Smart Navigation:

Using Artificial Intelligent Heuristics in Navigating Multiple Destinations

Hatem F. Halaoui
Computer Science
Haigazian University, Lebanon
Email: hhalaoui@haigazian.edu.lb

***Abstract*—Navigation applications are becoming an essential need in any mobile device. Finding the best path (time and distance) from an address to another is one of the most asked queries among driving users. Moreover, finding the best path with multiple destinations is a query that could be asked by many, including commercial companies' drivers (similar to the famous "Traveling Salesman Problem"). Google maps, Yahoo maps, and tens of other solutions are examples of such mobile applications. Calculating the best driving path between two addresses is subject to many factors including distance, road situation, road traffic, speed limitations and others. This paper presents the use of smart heuristic functions, as well as an efficient data structure to be used in finding efficient path between multiple points (addresses) rather than one destination. It presents spatial databases, current solutions, heuristics in Graph problems, and finally a smart solution (our new Algorithm A*Multiple) using a smart heuristic function to determine the best path between multiple destinations.**

Keywords: Smart Navigation, Artificial Intelligence, Heuristics, GIS.

I. INTRODUCTION

This section introduces the paper's main subject. First, the idea of heuristics is briefly presented. Second spatial databases are introduced as the underlying databases to store related data. Finally in this section, a brief introduction to Geographical Information Systems (GIS) and driving path application are presented.

A. Heuristics

Most of what we do in our daily lives involves heuristic solutions to real-time problems. As an adjective, heuristic pertains to the process of gaining knowledge by intelligent guess rather than by following some pre-established formula [1][2]. In map problems, when moving from one point to another to reach a certain destination, we have two options:

1- The algorithm tries all possible paths from all possible neighbors (next address on the way to destination). It keeps doing this until destination is reached. Finally it chooses the best path among all possibilities.

2- At each location, the algorithm chooses the next move smartly using some evaluation function (called the heuristic function)

The first option is very time consuming and does not match with real-time problems. As a result, a solution using the second is being adopted in this paper

B. Spatial Databases

Spatial databases are the main data warehouses used by Geographical Information Systems. Spatial databases are databases used to store information about geography like: geometries, positions, coordinates, and others [8]. Also, they might include operations to be applied on such data.

C. Geographical Information Systems and Driving Path Applications

Geographic Information System (GIS) is a collection of computer hardware, software for capturing, managing, analyzing, and displaying all forms of geographical information [8].

Finding the Directions (driving/walking) path is one of most asked queries in GIS applications. These are the most important factors that influence such criteria::

- Distance: Distance between the source and destination.
- Road situation: Is the road closed? Is it raining?
- Road traffic: How much traffic?
- Speed limitations: Is there many traffic lights? Is it a local road or highway? What is the average speed?

D. Navigating Using Heuristic Functions

In this paper, we present the issue of navigating multiple destinations in any order. Our main problem is to find the fastest path between a given source passing over all given destinations in any order. The importance of our approach is that existing solutions, like Google Maps [7], let the user choose his order of destinations rather than suggesting a fast path.

Moreover, calculating the fastest path with traditional Mathematical algorithms like Hamilton path has a high time complexity and hence time-expensive for real time problems like the one in this paper. As a result we use heuristic algorithms like A* to incredibly minimize the running time of such navigation real-time solutions.

Our approach offers the user a full path with an order of destinations claiming an efficient time. The main concern is that heuristic functions does not guarantee an optimal (best) solution. For this reason, choosing the heuristic function is an important factor for getting good results. Choosing a good heuristic function in order to choose our series of destination is an open research question. Moreover, choosing the heuristic function is highly dependent on the geography of surface in query.

The paper is organized as follows Section 2 presents some related work including widely used applications. Section 3 presents the main solution in this paper. Section

 ISBN: 978-1-61208-443-5

4 discusses some results and finally section 5 presents conclusions and future work.

II. STATE OF ART AND RELATED WORK

Most of current applications provide a one destination solution (users provide the application with source and destination). If users intend to visit multiple destinations, they have to decide the order of visits and make different queries each time.

