EUROPEAN ACADEMIC RESEARCH
Vol. IV, Issue 3/ June 2016

Impact Factor: 3.4546 (UIF)
DRJI Value: 5.9 (B+)

ISSN 2286-4822
www.euacademic.org

Solving the Traveling Salesman Problem using Google Services and Simulated Annealing Algorithm

LAURIK HELSHANI[1]
PhD(c), European University of Tirana
Tirana, Albania

Abstract:

The concept of the traveling salesman problem is to seeking a tour of a specified number of cities (visiting each city exactly once and returning to the starting point) where the length of the tour is minimized. The mathematical modeling of this problem has to do with the theory of graphs.

In this paper, I examine simulated annealing algorithm in the context of solving the traveling salesman problem. Test-tool programming is done in JAVA programming language and his work is displayed graphically using the 2D library.

The whole system is programmed as client-server system using RESTFul web services, Google-services and Simulated Annealing Algorithm.

Google services are used to determine the optimum route on Google map and solve the Travelling Salesman problem.

Key words: Traveling Salesman Problem (TSP); Simulated Annealing Algorithem (SA); Google Services; RESTful; JAVA

[1] Laurik Helshani is PhD candidate at Faculty of Economics. Profile: Management Information Systems in European University of Tirana (UET). His PhD thesis: "Solving Travelling Salesman Problem using Genetic Algorithm and programming android smart-phones".

I. INTRODUCTION

The Travelling Salesman Problem is an optimization problem which has various applications such as: vehicle routing and scheduling, planning and logistics.

The Travelling Salesman Problem (TSP) is a classic combinatorial optimization problem, which is simple to state but very difficult to solve. This problem is known to be NP-hard, and cannot be solved exactly in polynomial time. [1]

This paper proposes an effective local search algorithm based on simulated annealing technique to solve the TSP.

Google map is used to provide the user an intuitive and highly responsive mapping interface, to help him by finding and marking the places he wants to visit. My web-based application and the results are displayed on Google maps.

II. RELATED WORK

Goossaert, in the article [8] presented simulated annealing, an optimization technique aimed to find an approximation to the global minimum of a function. He explained how this technique applies to the traveling salesman problem. After some tests, he managed to find a good set of parameters for the algorithm which is adapted to solve the problem. Finally, he applied the algorithm and he found an optimal path in a 50-city tour. The data he is using are GPS coordinates of 50 European cities. A 32% improvement is observed from the initial tour to the optimal tour. The author provided an implementation in Python, along with graphic visualization of the solutions.

In the article [9], Arostegui, Kadipasaoglu & Khumawala compared to the relative performance of Tabu Search (TS), Simulated Annealing (SA) and Genetic Algorithms (GA) on various types of Facility Location Problems (FLP) under time-limited, solution-limited, and unrestricted conditions. The results indicate that TS shows very good

performance in most cases. It was concluded that the performance of TS, SA and GA for the different types of FLP is situational principally based on the measure of comparability.

III. MATHEMATICAL FORMULATIONS OF SHORTEST TSP

The TSP can be defined on a complete undirected graph $G = (V, E)$ if it is symmetric or on a directed graph $G = (V, A)$ if it is asymmetric. The set V ={1, . . . , n} is the vertex set, $E = \{(i, j): i, j \in V, i < j\}$ is an edge set and $A = \{(i, j): i, j \in V, i \neq j\}$ is an arc set. A cost matrix $C = (c_{ij})$ is defined on E or on A. The cost matrix satisfies the triangle inequality whenever $c_{ij} \leq c_{ik} + c_{kj}$, for all i, j, k. In particular, this is the case of planar problems for which the vertices are points $P_i = (X_i, Y_i)$ in the plane, and

$$c_{ij} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \qquad (1)$$

is the Euclidean distance. The triangle inequality is also satisfied if c_{ij} is the length of a shortest path from i to j on G. [2]

A. Graph theorical formulation

Find the shortest Hamiltonian circuit in a complete graph where the nodes represent cities. The weights on the edges represent the distance between cities. The cost of the tour is the total distance covered in traversing all cities. [3]

B. Cost function

If the problem consists of n cities c_i, i = 1 … n, any tour can be represented as a permutation of numbers 1 to n. [3]

$$d(c_i, c_j) = d(c_j, c_i) \qquad (2)$$

is the distance between c_i and c_j.

Given a permutation π of the n cities, v_i and v_{i+1} are adjacent cities in the permutation. The permutation π has to be found that minimizes [3]:

$$\sum_{i=1}^{n-1} d(v_i, v_{i+1}) + d(v_n + v_1) \quad (3)$$

The size of the solution space is (n-1)!/2 [3]

IV. GOOGLE SERVICES

You have to add Map API key to your application to access the Google maps server with the maps API. Register in the Google APIs Console to get a Map API key.

