«Задача коммивояжера»

Экспериментальная часть

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

Тестирование проводилось на двух глобально различных группах тестов:

1. тесты с маленьким количеством городов (*1 <= N <= 20).* В этой группе была возможность запустить экспоненциальное решение для дальнейшего сравнения результатов всех остальных результатов с глобальным оптимумом. Т.к. все используемые алгоритмы являются полиномиальными, то на тестах этой группы они выполнялись мгновенно, поэтому здесь сравнивалось только качество работы алгоритмов;
2. тесты с большим количеством городов (*100 <= N <= 1000).* В этой группе уже не было практической возможности найти гарантированно оптимальный маршрут, поэтому сравнивались только приближенные алгоритмы между собой. Их качество считалось, как качество относительно алгоритма, показавшего лучший результат на каждом конкретном тесте. Также, из-за того, что количество городов было уже довольно существенным, алгоритмов сравнивались не только по качеству работы, но и по времени.

Для тестирования использовался генератор, принимающий в качестве параметров два значения *N* – количество городов, и *MX –* максимальное значение координат каждого города. Сразу отметим, что при больших *N* использовалось большее значение *MX*, чтобы множество городов, выбираемых генератором было как можно более случайно. Также отметим, что никакой четкой зависимости между *N* и *MX* не было, т.к. уже при *MX* *>=* *N* множество городов будет сгенерировано довольно хорошим образом.

Также для каждой выбранной пары *N* и *MX* проводилось *Tests* тестов, а итоговые результаты усреднялись. Это было сделано во избежание случайных «везений» тех или иных алгоритмов, т.к. на длинной дистанции будут выявлены более общие тенденции касательно качества алгоритмов.

Результаты на маленьких тестах

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | MX | Tests | random | bitonic | closest neighbor | 2-approximation | 1.5-approximation |
| 9 | 100 | 50 | 1.41 | 1.03 | 1.08 | 1.15 | 1.07 |
| 12 | 100 | 50 | 1.64 | 1.08 | 1.14 | 1.2 | 1.12 |
| 15 | 100 | 50 | 1.89 | 1.09 | 1.15 | 1.22 | 1.11 |
| 18 | 1000 | 25 | 2.09 | 1.15 | 1.14 | 1.25 | 1.1 |
| 20 | 1000 | 25 | 2.32 | 1.16 | 1.16 | 1.27 | 1.12 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | MX | Tests | random 2 opt | bitonic 2 opt | closest neighbor 2 opt | 2-approximation 2 opt | 1.5-approximation 2 opt |
| 9 | 100 | 50 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| 12 | 100 | 50 | 1.01 | 1.01 | 1.0 | 1.02 | 1.01 |
| 15 | 100 | 50 | 1.02 | 1.01 | 1.01 | 1.02 | 1.01 |
| 18 | 1000 | 25 | 1.02 | 1.02 | 1.03 | 1.02 | 1.02 |
| 20 | 1000 | 25 | 1.03 | 1.02 | 1.01 | 1.03 | 1.01 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | MX | Tests | random 3 opt | bitonic 3 opt | closest neighbor 3 opt | 2-approximation 3 opt | 1.5-approximation 3 opt |
| 9 | 100 | 50 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| 12 | 100 | 50 | 1.01 | 1.01 | 1.0 | 1.01 | 1.01 |
| 15 | 100 | 50 | 1.01 | 1.01 | 1.01 | 1.01 | 1.00 |
| 18 | 1000 | 25 | 1.02 | 1.01 | 1.02 | 1.02 | 1.02 |
| 20 | 1000 | 25 | 1.02 | 1.02 | 1.01 | 1.02 | 1.01 |