This section presents an overview of related theoretical and applied related work. First, the use of Artificial Intelligence in path problems is presented, discussed and compared with traditional optimal solution algorithms. Moreover, one of the most used applied application (solutions) is presented.

A. *Driving Direction Applications: Google Maps as an example*

This sub-section presents a widely used application for finding driving directions: Google Maps.

Google Maps [7] is a Web-based service that provides detailed information about geographical regions and sites around the world. In addition to conventional road maps, Google Maps offers aerial and satellite views of many places. In some cities, Google Maps offers street views comprising photographs taken from vehicles. Google Maps [7] offers several services as part of the larger Web application, as follows:

- A route planner offers directions for users of routes and public
- Google Maps for Mobile offers a location service for motorists that utilizes the Global Positioning System
- Google Street View enables users to view and navigate through horizontal and vertical panoramic street level images of various cities around the world.
- Provides user interaction.

Figure 1 shows an example a driving directions query using Google Maps [7]. The query is to get driving directions, over multiple destinations in London: Paddington station, Harrods, House of Commons, and London Eye. It also offers Real-time Traffic information. However Google Maps [7] does not suggest any order of visits. The user has to provide Google Maps with the order and he has to make multiple trials and look for the best sequence of destinations to be visited.

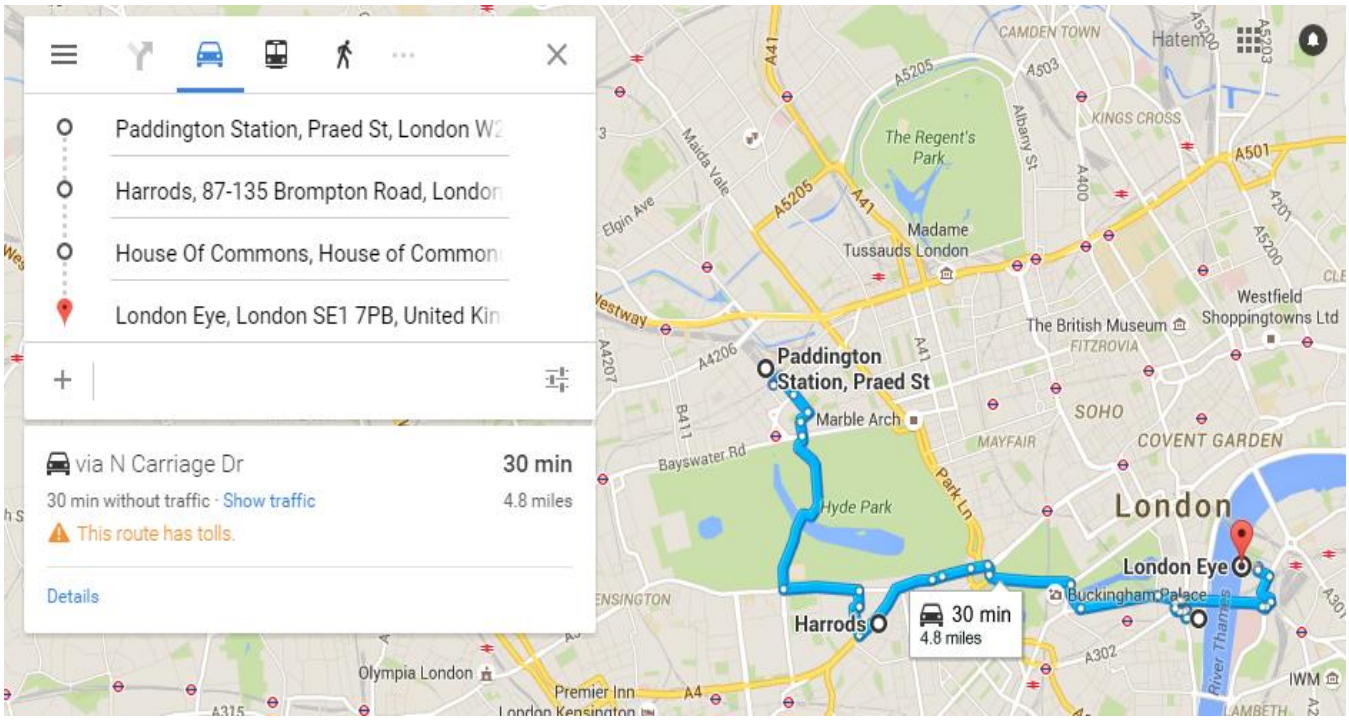

Figure 1. Path over multiple destinations in London

B. *Artificial Intelligence and Driving Directions*

Artificial intelligence is involved in graph searching algorithms. Russel and Norving [2] present many intelligent graph searching algorithms. Here are two important ones:

1. Greedy Best-First Search
2. A* Search

The main idea behind these algorithms is that they do not try all possible cases to give an answer. On the other hand, the algorithms use heuristic function to un-consider some of the paths. This issue saves huge amount of time but does not guarantee a best path. However, finding a good heuristic function could guarantee up to 95% finding the best path. Section 3 include the A* search algorithm which clears the idea in this section

C. *A* Traffic: Design, Algorithms and Implementation*

