ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ОТЧЕТ

по проекту

по дисциплине «Программирование»

Студент гр. 151 _____ Коротченок О.А.

Преподаватель _____ Гориховский В.И.

Санкт-Петербург
2021

Тема Проекта

В проекте описаны 7 методов решения задачи коммивояжера – известной NP-сложной задачи – рассматриваемой в такой трактовке:

- Дан полный ориентированный граф из n вершин в виде его матрицы смежности $n \times n$, состоящей из n натуральных чисел, помещающихся в int (c++). На главной диагонали стоят максимальные в int числа (аналог бесконечности).
- Требуется найти минимальный по размеру гамильтонов цикл, т.е. путь, который проходит по каждой вершине ровно один раз и имеет минимальную возможную суммарную длину ребер.

Решить задачу коммивояжера значит найти оптимальный цикл или цикл, приближенный к оптимальному.

Алгоритмы

1. **Полный перебор.**

Рассматриваются все возможные пути, генерируемые рекурсивно, из них выбирается наилучший – он будет как раз оптимальным. Асимптотическая сложность: $O(n!)$.

2. **Метод имитации отжига.**

Случайным образом генерируется первый путь. Далее по количеству итераций происходит генерация транспозиция двух элементов а затем анализ: если она улучшает результат, то она применяется, если она ухудшает результат, то она применяется с какой-то вероятностью. Асимптотическая сложность зависит от подстановки констант, в этом проекте - $O(n^2)$. Ответ дает приближенный, иногда оптимальный.

3. **Динамический алгоритм.**

Каждый путь представляется в виде битовой маски, где 1 стоит у городов, которые мы уже посетили, и 0 у тех, которые еще не посетили, и номера города, который был посещен последним. Хранится заполненный на первом этапе это в двумерном массиве $2^n \times n$, элемент с индексами [0][0] равен единице. Затем динамикой вверх для каждого пути определяем, насколько выгодно из него ехать в какой-то другой. Таким образом динамикой вверх мы получим оптимальный путь. Асимптотическая сложность: $O(n^2 2^n)$, также нужно хранить массив размера $2^n \times n$.

4. **Метод ветвей и границ.**

Сначала находится верхняя граница – первый “лучший” путь. В проекте для этого используется адаптированный алгоритм Краскала. Затем происходит полный перебор, но для каждого пути на каждом этапе определяется, имеет ли смысл рассматривать далее по двум параметрам: 1) Если сумма пройденных ребер больше текущего “лучшего” пути, то этот путь больше не рассматривается. 2) Если сумма пройденных ребер плюс размер минимального

остовного дерева на непройденных вершинах больше “лучшего” пути, то этот путь больше не рассматривается. Когда путь будет закончен, он сравнивается с текущим “лучшим” и минимальных из них становится новым “лучшим”. В итоге получаем оптимальный путь. У данного метода нет асимптотической сложности, так как она сильно зависит от входных данных. В лучшем случае это $O(n^3)$, в худшем - $O(n!)$.

5. Метод ближайшего соседа.

На каждом этапе в качестве следующей вершины выбирается ближайшая к текущей. Алгоритм имеет все плюсы и минусы группы жадных алгоритмов: работает быстро (тут – линейно), но может давать ответы, сильно далекие от оптимального.

6. Генетический алгоритм.

За основу алгоритма взято поведение хромосом в популяции. Изначально генерируется случайные пути. Их количество – размер популяции, который не будет меняться. На каждом этапе цикла будет случайным образом выбираться 2 хромосомы из популяции, между ними произойдет обмен участками, затем с детьми с какой-то вероятностью произойдет мутация – транспозиция двух элементов. Дети будут добавлены в популяцию, она сортируется по возрастанию и из нее удаляются две последних хромосомы. Асимптотическая сложность зависит от заданной константы – количества итераций. В этом проекте это $O(n^3)$. Результат получается приближенный к оптимальному.

7. Муравьиный алгоритм.

На каждой вершине расставляется по одному муравью. На каждом этапе цикла сначала все муравьи по очереди независимо путешествуют по графу, оставляя след из феромонов. На основе феромонов и длинны ребра каждый муравей выбирает следующую вершину на своем пути. После того, как все муравьи прошли, для каждого ребра считается его новый показатель количества феромонов с учетом испарения и прошедших муравьев. В итоге выбирается путь, по которому прошло больше всего муравьев. Асимптотическая сложность зависит от заданной константы – количества итераций. В этом проекте это $O(n^4)$. Результат получается приближенный к оптимальному.

Тестовые входные данные

1. **Матрица** 4×4

2147483647	1	2	2
2	2147483647	1	2
2	2	2147483647	1
100000	2	2	2147483647

Программа выдает:

```
Brute Force:
Root:1 -> 2 -> 4 -> 3 -> 1
Distance:7
Time:0

Closest Neighbour:
Root:1 -> 2 -> 3 -> 4 -> 1
Distance:100003
Time:0

Simulated Annealing:
Root:1 -> 3 -> 4 -> 2 -> 1
Distance:7
Time:0.007

Branch and Bound:
Root:1 -> 4 -> 2 -> 3 -> 1
Distance:7
Time:0.001

Genetic Algorithm:
Root:1 -> 3 -> 4 -> 2 -> 1
Distance:7
Time:0.47

Dynamic Algorithm:
Root:1 -> 3 -> 4 -> 2 -> 1
Distance:7
Time:0

Ant Colony Algorithm:
Root:1 -> 4 -> 3 -> 2 -> 1
Distance:8
Time:0.052
```

Все алгоритмы за исключением ближайшего соседа дают верный ответ – 7. Ближайший сосед был обманут – выбирая наименьшие ребра он в итоге пришел к ребру, которое не стоит брать - 10000, но выбора у него уже нет, так как остальные вершины он уже прошел.