Результаты на больших тестах

Здесь уже помимо среднего качества работы также упомянуто среднее время работы на всех тестах.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | MX | Tests | random | bitonic | closest neighbor | 2-approximation | 1.5-approximation |
| 100 | 4000 | 10 | 5.79 | 2.05 | 1.22 | 1.37 | 1.14 |
| 0.01 sec | 0.01 sec | 0.01 sec | 0.01 sec | 0.01 sec |
| 200 | 5000 | 10 | 8.58 | 2.88 | 1.2 | 1.33 | 1.14 |
| 0.01 sec | 0.01 sec | 0.01 sec | 0.01 sec | 0.01 sec |
| 400 | 5000 | 5 | 12.61 | 4.09 | 1.2 | 1.33 | 1.13 |
| 0.02 sec | 0.02 sec | 0.02 sec | 0.01 sec | 0.01 sec |
| 1000 | 10000 | 20 | 20.33 | 6.2 | 1.2 | 1.33 | 1.14 |
| 0.08 sec | 0.12 sec | 0.03 sec | 0.03 sec | 0.05 sec |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | MX | Tests | random 2 opt | bitonic 2 opt | closest neighbor 2 opt | 2-approximation 2 opt | 1.5-approximation 2 opt |
| 100 | 4000 | 10 | 1.08 | 1.08 | 1.04 | 1.06 | 1.01 |
| 0.02 sec | 0.01 sec | 0.01 sec | 0.02 sec | 0.01 sec |
| 200 | 5000 | 10 | 1.07 | 1.07 | 1.03 | 1.04 | 1.01 |
| 0.03 sec | 0.03 sec | 0.02 sec | 0.02 sec | 0.04 sec |
| 400 | 5000 | 5 | 1.08 | 1.07 | 1.03 | 1.03 | 1.01 |
| 0.09 sec | 0.07 sec | 0.05 sec | 0.06 sec | 0.05 sec |
| 1000 | 10000 | 20 | 1.08 | 1.08 | 1.01 | 1.04 | 1.01 |
| 0.55 sec | 0.42 sec | 0.34 sec | 0.32 sec | 0.29 sec |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | MX | Tests | random 3 opt | bitonic 3 opt | closest neighbor 3 opt | 2-approximation 3 opt | 1.5-approximation 3 opt |
| 100 | 4000 | 10 | 1.05 | 1.06 | 1.02 | 1.05 | 1.01 |
| 0.17 sec | 0.14 sec | 0.12 sec | 0.17 sec | 0.09 sec |
| 200 | 5000 | 10 | 1.06 | 1.06 | 1.01 | 1.02 | 1.01 |
| 1.44 sec | 1.22 sec | 1.1 sec | 1.19 sec | 0.87 sec |
| 400 | 5000 | 5 | 1.05 | 1.05 | 1.01 | 1.02 | 1.0 |
| 12.42 sec | 11.37 sec | 7.51 sec | 8.65 sec | 8.08 sec |
| 1000 | 10000 | 20 | 1.06 | 1.07 | 1.01 | 1.02 | 1.0 |
| 209.93 sec | 188.92 sec | 155.23 sec | 154.96 sec | 156.97 sec |

Анализ полученных результатов

Анализируя полученные результаты, можно сделать вывод, что бо́льшую роль на итоговую длину маршрута оказывать метод локальных оптимизаций, нежели выбор стартового маршрута. Так, например, метод *3-opt*, примененный к самому слабому алгоритму *random* (который сам по себе показывает качество 20.33 при *N* = 1000), позволяет довести его до качества 1.06.

Хотя, безусловно, выбор стартового маршрута также влияет на итоговый результат. В нашей выборке алгоритмов, их результат колеблется в диапазоне 7%.

Если расставлять алгоритмы стартового маршрута от самого худшего к самому лучшему, то получится такая последовательность: *random, bitonic, closest neighbor, 2-approximation, 3-approximation.*

Если сравнить алгоритмы локальной оптимизации, то можно сказать, что алгоритм *3-opt* показывает чуть лучшие результаты, нежели *2-opt,* примерно на *1-3%.* Однако, это стоит ему куда более длительного времени работы. Например, при *N = 1000,* ему требуется около двух с половиной минут, в то время как *2-opt* отрабатывает в течении секунды. Это говорит о том, что начиная примерно с *N = 1000,* стоит использовать метод *2-opt*, т.к. особой потери качества не будет, зато будет существенный выигрыш в скорости.