This section presents the application algorithms and the application of the intelligent driving path application used in our previous work, which is extended in this paper.

A* [2] is an Artificial Intelligent graph algorithm proposed by Pearl. The main goal of A* is to find a cheap cost graph path between two vertices in a graph using a heuristic function. The main goal of the heuristic function is to minimize the selection list at each step. In the graph example, finding the shortest path from a node to another has to be done by getting all possible paths and choosing the best, which is very expensive when having a huge number of nodes. On the other hand, using an evaluation function (heuristic) to minimize our choices according to intelligent and practical criterion would be much faster.

The heuristic function is not a constant static function. It is defined according to the problem in hand and passed to the A* as a parameter. In the case of A* search for a direction path, F is built up from two main factors:

H = Straight Line distance to destination.

G = Distance Traveled so far.

$$F = H + G \quad (1)$$

At each node n, we compute F (n) and we choose our next step accordingly.

A* Algorithm

A*(Graph, Source, Destination)
Task: takes a Graph (Vertices and Edges), Source and Destination (Vertices) and returns the Best path solution (stack of vertices) from Source to Destination

If Source = Destination then return solution (stack)
Else expand all neighbors Ni of Source
  Mark Source as Unvisited
  For each Neighbor Ni
    Get VNi = H(Ni, Destination)
    Add all (Ni, Vi) to the Fringe (list of all expanded Vertices)
    From the Fringe, Choose an Unvisited Vertex V with Least Vi
    If no more Unvisited return Failure
    Else Apply A*(V, Destination)

H(V, Destination)
Task: takes a vertex V and evaluate it using a heuristic function
Return: DistanceSoFar + StrightLineDistance (V, Destination)

Where
Distance_So_Far= Distance taken so far to reach the Node V
Stright_Line_Distance (V, Destination) = straight line distance to destination calculated by using the coordinated of V and destination

Figure 2 is an example of the A* algorithm behavior to find a path starting from “Arad” to “Bucharest”, cities in Romania [2]. First of all we start at Arad and go to the next neighbor with the best heuristic function (Sibiu). Second, explore all neighbor of Sibiu for the best heuristic function. The algorithm continues choosing the best next step (with the least value of heuristic function) until it reaches Bucharest. The interesting thing is that all vertices with values (calculated using the heuristic function) kept in the fringe in order to be considered at each step.

