A Traveling Salesman Solution For The Capitals of All African Nations

Brian Gianforcaro
Department of Computer Science
Rochester Institute of Technology

November 1, 2009

Abstract

A Traveling Salesman Problem is the task of finding the shortest round trip path a traveling salesperson can take to visit each vertex of a given graph. They are usually implemented using a genetic algorithm. Our salesperson happens to be traveling to the capitals of every country in Africa that is a recognized member of the United Nations.

1 The Problem

1. Algeria - Algiers
2. Angola - Luanda
3. Benin - Porto-Novo
4. Botswana - Gaborone
5. Burkina Faso - Ouagadougou
6. Burundi - Bujumbura
7. Cameroon - Yaounde
8. Cape Verde - Praia
9. Central African Republic - Bangui
10. Chad - N'Djamena
11. Comoros - Moroni
12. Congo, Republic of the - Brazzaville
13. Congo, Democratic Republic of the - Kinshasa
14. Cote d'Ivoire - Yamoussoukro
15. Djibouti - Djibouti
16. Egypt - Cairo
17. Equatorial Guinea - Malabo
18. Eritrea - Asmara
19. Ethiopia - Addis Ababa
20. Gabon - Libreville
21. The Gambia - Banjul
22. Ghana - Accra
23. Guinea - Conakry
24. Guinea-Bissau - Bissau
25. Kenya - Nairobi
26. Lesotho - Maseru
27. Liberia - Monrovia
28. Libya - Tripoli
29. Madagascar - Antananarivo
30. Malawi - Lilongwe
31. Mali - Bamako
32. Mauritania - Nouakchott
33. Mauritius - Port Louis
34. Morocco - Rabat
35. Mozambique - Maputo
36. Namibia - Windhoek
37. Niger - Niamey
38. Nigeria - Abuja
39. Rwanda - Kigali
40. Senegal - Dakar
41. Seychelles - Victoria
42. Sierra Leone - Freetown
43. Somalia - Mogadishu
44. South Africa - Pretoria
45. Sudan - Khartoum
46. Swaziland - Mbabane
47. Tanzania - Dar es Salaam
48. Togo - Lome
49. Tunisia - Tunis
50. Uganda - Kampala
51. Zambia - Lusaka
52. Zimbabwe - Harare

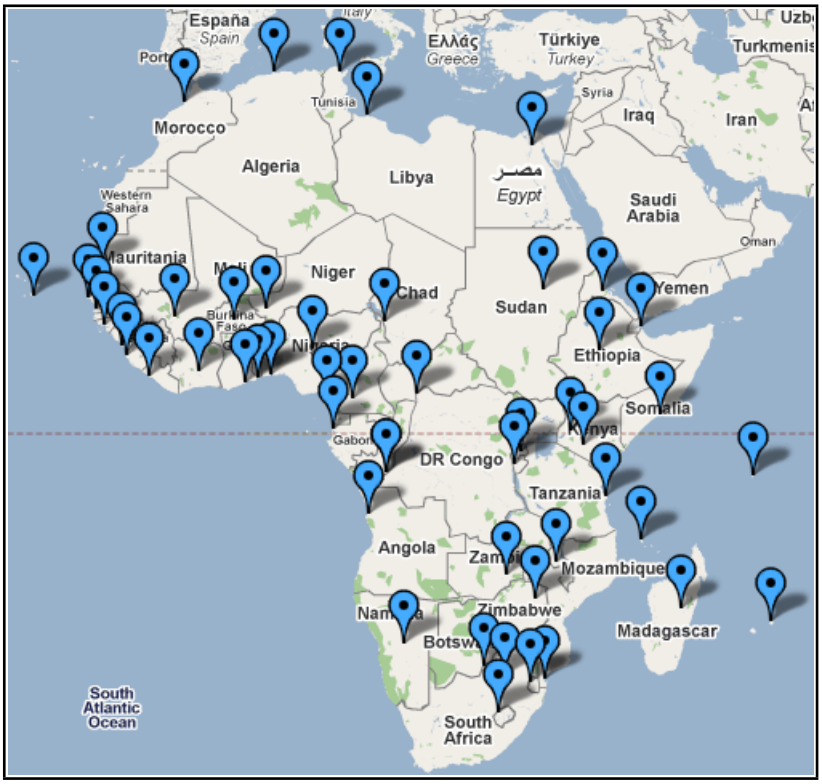

Figure 1: Capitals of African Nations

2 Overview

The traveling salesman problem (TSP) is a thoroughly researched problem in theoreticaly computer-science. TSP is classified as NP-Hard eg. as hard or harder to solve than a problem solvable in nondeterministic polynomial time. This means the worst case performance for a TSP algorithm will likely increase exponetially with the numbers of cities to traverse.

The big-O complexity of a brute force TSP algorithms (check all vertices against all other vertices) is $O(n!)$.

3 Programs

1. The Python 2.6 [1] programming language and interpreter.

2. Wikipedia's list of city latitude/longitudes [2].

3. Software originally written by John Montgomery [3] in 2007.General exercise in different types of TSP solving algorithms.

