Spatial Model of Road Network in Colombo City for Optimum Online Path Finding

S.E. Jeningthas[1], HUW Ratnayake[2]
Department of Electrical & Computer Engineering
Faculty of Engineering Technology, The Open University of Sri Lanka
senjeningthas@gmail.com[1], udithaw@ou.ac.lk[2]

***Abstract* – The Optimum Online Path Finding System is an online path guide for bus travelers within the Colombo city. It is designed to show the public, the optimum route that they can take to travel from one place to another since there are several bus routes within the city.**

The system is implemented using bus route details provided by the National Transport Commission (NTC). The system answers questions from the public regarding bus routes. It shows them possible routes from a source to a destination from where they can make decisions regarding their travel.

Optimum path finding algorithm is based on the A* algorithm. The prototype implemented can successfully show graphically, the optimum path with the bus route numbers.

1. Introduction

There are several bus routes and local routes within the Colombo city to travel from one place to another. This system answers questions of the public regarding bus and other transport means. It shows them possible routes with bus route numbers from a source to a destination therefore they can make decisions regarding their travel.

Since going through the website is less time consuming and less expensive this is a better additional service that NTC can ensure for the public. The NTC is the organization which allocates bus routes to both State and private buses in Sri Lanka.

This system takes user input of start and end location, and produces a set of directions, as well as a map, specifying the directions between the two locations. This paper is organized as follows: the methodology of the search algorithm is given in section 2, how the system relates to real road network is discussed in Section 3, the algorithm of finding the optimum path is described in section 4, section 5 describes implementation details, and further improvements and conclusions are given in sections 6 and 7 respectively.

2. Methodology

The A* Path finding algorithm is one of the most commonly used implementations of Artificial Intelligence. The algorithm can find the optimum path between the two points in a search area. The processing of the algorithm is divided into the following steps:

- Dividing the search area
- Scoring the location
- Starting the search
- Saving the data

2.1 Dividing the Search Area

The A* Path finding algorithm reduces the search area to a two dimensional array, which represents the location that are active or inactive. On All major roadways present on Colombo city are identified as active, while all other areas (buildings, treed areas, etc) have been identified as inactive. The algorithm searches the adjacent location of the start location and searches outwards until the target point is found.

2.2 Scoring the Squares

The F-cost is the key to determine which location should be chosen. The following equation gives the F-cost: ***F = G + H***

In the above equation, **G** is the movement cost to move from the start location to the current location, and **H** is the estimated movement cost to move from the current location to the target location. The optimum path is simply found by repeatedly choosing the location with the lowest F-score. The method, which can estimate the **H** score, is called the Manhattan method. The F-score is calculated by adding the **G** and **H**. The following figure illustrates the scores of the F-cost equation.

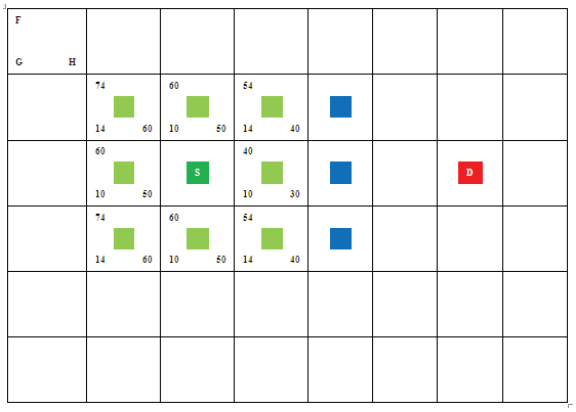

Figure: 2.1 - A* F-scores

The figure 2.1 shows F score is in the top of each chosen location. The G score is in the bottom left, and the H score is in the bottom right. To reiterate, the F score is calculated by adding the G and H scores.