 ISBN: 978-1-61208-443-5

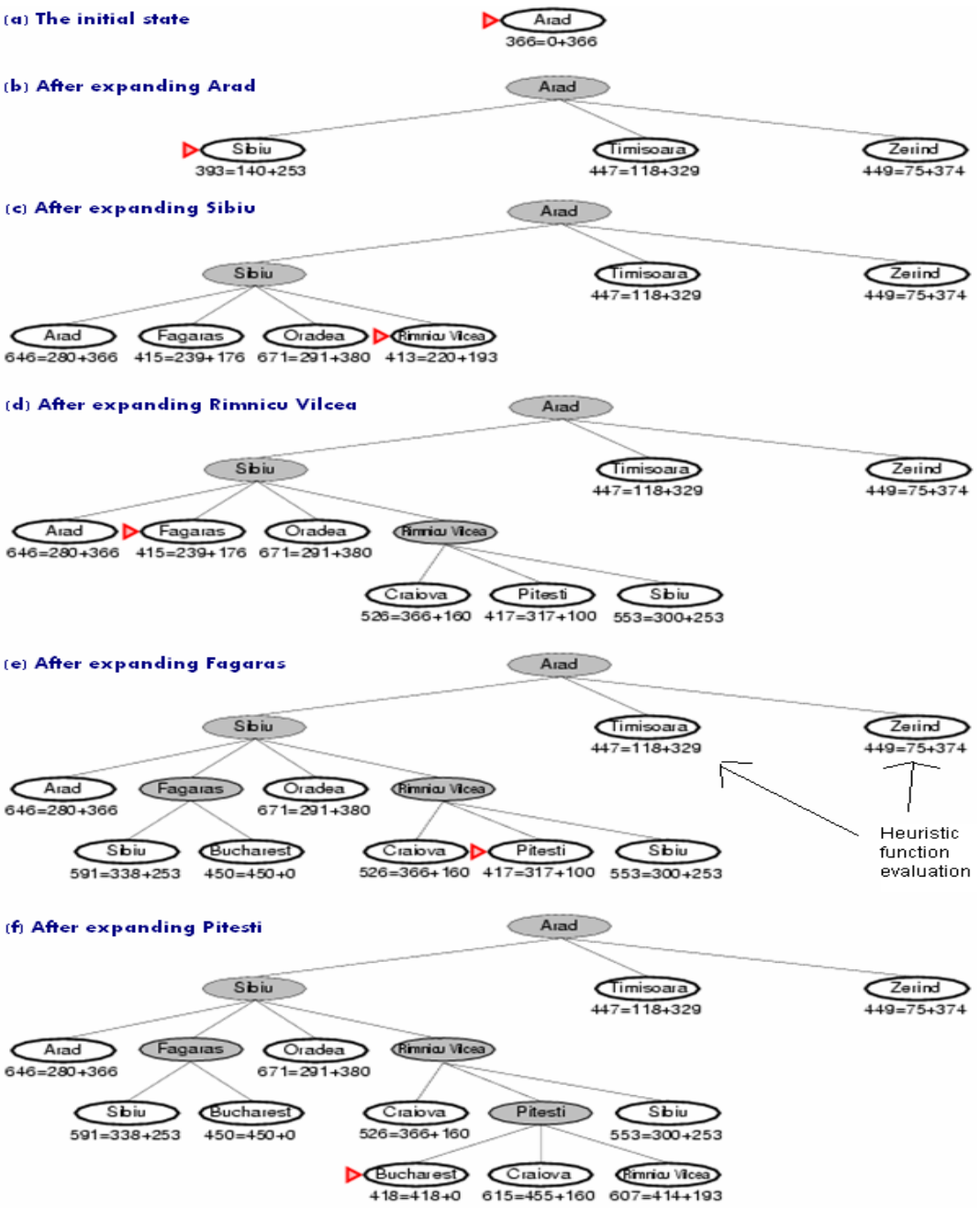

Figure 2. A* algorithm behavior to find a path starting from "Arad" "Bucharest city" [2]

D. *A*Traffic: A Variation of A* with Road Traffic as a Factor*

A*Traffic [5] (our previous approach) is a variation of A* with the ability to take Online traffic into consideration. The main job is done in the heuristic function where a new factor is used to choose the next step. The new factor is the average traffic value (got online from real time databases) represented in the following form time/distance (example: 3 min/km). The new Heuristic function will be:

$$F = H + G + T \quad (2)$$

Where:
H = Straight Line distance to destination (distance between two coordinates).
G = Distance Traveled so far.
T= Average Traffic delay got from real online sensors.

E. *Testing Tool: Query Example*

This sub-section present the layout of the testing tool developed to test the algorithm proposal "A*Traffic". For this purpose, an example query is presented.

