Travelling Salesman Problem
Report

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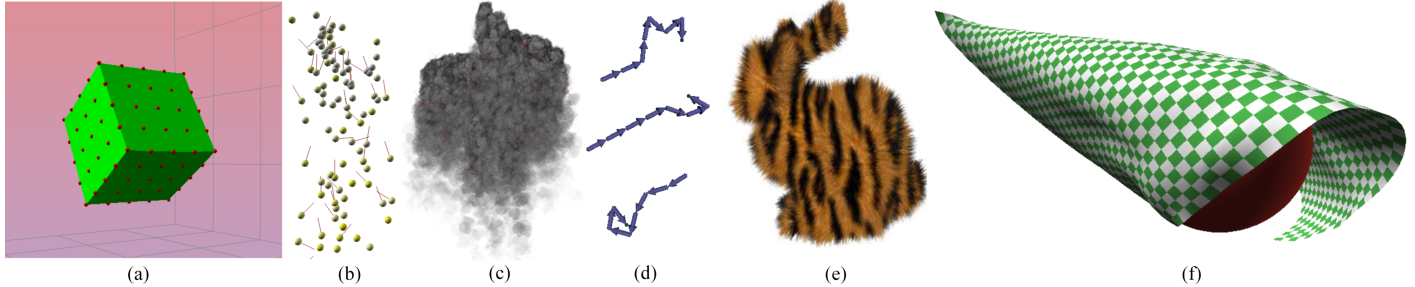Algorithms and Data Structures (SET09117)

Figure 1: *Place a teaser image at the top of your report to show key examples of your work (e.g., multiple screenshots of the different test situations) - Every figure should have a caption and a description. For example, each figure is labelled and explained: (a) soft bodies, (b) particles, (c) inverse kinematics, (e) fur shells, and (f) position-based dynamics for cloth effects.*

Abstract

PLACE ABSTRACT HERE.

Keywords: travelling salesman problem, nearest neighbour, optimisation

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.

---

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

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
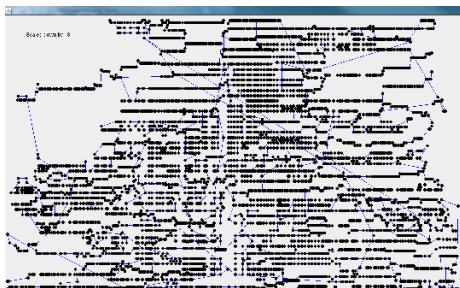
3 Results

Data set rl5915

rl5915 has a size of 5915 and when unsorted has a total distance of 1.0145047117318021E7.
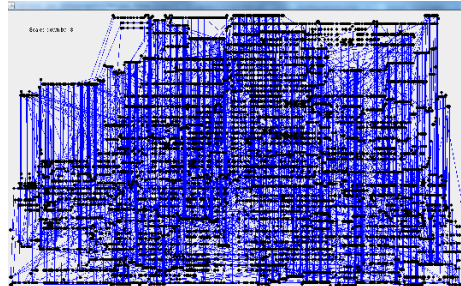
Nearest Neighbour

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
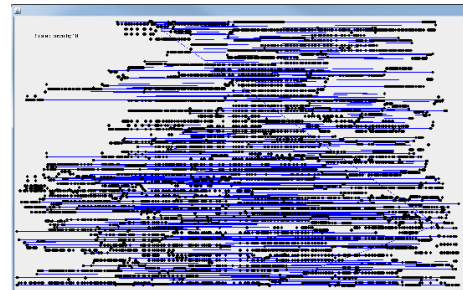
4 Conclusion and Future work

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