2. **Матрица** 10×10

В качестве входных данных дается матрица смежности для графа, являющегося правильным 10-угольником с ребром равным 5. В такой ситуации мы заранее знаем, что оптимальный путь равен 50.

Программа выдает:

```
Brute Force:
Route:1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10 -> 1
Distance:50
Time:57.828

Closest Neighbour:
Route:1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10 -> 1
Distance:50
Time:0

Simulated Annealing:
Route:1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10 -> 1
Distance:50
Time:0.02

Branch and Bound:
Route:1 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1
Distance:50
Time:0.08

Genetic Algorithm:
Route:1 -> 10 -> 8 -> 7 -> 9 -> 6 -> 4 -> 5 -> 3 -> 2 -> 1
Distance:78
Time:2.393

Dynamic Algorithm:
Route:1 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1
Distance:50
Time:0.061

Ant Colony Algorithm:
Route:1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10 -> 1
Distance:50
Time:0.936
```

Тут надо обратить внимание на время работы алгоритмов. В то время как остальные алгоритмы работают за адекватное время, полный перебор уже работает минуту. Следовательно, на 11 вершинах он уже будет работать одиннадцать минут. Таким образом, на больших числах его уже использовать не имеет смысла из-за неэффективности по времени.

3. **Матрица** 19 × 19

В качестве входных данных дается матрица смежности для графа, являющегося правильным 19-угольником с ребром равным 5. В такой ситуации мы заранее знаем, что оптимальный путь равен 95.

Программа выдает:

```
Closest Neighbour:
Route:1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10 -> 11 -> 12 -> 13 -> 14 -> 15 -> 16 -> 17 -> 18 -> 19 -> 1
Distance:95
Time:0.001

Simulated Annealing:
Route:1 -> 8 -> 9 -> 10 -> 11 -> 12 -> 13 -> 14 -> 19 -> 18 -> 17 -> 16 -> 15 -> 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1
Distance:159
Time:0.073

Branch and Bound:
Route:1 -> 19 -> 18 -> 17 -> 16 -> 15 -> 14 -> 13 -> 12 -> 11 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1
Distance:95
Time:1.366

Genetic Algorithm:
Route:1 -> 2 -> 7 -> 8 -> 12 -> 13 -> 15 -> 9 -> 6 -> 3 -> 17 -> 18 -> 19 -> 14 -> 5 -> 4 -> 11 -> 16 -> 10 -> 1
Distance:313
Time:7.929

Ant Colony Algorithm:
Route:1 -> 4 -> 3 -> 2 -> 5 -> 6 -> 8 -> 7 -> 9 -> 10 -> 11 -> 12 -> 13 -> 14 -> 15 -> 16 -> 17 -> 18 -> 19 -> 1
Distance:123
Time:11.848

Dynamic Algorithm:
Route:1 -> 19 -> 18 -> 17 -> 16 -> 15 -> 14 -> 13 -> 12 -> 11 -> 10 -> 9 -> 8 -> 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1
Distance:95
Time:66.983
```

Здесь проходит граница для динамического алгоритма: при n = 20 больше у меня код не компилируется из-за ошибок с памятью, так что дальше не имеет смысл включать динамический.

4. Матрица 40×40

В качестве входных данных дается матрица смежности для графа, являющегося правильным 40-угольником с ребром равным 20. В такой ситуации мы заранее знаем, что оптимальный путь равен 800.

Программа выдает:

```
Closest Neighbour:
Route:1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10 -> 11 -> 12 -> 13 ->
Distance:800
Time:0.001

Simulated Annealing:
Route:1 -> 3 -> 8 -> 10 -> 9 -> 7 -> 6 -> 5 -> 4 -> 15 -> 19 -> 20 -> 21 ->
Distance:2091
Time:0.502

Branch and Bound:
Route:1 -> 40 -> 39 -> 38 -> 37 -> 36 -> 35 -> 34 -> 33 -> 32 -> 31 -> 30 -
Distance:800
Time:40.514

Genetic Algorithm:
Route:1 -> 19 -> 15 -> 40 -> 4 -> 30 -> 31 -> 27 -> 22 -> 18 -> 23 -> 34 ->
Distance:4972
Time:41.013

Ant Colony Algorithm:
Route:1 -> 40 -> 39 -> 37 -> 38 -> 34 -> 35 -> 36 -> 31 -> 32 -> 33 -> 27 -
Distance:1113
Time:378.543
```

Тут поподробнее о каждом результате.

Ближайший сосед дает оптимальный путь по той причине, что это очень приятный для него пример – тут брать наименьшее ребро всегда работает, так как это и будет ребро 40-угольника.

Метод имитации ограничен своими константами и тем, что он не всегда выбирается из локального минимума, причем часто для решения этой проблемы требуется слишком много невыгодных поначалу транспозиций.

Ветви и границы дают (как всегда) оптимальный путь, но уже сказываются проблемы с неопределенной асимптотической сложностью, однако по времени результат все еще приемлемый.

Генетический алгоритм меня, если честно, сильно разочаровал. Время работы адекватное, но ответ слишком сильно отличается от оптимального. Сказывается то, что константы задаются по одинаковой логике или просто одинаковые для разных входных данных. Может быть стоит поиграться с процентом мутаций, может позапускать чаще, может стоит менять размер популяции. Но вывод такой: на больших числах этот генетический алгоритм работает плохо.

Муравьи работаю долговато, но зато ответ очень даже приличный. Там тоже можно поиграть с константами – увеличение приоритета длины пути над количеством феромонов на данном примере может даже привести к тому, что он будет давать оптимальный результат.