A Query example

This example (Figure 3) demonstrates the main feature of the software. It provides the user with the driving directions between "HU, Kantari St, Hamra" (Haigazian University) and "AUB, Bliss St, Hamra" (American University in Beirut) in Beirut, Lebanon. The blue path generated is a short path (using A*Traffic) to follow in order to drive from the start address to the destination address.

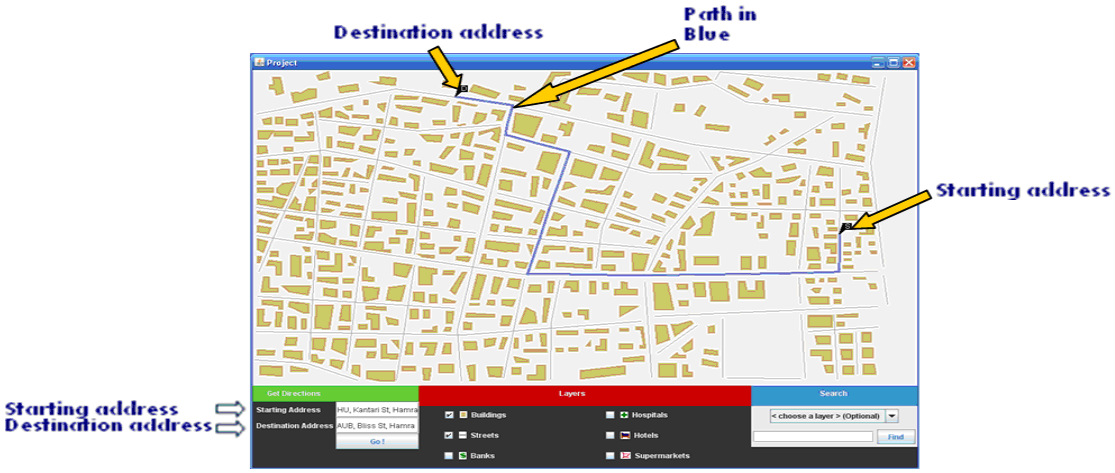

Figure 3. Path from "HU, Kantari St, Hamra" to "AUB, Bliss St, Hamra"

III. USING HEURISTICS AND TIME-WEIGHTED GRAPH IN MULTIPLE DESTINATIONS DIRECTION SOLUTIONS

This section, presents our new extension of the previous solutions to provide a good solution having multiple destinations to be visited.

In this section, we propose our smart solution for navigating multiple destinations. The section includes proposals for:

1. Time Weighted Graphs (TWG) as the main data structure used in our solution.
2. A*Multiple: the proposed smart algorithm
3. Execution examples of A*Multiple

A. *Time Weighted Graphs*

In our previous paper [6], we present TWG as our main data structure: a graph representing the map with edges weighted by numbers (minutes) representing the estimated time needed to drive the edge.

Distance is usually the main edge weight in Graphs. In TWG, time is used instead. A graph edge in the graph represents a road, street, highway, or part of any. Each of these has an average speed limit that is calculated using road speed sensors. The job of the real-time street sensor is to watch traffic and send the average speed during some period. The graph edge weight in terms of time (minutes) is computed as follows:

Average speed (AV) (minutes) =

Sum of speeds of n cars over a Period T (miles/minutes) ÷ n

The initial weight of the edge (minutes) (W) = (edge distance (miles) ÷ AV (miles/minutes))

B. *Example: Part of Manhattan in a Time-weighted Graph*

This Section shows the data structuring (transferring the map into TWG) of part of Manhattan (taken from Google Maps). Figures 4 and 5 show the location of vertices in the map (Figure 4) and digital graph after construction (Figure 5).