2.3 Starting the Search

After scoring the squares, the squares with the lowest F scores are added to an open list. The algorithm then checks all the adjacent squares and adds them to the open list if the adjacent squares are not on the list. Next, the algorithm checks for a better path if the adjacent squares are on the open list already. The process is repeated until the target point is found on the open list. The following figure illustrates the result of the process:

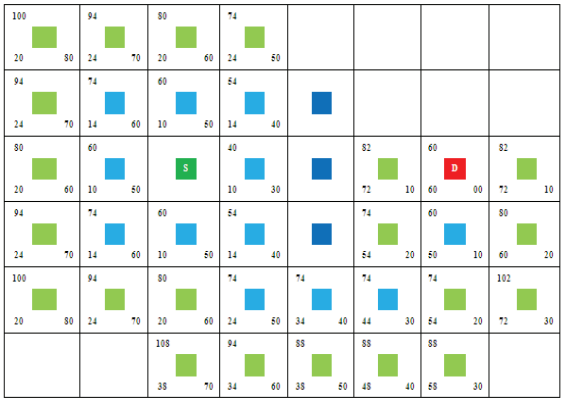

Figure: 2.2 - Proceeding with the A* Algorithm

The search is completed when the target point is on the open list as shown in the above figure 2.2. If the algorithm fails to find the target point, then the open list will be empty. That means there is no optimum path.

2.4 Saving the Data

After the optimum path is found, the algorithm saves the path backwards from the target point to the start point. The algorithm works backwards moving from one square to its parent square until the start point is reached. The following figure illustrates the process:

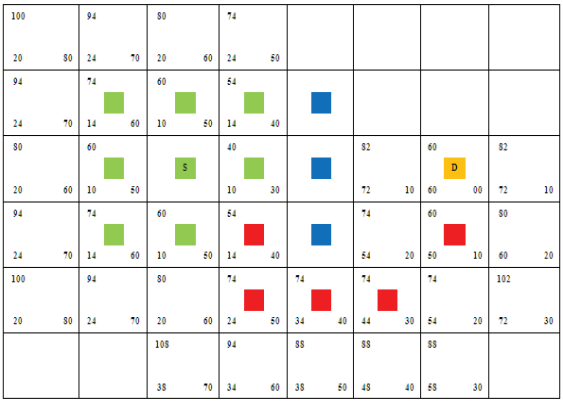

Figure: 2.3 - Saving the Calculated Path

The dots represent the optimum path. To reduce the overall search time, a number of methods can be used. One particularly quick method is the use of binary heaps. This method is at least 3-4 times

faster than the other methods. In our implementation, the map is divided into number of squares, and the data is stored in a two dimensional array.

2.5 Overview of the system

The following diagram depicts an overview of the proposed system.

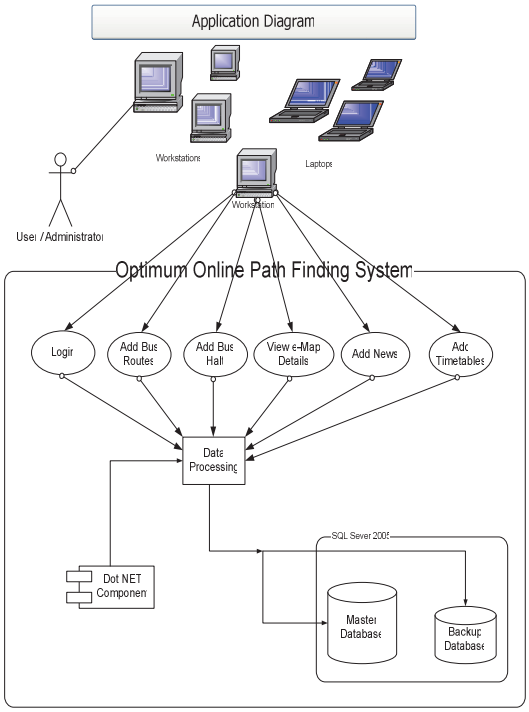

Figure: 2.4 – Overview of the System

3. Design of spatial network

Let a network consist of a set of nodes and a set of links connecting to the nodes. A network can be represented by the notation G = (N, A), where N is the set of nodes and A is the set of links of the network G.

Let LC(i, j) be the nonnegative **Link Cost**(LC) required to travel from node i to node j and LEC(0,i) be the **Link End Cost**(LEC), or minimum path cost from origin to node i through link(0,i) which refer to the directed link leading from node 0 to node i.

Link-based optimum path optimality condition is given as follows.