A. Google maps

Google maps provide an intuitive and highly responsive mapping interface with aerial imagery and detailed street data. In addition, map controls and overlays can be added to the map so that users can have full control over map navigation. Map panning can also be performed by dragging the map via the mouse or by using "arrow" keys on a keyboard. Google maps can be customized according to application specific needs. [4].

B. Geo- coding

GeoCoder is a class for handling geo-coding and reverse geo-coding. Geo-coding is a process of converting addresses into geographical coordinates (latitudes and longitudes). Reverse geo-coding is the process of transforming (latitude, longitude) into addresses. [4]

C. Over layers

Overlays are objects on the map that are bound to latitude/longitude coordinates. Google Maps has several types of overlays:

1) *Marker - Single locations on a map. They are used to identify locations on the map.*

2) Polyline - Series of straight lines on a map.

D. Directions

The Google Maps Directions API is a service that calculates directions between locations using an HTTP request.

E. Google Places

The Google Places API Web Service is a service that returns information about places — defined within this API as establishments, geographic locations, or prominent points of interest — using HTTP requests. Place Searches return a list of places based on a user's location or search string.

V. SIMULATED ANNEALING ALGORITHM (SA)

A. Strategy

The information processing objective of the technique is to locate the minimum cost configuration in the search space. The algorithms plan of action is to probabilistically re-sample the problem space where the acceptance of new samples into the currently held sample is managed by a probabilistic function that becomes more discerning of the cost of samples it accepts over the execution time of the algorithm. [5]

B. Working of SA

In simulated annealing algorithm, the problem of finding an optimal solution is analogous to a process of reducing the overall “energy" of a system via state transitions. In the traveling salesman problem, a state is defined as a route, or a permutation of the cities to be visited. The energy of a state is the total distance of a route. In every iteration of the algorithm, a candidate state transition, also known as a neighbor of the current state, is generated by exchanging. The neighbor route is then evaluated by an acceptance function that makes the state transition if the neighbor route has a lower energy or chosen

with a probability that depends on the difference between the energies and on a global parameter T (called the temperature). [6]

C. Testing and result interpretation

Test-tool programming is done in JAVA programming language and his work is displayed graphically using the 2D library of JAVA.

The tool simulates graphically, what the algorithm is doing, generating random points in the XY-plane that symbolize places to visit, and draws straight lines between points that symbolize the roads that lead to these places.

The algorithm is tested on computer with these characteristics:

Operation System: Windows 7 Ultimate, 32-bit
CPU: Intel Core 2 Duo, 1.83 GHz
RAM: 4 GB

#places	Cooling rate	The initial length of the route	The length of the route after optimization	Optimization in %	Runtime in sec.
30	$1*e^{-4}$	14.9	4.3	345.360	0.04
30	$1*e^{-5}$	15.2	4.7	323.509	0.335
50	$1*e^{-5}$	26.1	6.4	405.615	0.441
50	$1*e^{-6}$	28.1	6.1	459.755	2.974
100	$1*e^{-4}$	52.5	9.9	530.889	0.167
100	$1*e^{-5}$	56.3	8.2	690.034	0.656
150	$1*e^{-4}$	83.9	15.4	543.696	0.069
150	$1*e^{-5}$	82.2	11.0	747.626	0.915
150	$1*e^{-6}$	82.6	10.2	808.325	6.635
150	$1*e^{-7}$	80.1	10.5	763.005	68.051

Tastcases oF SA- Algorithm

As seen from the above table, there are two key variables, which effects directly to the performance of the simulated annealing algorithm work, to its termination and to the final result (optimization of the route of travelling salesman). These variables are: Number of places to visit (#places) and temperature step (The degree of cooling temperature). The

other four columns: The initial length of the route, the length of the route after optimization, Optimization expressed as a percentage and runtime in seconds are results of the tool.

Optimization reaches the culmination (~808.325%), if the temperature decreases with $1*e^{-6}$ cooling rate and it happens in 6.635 seconds.

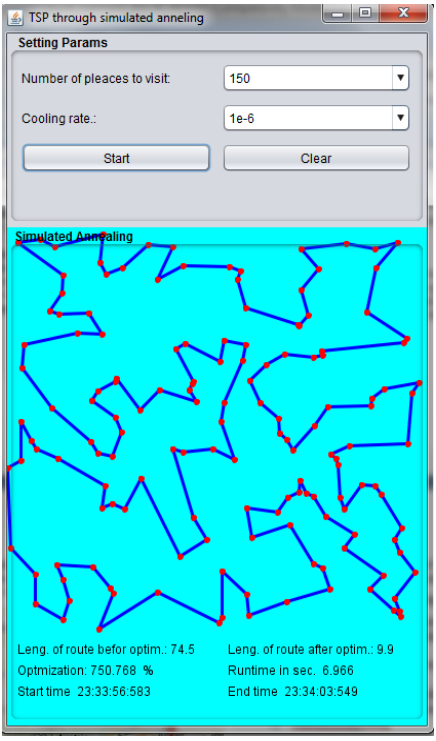

Fig 1. Programmed Test-tool in JAVA

VI. THE PROGRAMED SYSTEM

The whole system consists from web application as a client and a system as a server.