Figure 4. Modeling graph vertices

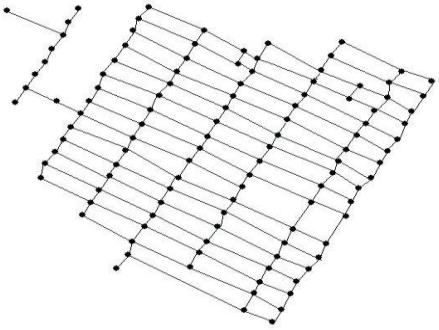

Figure 5. Building the graph edges (directed)

*The A*Multiple Algorithm*

The main idea behind A*Multiple is to find the best path (shortest in time) to visit multiple destinations in one tour. The algorithm uses a heuristic function to find the next destination and then uses the A*Traffic (which also use the same heuristic function) to travel to that destination.

A*Multiple (Source, Destinations)

Task: find an efficient path from source passing over all members in Destinations array.

Returns: 2 Lists

- VSL: The Vertices Solution List VSL, which is the vertices t visit in order
- PSL: Path Solution List PSL, which is the list of paths to take each time to each destination (vertex)

Pseudo code

If Destination is Empty return Done
For all Vertices Vi in Destinations
Di=H(source, Vi)
Get the Vs with the Minimum Di
Remove Vs from Destinations
Add Vs to the Vertices Solution List VSL
Add A*Traffic (Source, Vs) to the Path Solution List PSL
If A*Traffic fails return Failure.
A*Multiple (Vs, Destinations).

*How does A*Multiple Work?*

This section presents the execution of A*Multiple. To present our approach better, consider the following problem:

Suppose I am at Paddington station and want to visit the following destinations in London: "Eye of London", "House of Commons", and "Harrods". If my only priority is time, means that I can visit them in any order with efficient time. In this case, I have to choose my next destination (at each step) smartly.

After creating the Time-Weighted graph (vertices shown in black in figure 7, over 5000 vertices) over the map of London (from Google Maps), the A*Multiple will return the following:

VSL: Harrods, House of Commons, Eye of London.

PSL: Path1, Path2, Path3.

Where VSL is the ordered list of destinations to be visited, PSL is the list of paths from each destination in VSL to the next one, Path1: Paddington – Harrods, Path2: Harrods – House of Commons, and Path3: House of Commons – Eye of London.

Each of these paths is calculated using A*Traffic.

Figure 6 shows these solutions in different colors: orange (Path1), Blue (Path2) and Pink (Path3). It also gives estimated time of each path according to current (at time of calculation) traffic situation.

