EUROPEAN ACADEMIC RESEARCH
Vol. III, Issue 12/ March 2016

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

ISSN 2286-4822
www.euacademic.org

An Android Application to solve the shortest path problem using Google Services and Dijkstra's Algorithm

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

Abstract:

In this paper, I examine Dijkstra's algorithm in the context of solving the shortest path problem. The mathematical modeling of this problem has to do with the theory of graphs. The whole system is programmed as client system using Google Services and Android OS.

Google Services are used to determine the optimum path on Google map and solve the shortest path problem as alternative of Dijkstra's Algorithm.

Key words: Shortest path problem, Dijkstra's Algorithm, Google Services, Android

I. INTRODUCTION

Dijkstra's algorithm was found by the computer scientist named "Edsger Dijkstra" in 1959 in the object to solve the shortest path problem for the positive linking graph. This algorithm will calculate the shortest path from one point to another point in the graph one by one until the condition was met. [1]

A. Avoiding Confusions about shortest path

There are few points I would like to clarify before we discuss the algorithm. There can be more than one shortest path

between two vertices in a graph. The shortest path may not pass through all the vertices. It is easier to find the shortest path from the source vertex to each of the vertices and then evaluate the path between the vertices we are interested in. [2]

II. MATHEMATICAL FORMULATION OF SHORTEST PATH

Given a directed network G = (V; E; c) for which the underlying undirected graph is connected. Furthermore, a source vertex r is given. Find for each v ∈ V a dipath from r to v (if such one exists).

A. Mathematical Programming Formulation

Suppose r is vertex 1. For n - 1 paths have to leave r. For any other vertex, the number of paths entering the vertex must be exactly 1 larger than the number of paths leaving the vertex.

Let x_e denote the number of paths using each edge e ∈ E. This gives the following mathematical model:

$\min \sum_{e \in E} c_e x_e$
s.t. $\sum_{v \in V} x_{ev1}$ - $\sum_{v \in V} x_{1v}$ = -(n-1)
$\sum_{v \in V} x_{vu}$ - $\sum_{v \in V} x_{uv}$ = 1 u ∈ {2, … , n}
$x_e \in Z_+$ e ∈ E. [3]

III. DIJKSTRA'S ALGORITHM

The following algorithm can be used for directed as well as un-directed graphs. For the sake of simplicity, I will only consider graphs with non-negative edges.

Pseudo code with comments of Dijkstra's Algorithm	
dist[s] ← 0	distance to source vertex is zero
for all v ∈ V-{s}	set all other distances to infinity
do dist[v] ← ∞	S, the set of visited vertices is initially empty
S ← ∅	Q, the queue initially contains all vertices
Q ← V	while the queue is not empty
while Q ≠ ∅	select the element of Q with the min. distance
do u ← mindistance(Q, dist)	add u to list of visited vertices
S ← S ∪ {*u*}	if new shortest path found
for all v ∈ neighbors[u]	set new value of shortest path
do if dist[v] > dist[u]+w(u, v)	if desired, add traceback code
then d[v] ← d[u] + w(u, v)	
return dist ;	

[4]

A. Complexity analysis

There are two major operations in Dijkstra's algorithm:

Find minimum, which has $O(n^2)$ complexity; and Update labels, which has O (m) complexity. Therefore, the complexity of Dijkstra's algorithm is = $O(n^2+m)$.

There are several implementations that improve upon this running time.

One improvement uses Binary Heaps: whenever a label is updated, use binary search to insert that label properly in the sorted array. This requires O(log n). Therefore, the complexity of finding the minimum label in temporary set is O(m log n), and complexity of the algorithm is O(m + m log n) = O(m log n). [5]

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. [6].

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. [6]

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. PROGRAMMED SYSTEM

The whole system is programmed as android application to determine the shortest path using Google services. The system starts from getting different Geo Locations (Latitude and Longitude) on the map triggered by the user in two ways:

1) The user clicks on the Google Map and marks two place he wants to visit
2) The user knows the name or address of the place he wants to visit and search through autosuggestion. Then the list will be proposed to user, then user selects the desired place, and after that, the place will be marked on the map. This is possible by using a Google web service called Google places.