4. The results were then visualized using Google maps static mapping API [4].

4 Solution

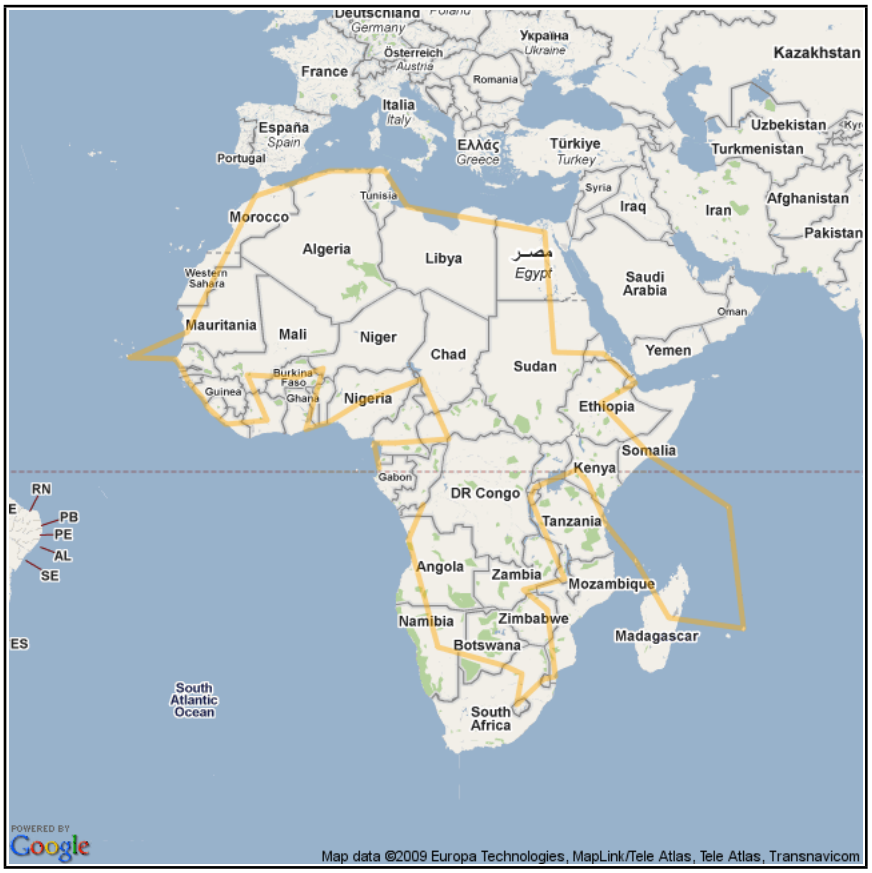

Figure 2: Final Path Through Africa

Final Order:

1. Congo, Republic of the - Brazzaville
2. Congo, Democratic Republic of the - Kinshasa
3. Angola - Luanda
4. Namibia - Windhoek
5. Botswana - Gaborone
6. South Africa - Pretoria
7. Lesotho - Maseru
8. Swaziland - Mbabane
9. Mozambique - Maputo
10. Zimbabwe - Harare
11. Zambia - Lusaka
12. Malawi - Lilongwe
13. Burundi - Bujumbura
14. Rwanda - Kigali
15. Uganda - Kampala
16. Kenya - Nairobi
17. Tanzania - Dar es Salaam
18. Comoros - Moroni
19. Madagascar - Antananarivo
20. Mauritius - Port Louis
21. Seychelles - Victoria
22. Somalia - Mogadishu
23. Ethiopia - Addis Ababa
24. Djibouti - Djibouti
25. Eritrea - Asmara
26. Sudan - Khartoum
27. Egypt - Cairo
28. Libya - Tripoli
29. Tunisia - Tunis
30. Algeria - Algiers
31. Morocco - Rabat
32. Mauritania - Nouakchott
33. Cape Verde - Praia
34. Senegal - Dakar
35. The Gambia - Banjul
36. Guinea-Bissau - Bissau
37. Guinea - Conakry
38. Sierra Leone - Freetown
39. Liberia - Monrovia
40. Cote d'Ivoire - Yamoussoukro
41. Mali - Bamako
42. Burkina Faso - Ouagadougou
43. Niger - Niamey
44. Ghana - Accra
45. Togo - Lome
46. Benin - Porto-Novo
47. Nigeria - Abuja
48. Chad - N'Djamena
49. Central African Republic - Bangui
50. Cameroon - Yaounde
51. Equatorial Guinea - Malabo
52. Gabon - Libreville

Total Round Trip $\approx$ 22253.2035387 miles

5 Runtime

The two algorithms, simulated annealing and brute force random permutations where both run in numerious instances over the life of the project. In combined total they were run upwards of fity times a peice.

The version of the algorithm wich used simulated annealing on average found the optimal solution in 6 seconds. While the bruteforce method often took upwards of

6 Analysis

I believe the final optimal path's are within a reasonable uncertenty of the actual path. Given the big-O of the brute force algorithm $O(n!)$ or $O(80658175170943878571660636856403766975289505440883277824000000000000)$ a runtime complexity of this magnitude is obviously out of my leauge for finding exact values. At a maxiumum I was able to attempt 100,000,000 permutations of the path.

Two area's of possible improvement I can pinpoint are:

1. The language itself. I chose the python programming language for my TSP implementation. The language is interpreted and it's runtime speed can at times be 20% slower than an equivilant C program.

2. The random.shuffle implementation. Given how python's random.shuffle API is implemented it talkes a set of values and shuffles them based

on a random number between 0,1 generated by the operating system. However given the size of the set , 52 individual points, the likely hood that random.shuffle will generated all permutations without excessive doubles is increadibly unlikely. This is the cause of a lot of literally useless computation time, however because of the size of the data set, caching data sets is not plausible let alone if even possible.

Re-writing the algorithm in C would be a great performance benifit. Also having a proper resources to cache routes or a effective random route generater I beleive a more accurate optimal route might be found.

References

[1] http://python.org

[2] http://en.wikipedia.org/wiki/Latitude_and_longitude_of_cities

[3] Montgomery, John *Tackling The Travelling Salesman Problem* http://www.psychicorigami.com/category/tsp/ , **2007**

[4] http://code.google.com/apis/maps/documentation/staticmaps/