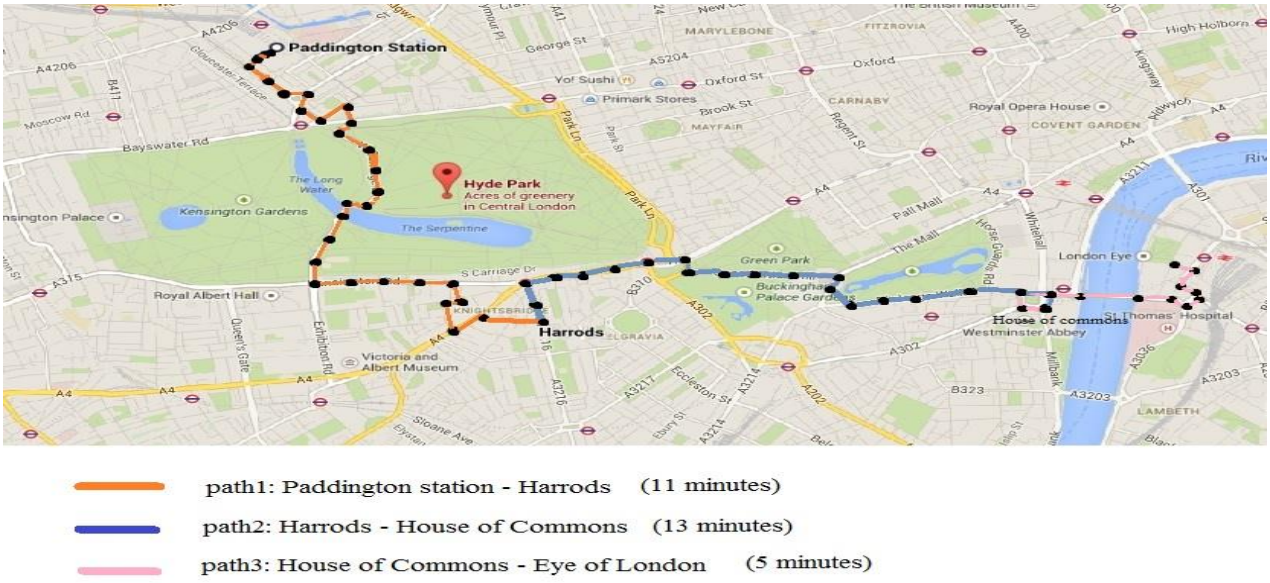

Figure 6. Paths for Multiple destinations (Paddington, Harrods, House of Commons, and Eye of London

IV. RESULTS

We have developed a testing tool (to test our approach) where 100 samples were tested in 3 groups. Our results showed that our solution is optimal in 81%. Table I presents the gathered results in each group/each case where:

- Optimal solution: Absolute best solution.
- Good solution: takes maximum of 20% more time than optimal solution.
- Bad solution: Takes more than 20% more time than optimal solution.

TABLE I. PERCENTAGES OF QUALITY OF SOLUTIONS

Distances	Optimal solution	Good Solution	Bad Solution
More than 10 destinations Over 5321 vertices	73 %	19%	8%
Less than 10 destinations Over 5321 vertices	88%	9%	3%
Average	81%		

Our approach is being evaluated according to optimal solutions (best solution). These optimal solutions were computed manually. Finding such solutions is time consuming and not applicable real time problems like navigation problems.

The Comparison of our solution to Google Maps [8], shows that ours suggests a smart order of visits over all destinations while we have to give the order of destinations to Google Maps In order to get the best solution we have to do try all possible orders of destinations, then we have to check for the best order (least time). This is not reliable when having a long list of destinations (over 10).

V. CONCLUSIONS AND FUTURE WORK

In brief, our approach, using A*Multiple algorithm with time-weighted graphs, has the following advantages:

- It uses time-weighted graphs, which takes distance and speed into consideration.
- It considers multiple destinations
- It saves a lot of execution time.

The reason this approach saves a lot of time is because if we do not use heuristics, we will have to find the path by getting all combinations (Hamilton Path). This will result into an exponential time algorithm. On the other hand, using heuristics considers options with best values when heuristic function is applied and we end up with polynomial time algorithm (A*Multiple and A*Traffic are in $O(n^3)$ since they rely on A* [2]).

Our future work is focused on how to use heuristics combined with discrete structures algorithms like Hamilton path and Hamilton circuit. Moreover, finding the best heuristic function remains an open research question in the future.

REFERENCES

[1] J. Pearl, *Heuristics: Intelligent Search Strategies for computer Problem Solving*. Addison Wesley, Reading, Massachusetts, 1984.

[2] S. Russell and Peter Norving, *Artificial Intelligence a Modern Approach*. Prentice Hall, Upper Saddle River, New Jersey, 2003.

[3] H. Halaoui, Smart Traffic Online System (STOS): Presenting Road Networks with time-Weighted Graphs". IEEE International Conference on Information Society (i-Society 2010) London, UK. June 2010, pp. 349-356.

[4] Google Earth Blog Google Earth Data Size, Live Local, New languages coming Available: http://whatis.techtarget.com/definition/Google-Maps. Retrieved: September, 2015.

[5] H. Halaoui, "Smart Traffic Systems: Dynamic A*Traffic in GIS Driving Paths Applications". Proceeding of IEEE CSIE09. IEEE, Los Angeles, USA. March, 2009, pp. 626-630.

[6] H. Halaoui," *Intelligent Traffic System: Road Networks with Time-Weighted Graphs"*. International Journal for Infonomics (IJI), Volume 3, Issue 4, December 2010, pp. 350-359.

[7] Google Maps. Available: https:// Maps.google.com. Retrieved: September, 2015.

[8] H. Halaoui. "Spatial and Spatio-Temporal Databases Modeling: *Approaches for Modeling and Indexing Spatial and Spatio-Temporal Databases".* VDM Verlag, 2009.

 ISBN: 978-1-61208-443-5