A. Client side

The system is implemented to solve the Travelling Salesman problem to determine the optimum route on Google map using

Google API and Simulated Annealing Algorithm in web – application. The system starts from getting different Geo Locations (Latitude and Longitude) on the map triggered by the user. The user clicks on the Google Map and markers the place he wants to visit.

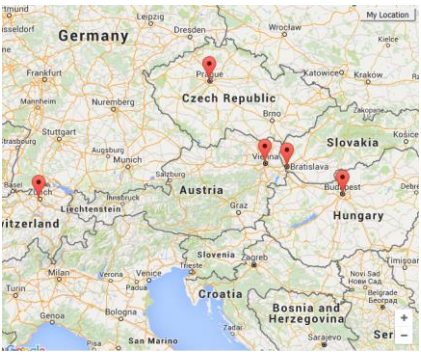

Fig 2. User marks the places on Google map through markes

Once the user has chosen places to visit and has select function "Optimize and map the route", the system derives all the necessary information from Google Map and packs them into JSON. JSON data are sent by web application to the server via HTTP-GET request through JQuery.

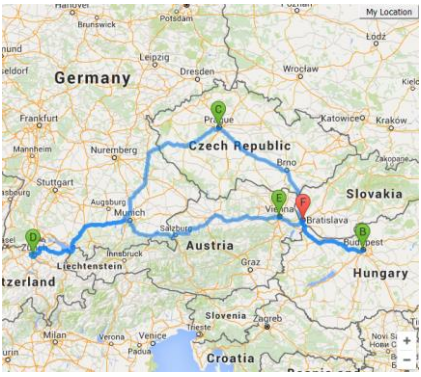

Fig 3. The optimized route on Google map drawn through polilynes

B. Server side

Travelling Salesman Problem has been analyzed and Simulated Annealing has been implemented, in Java, as RESTful web service, successfully. REST involves the transfer of *resources* between clients and servers. This server side system consumes JSON formatted data, produces XML- formatted data and is running on tomcat sever.

The server consists of the following classes:

a) Place→ models a google place. It contains all the characteristics of a google place: id, name, address, latitude and longitude
b) Tour → stores a candidate tour and is a list of places (ListArray<Place>)
c) TourManager → holds the places of a tour and is a list of places (ListArray<Place>)
d) SimulatedAnnealing

VII. CONCLUSIONS

Simulated Annealing Algorithm is based on neighborhood search. It has a strong analogy to the simulation of cooling of material.

Simulated Annealing Algorithm has a great influence on the optimization of round trip of TSP, based on results in above table.

Simulated Annealing reaches an approximate solution (close to the optimal).

The purpose of this paper is to show how to use Google - Services in combination with Simulated Annealing algorithm to solve TSP problem. This purpose makes working to discern from the papers of the earlier or existing.

ACKNOWLEGMENT

This paper is made possible through the help and support from my family.

REFERENCES

1. J.-Y. Potvin, "The traveling salesman problem", Annals of Operations Research 63, Canada, 2006, p.: 339-370.
2. R. Matai, S. Singh, M. Mittal: "Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches, Traveling Salesman Problem", Theory and Applications, India, 2010.
3. P. Eles: "Heuristic Algorithms for Combinatorial Optimization Problems: Simulated Annealing", Department of Computer and Information Science, Linköpings Universitet, Sweden, 2010.
4. Anupriya, Mansi Saxena: An Android Application for Google Map Navigation System Implementing Travelling Salesman Problem, in Proceedings of International Journal of Computer & Organization Trends –Volume3 Issue4, 2013.
5. F. Yang: "Solving Traveling Salesman Problem Using Parallel Genetic Algorithm and Simulated Annealing", 2010.
6. M. Kämpf: "Probleme der Tourenbildung", Technische Universität Chemnitz, Deutschland, 2006.
7. J. Brownlee: "Clever Algorithms Nature-Inspired Programming Recipes", Revision 2. 2012, ISBN: 978-1-4467-8506-5; p 175-176
8. D. Goossaert. (2010, April 6). Simulated annealing applied to the traveling salesman problem. Available at: http://codecapsule.com/2010/04/06/simulated-annealing-traveling-salesman/
9. M. Arostegui, S. Kadipasaoglu, B. Khumawala, "An empirical comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for facilities location", International Journal of Production Economics, vol 103, no 2, (2006)