x* раз), зато могут быть применены к большим графам.

Точные методы

Так или иначе, все точные методы основываются на полном переборе всех возможных вариантов. На сегодняшний день известны следующие точные методы решения данной задачи:

1. метод полного перебора, которые заключается в полном переборе всех возможных маршрутов и выборе самого лучшего из них. Такой метод, очевидно, работает за *O(N!)*, поэтому может быть применен для совсем маленьких графов, где *N <= 12.*
2. решение с помощью динамического программирования по маскам. *dp[mask][last] –* значение минимальной длины такого пути, которые проходит через все города, упомянутые маской *mask*, и заканчивающегося в городе *last.* Такое решение работает уже за *O(2NN2),* поэтому может быть применено уже для чуть большего спектра графов, а именно *N <= 25.*
3. метод ветвей и границ. Этот метод также заключается в переборе всех маршрутов, однако, куда более умном переборе, в сравнении с тем же первым методом. Основная идея заключается в отбрасывании решений, которые гарантированно не смогут быть оптимальными. Это делается с помощью сравнения нижней оценки решения, которое может быть получено из текущего, с найденным оптимальным решением на данный момент. Например, если мы зафиксировали какую-то часть пути, длина которой уже больше оптимального ответ, то очевидно, что пытаться продолжать такой стартовый путь просто не имеет смысла. Т.к. данный метод является перебором с отсечениями, то точную асимптотику времени работы для него указать нельзя, однако на практике такой метод работает еще для чуть больших графов, а именно графов, в которых *N <= 50.*

В этом проекте мы решили больше сосредоточиться на изучении приближенных методов, т.к. на практике чаще приходиться работать именно с большими графами, в которых уже никакой точный метод сработать не сможет. Поэтому в качестве точных решений, мы рассмотрим только первые два – полный перебор всех вариантов и метод динамического программирования.

Приближенные методы

Во всех приближенных решениях можно выделить два этапа:

* нахождение «стартового» маршрута;
* улучшение стартового маршрута методом локальных оптимизаций.

На сегодняшний день известно множество способов нахождения стартового маршрута. А именно:

1. **Случайный** метод – 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. **Битонический коммивояжер**. Т.к. поиск оптимального маршрута среди всех является *NP-*трудной задачей, можно попытаться как-то ограничить множество маршрутов, среди которых мы ищем оптимум. Одним способом это сделать – является рассмотрение только «битонических» путей. «Битонический» путь – это путь, который сначала идет слева направо (нижняя часть пути), а затем справа налево (верхняя часть). Примечательно то, что оптимальный «битонический» путь уже можно найти за полиномиальное время. Сделать это можно за время *O(N2)* с помощью динамического программирования;
4. **Метод минимального остовного дерева**. Этот метод заключается в выделении минимального остовного дерево в нашем графе, и дальнейшем рассмотрении эйлерова цикла в этом дереве. В полученном цикле оставляется только первое вхождение каждой вершины и полученный замкнутый путь возвращается в качестве искомого. Примечательно, что этот метод дает неплохое приближение исходной задачи, а именно его результат будет не больше, чем в 2 раза хуже оптимального;
5. **Метод минимального паросочетания.** Этот метод является небольшой модификацией предыдущего. Он все также находит минимальное остовное дерево, однако делает граф эйлеровым чуть более «умным» способом. Для этого он вместо раздвоения каждого ребра находит совершенное паросочетание на нечетных вершинах. Оказывается, что такой метод дает *1.5-приближение* исходной задачи;
6. **“Антижадный” алгоритм.** Идея "антижадного" алгоритма заключается в том, что из графа последовательно удаляются ребра наибольшей длины при одновременном соблюдении двух правил: в текущем графе, во-первых, из каждой вершины должно выходить, как минимум, два ребра; во-вторых, не должно образовываться циклов менее, чем из *n* ребер. Легко понять, что с помощью такого удаления мы в конце концов оставим только ребра, образующие замкнутый цикл, а т.к. на каждом шаге мы старались удалять максимальное по весу ребро, то ребра в этом пути в целом будут иметь небольшую длину;
7. **«Муравьиный» алгоритм.** Он заключается в эмуляции поведения муравьиной колонии. А именно вводится вероятность перехода каждого муравья из одного в города другой и в дальнейшем ищется наиболее вероятный замкнутый путь колонии муравьев.

В данном проекте мы решили сосредоточиться на 5 алгоритмах из вышеперечисленных, а именно *случайном методе*, *методе ближайшего соседа, битоническом коммивояжере, методе минимального остовного дерева* и *методе минимального паросочетания.*

Теперь перейдем к методам улучшения стартового пути, также известным под названием «методы локальных оптимизаций»:

1. **2-opt**. Метод заключается в выполнении последовательных итераций, на каждой из которых выбирается пара ребер *a => b* и *c => d,* которая заменяется на пару *a => c* и *b => d.* При этом пара выбирается таким образом, чтобы улучшение результата было максимально возможным;
2. **k-opt**. Является небольшой модификацией предыдущего метода, а именно на каждой итерации теперь выбирается не 2 ребра и заменяется на 2 других, а *k* ребер на *k* других.
3. **Генетический алгоритм**. Основной для него служит уже не один стартовый маршрут, а сразу несколько, которые в совокупности называются популяцией. Далее происходит последовательность итераций, на каждой из которых выбираются два маршрута, которые скрещиваются между собой, получая тем самым более выгодный маршрут. Самый слабый маршрут при этом *погибает* в конце итерации.

В данном проекте мы сосредоточимся на первых двух методах локального поиска.

Широта охвата предметной области

Подводя небольшой итог, хочется сказать, что мы охватили 60% точных методов, 69% методов выбора стартового маршрута и 55% методов локального поиска (здесь делается поправка на то, что возможно, не все существующие на данный момент алгоритмы были упомянуты в данном документе). Таким образом, общая широта охвата – 61%, что довольно неплохо, ведь данная задача изучается уже более 200 лет, и за все это время было придумано множество различных подходов к ее решению.