LEC(0,i) + LC(i,j) ≤ LEC(i,j) , FORALL 0,i,j ∈ N

3.1 Spatial Network

The following Figure: 3.1 shows the spatial network querying example

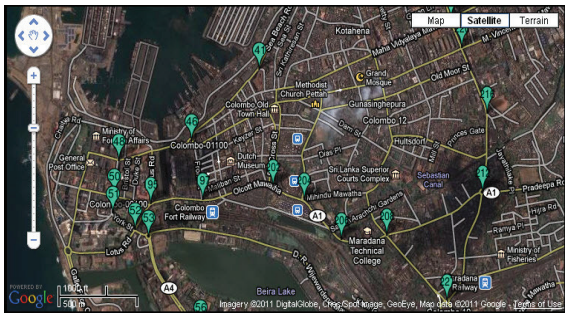

Figure: 3.1 - Spatial network

3.1.1 Shortest path query

Path between the two locations (Source and Destination) adheres to *one way routes* and *Bus routes numbers* which form the shortest path of all available paths.

3.1.2 Optimum path query

When finding the minimum travel time between any two locations (Source and Destination), it adheres to one way routes and Bus routes number which is the fastest route in the available paths.

When selecting fastest route following conditions will be taken into consideration.

(i) Traffic status
(ii) Weather
(iii) Road construction

3.2 Spatial network data models

There are three levels of Database Design.

3.2.1 Conceptual Data Models

Graphs

- A *graph*, G = (V,E)
- V = a finite set of *vertices*
- E = a set of edges E , between vertices in V

3.2.2 Logical Data Model

- Data types - Graph, Vertex, Edge, Path, …
- Operations - Connected( ), Shortest_Path( ), Optimum_Path( ), ...

Common data types

- Vertex: attributes are label, isVisited, (location for spatial graphs)
- DirectedEdge : attributes are start node, end node, label
- Graph : attributes are setOfVertices, setOfDirectedEdges, ..
- Path : attributes are sequenceOfVertices

3.2.3 Physical Data Models

- Record and file representation
- File-structures and access methods

Categories of record/file representations

- Main memory based
- Disk based

Main memory representations of graphs

- Adjacency matrix M[A, B] = 1 if and only if edge(vertex A, vertex B) exists
- Adjacency list : maps vertex A to a list of successors of A
- Disk based
  - Normalized - tables, one for vertices, other for edges
  - De-normalized - one table for nodes with adjacency lists

Concept of Transitive Closure

- .Consider a graph G = (V, E)
- Let G* = Transitive closure of G
- Then T = graph (V*, E*), where
- V* = V
- (A, B) in E* if and only if there is a path from A to B in G.

4. Algorithm of Link based optimum path

To illustrate the searching algorithm to find the optimum or shortest path between two places in the city, the following example given in Figure 4.1 with the real road network can be used.

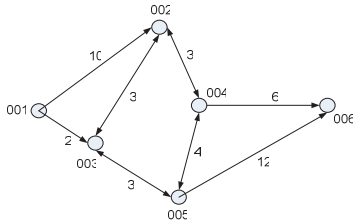

Figure: 4.1 - Real road network

In this diagram, each node represents a place or a bus stop, and the vertices between nodes represent roads that join places together. 001 is the starting node, and 006 is the goal node. Each link is labelled with a travel time or distance, which shows traffic or how long that road is.

The aim of this problem is to find the optimum path or shortest possible path from one place 001 to another place 006. The following search space of the map in Figure 5.9 4.2 depicts a search tree by showing each available search path up to a leaf node in the tree.

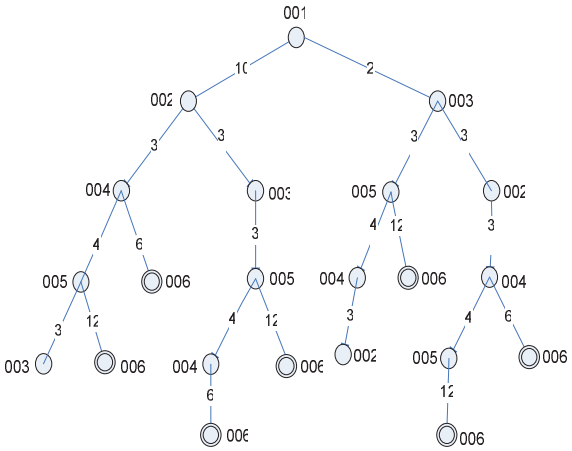

Figure: 4.2 - Available search paths

It can be seen that the above tree has nine leaf nodes, seven of which are reached goal nodes. Two of the paths lead to cyclical paths so they are abandoned. The seven distinct paths that successfully lead from node 001 to node 006 can be traced from the tree as follows:

