Travelling Salesman Problem Report

Conner Weatherston *
Edinburgh Napier University
Algorithms and Data Structures (SET09117)

Abstract

PLACE ABSTRACT HERE. This report is aimed at finding a suitable algorithm in order to solve the travelling salesman problem.

Keywords: travelling salesman problem, nearest neighbour, optimisation, algorithms

1 Introduction

This report is looking at possible improvements of the nearest neighbour algorithm in order to solve the travelling salesman problem (commonly referred to as tsp). The tsp essentially is a question which asks 'Given a set of cities, what is the shortest route possible that visits each city only once and will return to the origin.

The main issue with the travelling salesman problem is the possible routes available grows exponential with size. In a tsp with 10 cities there is 181,400 possible routes. This is calculated using the formula

$$P = \frac{(N-1)!}{2} \tag{1}$$

where P is number of possibilities and N is number of cities (points).

Only half of the possible routes are counted as each route has an equal reverse route that has the exact same distance. The P-1 is there since the starting city is defined and the other cities can have different permutations.

2 Method

One possible way of computing a tsp is by using the nearest neighbour algorithm. This involves sorting the list so that the next city in the list is the closest city to the current city.

Pros

- Easy to implement.
- Very quick results for small data sizes.
-

Cons

- Requires large storage of data.
- Large searching problems (Have to continually iterate over list until it is empty.)
- Assumptions are made about distance (Some routes may be infeasible).
- Brute force method.

Pseudocode

```
Data: ArrayList input
Result: returns Nearest Neighbour list
current city = input first value while cities in input do
    add current city to result distance = max value foreach city in
    input do
        if distance(current city, city) < distance then
            closest city = city distance = distance(current
            point,city)
        end
    end
    remove closest city from input current city = closest city
end
```

Algorithm 1: Nearest neighbour algorithm

A variation to the nearest neighbour algorithm is comparing the distance along an axis. In theory this should give a quicker computation time as it is only comparing two numbers and avoids using any squaring and square rooting.

As it is based on the nearest neighbour still there is very little difference in the algorithm.

```
Data: ArrayList input
Result: returns Nearest Neighbour list
current city = input first value while cities in input do
    add current city to result closest x = max value foreach city in
    input do
        if city.x < current city.x then
            closest city = city closest x = city.x
        end
    end
    remove closest city from input current city = closest city
end
```

Algorithm 2: Nearest x neighbour algorithm

```
Data: ArrayList input
Result: returns Nearest Neighbour list
current city = input first value while cities in input do
    add current city to result closest y = max value foreach city in
    input do
        if city.y < current city.y then
            closest city = city closest y = city.y
        end
    end
    remove closest city from input current city = closest city
end
```

Algorithm 3: Nearest y neighbour algorithm

3 Results

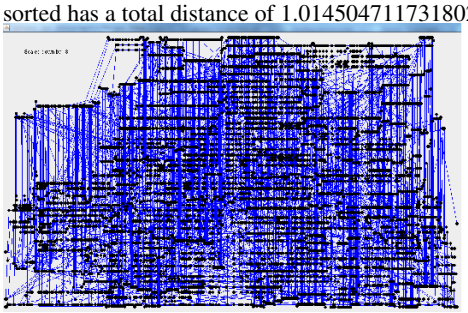
The results were obtained from the same machine to ensure consistency and to prevent inaccurate results. The specifications of the computer used to collect the results are as follows:

*e-mail:40167111@live.napier.ac.uk

- Processor: Name it.
- Graphics card: Name it.
- Ram:
- Hard drive.

Also while collecting the results extremes on both sides were removed in order to create a more accurate average. This was to take into account Java garbage collection.

Data set rl5915 hline rl5915 has a size of 5915 and when unsorted has a total distance of 1.0145047117318021E7.



Nearest Neighbour

Attempt	Total Distance (m)	Time Taken (ms)
1	707498.63	207
2	707498.63	188
3	707498.63	185
4	707498.63	186
5	707498.63	185
Average	707498.63	190

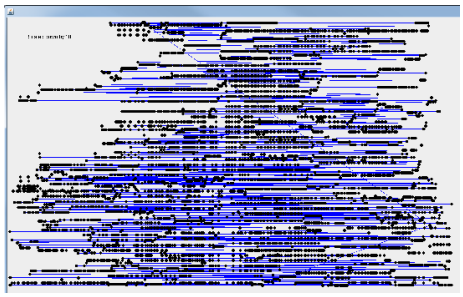
Nearest X Neighbour

Attempt	Total Distance	Time Taken
1	11C	22C
2	9C	19C
3	10C	21C
4	10C	21C
5	10C	21C
6	10C	21C
Average	10C	21C

IMAGE FOR SORTED ROUTE GOES HERE.

Nearest Y Neighbour

Attempt	Total Distance	Time Taken
1	7787705.43	98
2	7787705.43	80
3	7787705.43	76
4	7787705.43	77
5	7787705.43	76
Average	7787705.43	81



Due to the nature of the nearest neighbour algorithm some of the variations implemented would only result in a difference in time taken to compute the algorithm. However it is possible to compare the nearest neighbour with nearest x and nearest y as they produce different routes.

WHAT WAS THE FASTEST/ SLOWEST / BEST RESULT/ WORST RESULT FOR SMALLEST

WHAT WAS THE FASTEST/ SLOWEST / BEST RESULT/ WORST RESULT FOR LARGEST

WHAT WAS THE FASTEST/ SLOWEST / BEST RESULT/ WORST RESULT FOR AVERAGE

WHAT WAS SURPRISING.

WHAT IMPROVEMENTS CAN BE MADE.

Comparing to original nearest neighbour to nearest neighbout version 3 there is a notable increase in performance due to rewriting the algorithm slightly. This was done by increasing storage requirement and storing the distance of the current city and the city that the for loop was at. As the Point2D distance function uses multiplication and square rooting. This causes a hit to performance. So by storing the result of the function it means that the function is only performed once instead of the potential two times (the other time being if the distance is shorter than the last distance) in the for loop. To improve upon it more the square root was not necessary as it is only comparing distances between the two. So using the distanceSQ has a postive impact on the computation time on the algorithm.

The nearest neighbour is slower compared to the nearest x and nearest y algorithms. The reason for this is also due to the distance function. It is quicker to straight up compare two numbers like what is done in nearest x/y than calculating the distance between two points.

In version 2 the total distance varies so much due to starting at a random position. Due to this

4 Conclusion and Future work

The report should finish with a summary to give a brief overview of what the reader should remember most. What was most important? The future work part only needs to be covered in your final report.

References