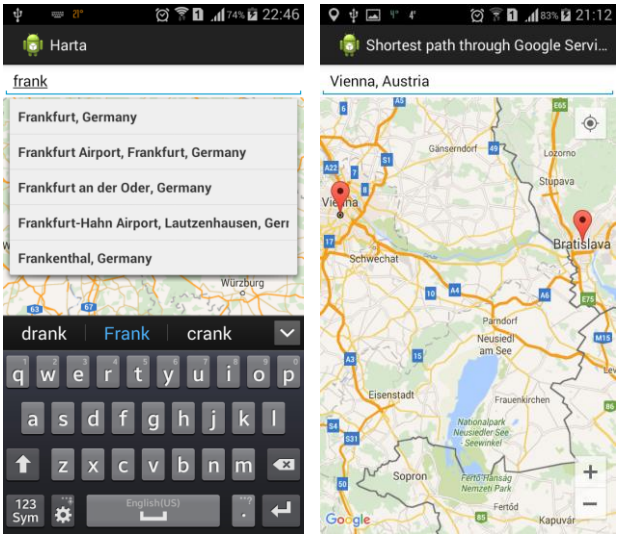

Figure 1: Autosuggestion and marked places on implemented Android application

Once the user has chosen places and has select function "Find shortest path", the system derives all the necessary information

from Google map and packet them into JSON. JSON data are sent by application to the Google server via HTTP-Post request.

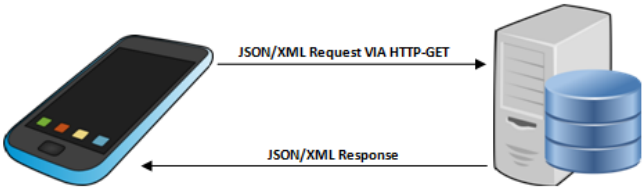

Figure 2: Communication between android application and Google Servers

Based on returned results from Directions Google Service, I map the path on Google Map through poly lines.

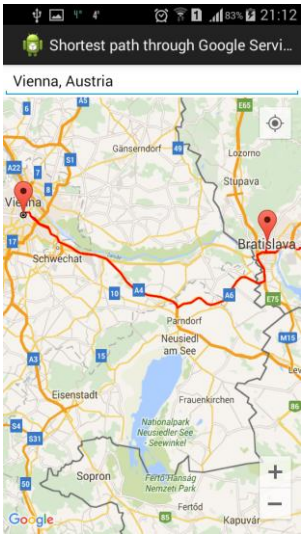

Figure 3: Optimized path and drawn on Google Map through poly lines

VI. CONCLUSIONS

Dijkstra's algorithm has a great influence on the optimization of shortest path problem, and there are several implementations that improve upon this running time.

I presented an analysis of various aspects associated with the complexity of Dijkstra's algorithm.

I implemented a solution to shortest path problem using Google services and Android OS as an alternative of Dijkstra's Algorithm for high graph densities or graph size.

Based on information on the website of Google the Google Maps Directions API has the following limits in place:
Users of the standard API:

- 2,500 free directions requests per day
- Up to 23 waypoints allowed in each request containing an API key, or up to 8 waypoints when no API key is supplied, or up to 8 waypoints when using the Directions service in the Google Maps JavaScript API
  Waypoints are not available for transit directions
- 10 requests per second [7]

BIBLIOGRAPHY

[1] , E. W. Dijkstra , "A note on two problems in connexion with graphs", accessed February 27, 2016, http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf
[2] Techie Me. "Shortest Path using Dijkstra's Algorithm", Last modified March 30, 2015. http://techieme.in/shortest-path-using-dijkstras-algorithm/
[3] Jesper Larsen & Jens Clausen, "The Shortest Path Problem", accessed January 10, 2016. http://www.imada.sdu.dk/~jbj/DM85/lec6a.pdf
[4] M. Yan, "Dijekstra's Algorithm", accessed February 25, 2016.
http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf
[5] Dorit S. Hochbaum, "Graph Algorithms and Network Flows", accessed January 06, 2016. http://www.ieor.berkeley.edu/~hochbaum/files/ieor266-2014.pdf

[6] Anupriya, Mansi Saxena: "An Android Application for Google Map Navigation System Implementing Travelling Salesman Problem", International Journal of Computer & Organization Trends, vol. 3 no. 4 (2013), accessed February 12, 2016.
[7] Google. "Google Maps Directions API", Last modified February 18, 2016. https://developers.google.com/maps/documentation/directions/usage-limits

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