Solution Path	Nodes	Traffic link cost
1	001-002-004-005-006	10+3+4+12=29
2	001-002-004-006	10+3+6=19
3	001-002-003-005-004-006	10+3+3+4+6=26
4	001-002-003-005-006	10+3+3+12=28
5	001-003-005-006	2+3+12=17
6	001-003-002-004-005-006	2+3+3+4+12=24
7	001-003-002-004-006	2+3+3+6=14

Table: 4.1 – Traffic Link Host

The optimum path is solution 7 that is 001, 003, 002,004, 006, which has a traffic link cost is 14.

4.1 The main evaluation function used in the search algorithm.

The square euclidian distance between two nodes : $Dx^2+Dy^2+Dz^2$

Here,

Dx = N1.Position.X - N2.Position.X;

Dy = N1.Position.Y - N2.Position.Y;

Dz = N1.Position.Z - N2.Position.Z;

A straight line between those two points i and j (i.e. the Euclidean distance):

The euclidian distance between two nodes : $Sqrt(Dx^2+Dy^2+Dz^2)$

$$d(i,j)=\sqrt{(x(i)-x(j))^2+(y(i)-y(j))^2+(z(i)-z(j))^2}$$

where x(i), x(j) , y(i), y(j) and z(i), z(j) are the coordinates for node **i** and the destination node **j** respectively.

$$\text{Distance Link Cost (DLC)} = \sum_{E=1}^{n} d(i,j)$$

ActualSpeed – AS
TraficStatus – TS
WeatherCondition – WC
Road Construction – CON

AS=(AVGSpeed-AVGSpeed*((TS*0.65)+(WC*0.15)+(CON*0.20)))

$$\textbf{Traffic Link Cost (TLC)} = \sum_{E=1}^{n} \frac{d(i,j)}{AS(i,j)}$$

4.3 Algorithm of link based optimum path

The attributes **LC**, **LCH**, and **LEC** are defined as follows:

- **LC** is the **Link Cost** of getting from the start node to the current node i.e. the sum of all the values in the path between the start and the current node
- **LCH** stands for **Link Cost Heuristic** which is an estimated Link Cost from the current Node to the goal node (usually the straight line Link Cost from this node to the goal)
- **LEC** is the sum of **LC** and **LCH** and is the best estimate of the time cost of the path going through the current node. In essence lower the value of LEC the optimum path is more efficient.

The purpose of ***LEC***, ***LC***, and ***LCH*** is to quantify how promising a path is up to the present node. Additionally, this algorithm maintains two lists, an ***Open*** and a **closed** list. The Open list contains all the nodes in the map that have not been fully explored yet, whereas the **closed** list consists of all the nodes that have been fully explored. A node is considered fully explored when the algorithm has looked at every node linked to it.

The pseudo-code for **link based optimum** path Algorithm is as follows:

1. Let S = starting point.

2. Assign LEC, LC and LCH values to S.

3. Add S to the Open list. At this point, S is the only node on the Open list.

4. Let B = the best node from the Open list (i.e. the node that has the lowest LEC - value).

a. If B is the goal node, then quit – a path has been found.

b. If the Open list is empty, then quit – a path cannot be found

5. Let C = a valid node connected to B.

a. Assign LEC, LC, and LCH values to C.

b. Check whether C is on the Open or Closed list.

i. If so, check whether the new path is more efficient (i.e. has a lower LEC-value).

1. If so update the path.

ii. Else, add C to the Open list.

c. Repeat step 5 for all valid children of B.

6. Repeat from step 4.

5. Implementation

This application runs in local IIS server.

5.1 Capture location detail

Figure 5.1 shows the locations where latitude and longitude are captured from the Google map to update the databases.

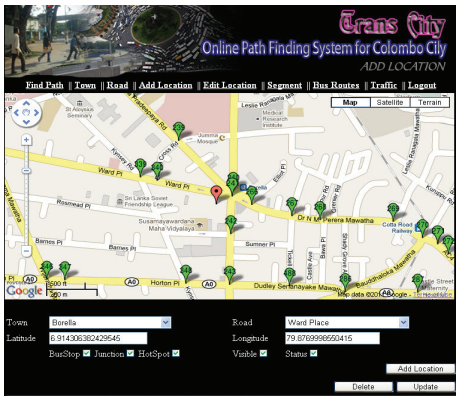

Figure: 5.1 – Location details

5.2 Adding segment details

Segment details can be added to the text boxes and saved into the database.

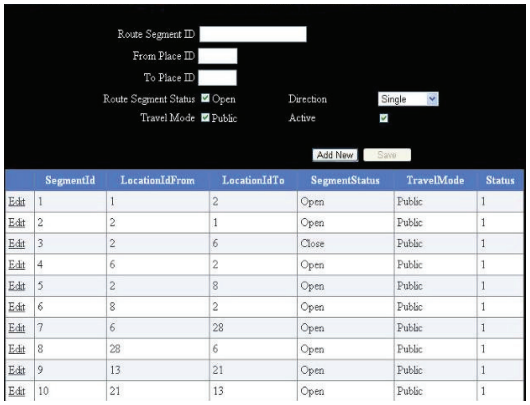

Figure: 5.2 – Segment details

5.3 Adding traffic details

Figure 5.3 shows traffic details page. Administrator has to select the specific segment number from a combo box. Then enter the time intervals, Traffic status, weather condition, construction situations and average speed to the table. If it is confirmed administrator can confirm the traffic details.

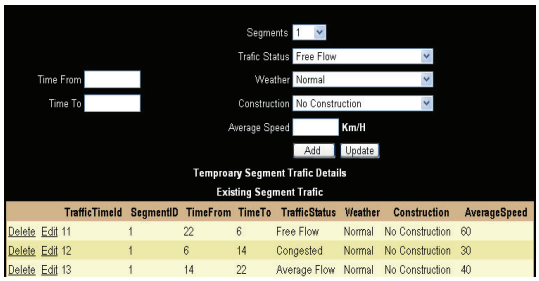

Figure: 5.3 – Traffic details

6. Future Improvements

Due to the limited time period for implementation, this project covers only the Colombo city. The system can be implemented further by adding bus routes of other districts and come up with a system for whole of Sri Lanka rather than only the Colombo city.

Private cabs and van service module were not included. Those details can also be added to make the site more informative.

GPRS access is not included. Those facilities can also be added to make the site more helpful for the users.

Further, this system can be improved to work as a mobile application.

7. Conclusion

The map section of the program is very useful for users who wish to view a specific area of the Colombo city. This section builds on the available e-Map web service provided by Google by allowing the user to zoom and shift the map to any desired location.

The Optimum Online Path finding section of the program provides accurate direction between a source and destination of road networks and the relevant bus numbers within the Colombo city. In addition, by providing satellite views of the location, the user can feel more comfortable and familiar with their routes before traveling to the location. The system considers 'one way', 'both way' and roads blocked on the road networks and send dynamic alert messages to display on the web.

The Optimum Online Path finding system is successfully implemented so that the public is made aware of the bus routes within Colombo city. The application is quick and robust, functioning at the type of speed expected from a modern application.

References

[1] http://www.transport.gov.lk/

[2] http://publish.uwo.ca/~jmalczew/gida_1 /Zhan/Zhan.htm

[3] http://www.husdal.com/2000/06/25/fastest-path-problems-in-dynamic- transportation-networks/

[4] MapQuest Advantage API Developer Guide, Java Interface Version, Version 5.2.0, October 2007

[5] C. Jun, GA-Based Path Finding for Public Transport Network, Dept. of Geo informatics, University of Seoul, Seoul, Korea -cmjun@uos.ac.kr

[6] Liang Dai , Fast Shortest Path Algorithm for Road Network and Implementation

[7] http://www.hot-maps.de/asia/sri_lanka/colombo/homeen.html

[8] http://www.tfl.gov.uk/

[9] http://www.mysrilanka.com/travel/lankamap/

[10] http://www.navteq.com

[11] http://www.mjharden.com

[12] http://www.esri.com/demographicadata

[13] http://www.aag.prg

[14] http://www.geo-imaging.com

[15] H.M. Deitel and P.J. Deitel, Net How To Program, http://www.proxix.com

[16] G. Booch, Object-Oriented Analysis and Design with Application

[17] Chris Hart, David Sussman & Johan Kauffman Beginning ASP.Net 2.0

[18] http://www.codeproject.com (A* algorithm)