Vehicle Routing and Road Traffic Simulation: A Smart Navigation System

John Garofalakis[1,2] Nakou Polyxeni[1] Plessas Athanasios[1,2]
garofala@ceid.upatras.gr nakou@ceid.upatras.gr plessas@ceid.upatras.gr

1 University of Patras, Computer Engineering & Informatics Department, Greece
2 Computer Technology Institute, Patras, Greece

Abstract

In-vehicle navigation permits people to find their way while driving. Combining these applications with road traffic data may result in diminishing traveling time. In this paper we deal with the problem of vehicle routing on today's congested roads. The result of this study is a web application that provides the user with two possibilities. The user can create a directed graph that corresponds to the city's road network and he can also find the optimal path that leads to its destination as far as distance and traffic congestion are concerned. As we didn't have real data about the traffic, we adopt a simulation technique to mimic actual traffic conditions.

Keywords: Intelligent Transportation Systems, In-Vehicle Routing Algorithms, Road Traffic Simulation, Traffic Congestion, Traffic Management Systems.

1. Introduction

Recent studies have shown that drivers in the largest 68 metropolitan areas in USA spend on average 40 hours per year stranded in traffic [Franzese et. Al. (2002)]. Intelligent Transportation Systems (ITS) and Traffic Management Systems try to provide passengers with efficient instructions, in order to avoid congested roads and reach their destination sooner.

Traffic Management Systems (TMS) are an ITS's functional area that is responsible for gathering information about traffic and deliver it to drivers. TMS involve detection, communication and control [Bok S. et. Al. (2004)]. The main idea is a surveillance system that detects traffic conditions and transmits the information to a traffic management center. The traffic management center processes the real-time traffic data and sends them to the drivers. The contribution of Global Positioning System (GPS), which give the position of the vehicle with accuracy of a few tens of meters, is of great importance for the development of ITS. The information flow between the vehicle and the traffic control center is shown in figure 1.

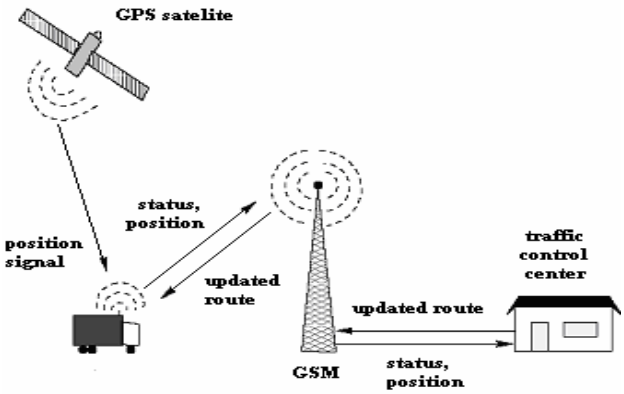

Figure 1. *Information flow between vehicle and traffic control center*

Intelligent transportation systems are information technologies that, when implemented in vehicles can lead to vast savings in journey time, improved driver safety and convenience, and reductions in energy consumption and pollution. Among the capabilities of current and future ITS technologies are the following [Bok S. et. Al. (2004)]:

- Traffic can be managed on the main roads of major metropolitan areas through (1) control of road junctions and access to major routes, (2) rapid detection of and response to incidents (e.g., accidents and heavy congestion), and (3) communication of advice to drivers and passengers about traffic conditions.
- A wide variety of information systems can be available to travellers (e.g., route guidance, nearby restaurants and other traveller facilities, travel time to the airport).
- Drivers and passengers can receive information by means of either eye-level "heads-up" displays or voice synthesizers installed within the vehicle.
- Drivers can be automatically alerted to the proximity of other vehicles or obstacles. Under some advanced ITS technologies, automatic steering and intervehicle spacing control both facilitate collision avoidance.

In this paper we concentrate on the usage of ITS in alleviating the annoying consequences of traffic congestion. More specifically we present a web application that drivers can use for finding the shortest way to their destination. The main characteristic of this application is that the computation of the shortest way takes into account not only the length of the road sections that constitute the final route but also the level of traffic.

We organize the rest of the paper in the following way. In section 2, we present related work to this field. In section 3, we analyze the concept of traffic simulation, which is incorporated in our study. In section 4, we describe the application and its

design principles. Finally, in section 5, we give our conclusions and the direction for future work.

2. Related Work

Several new technologies that have become generally available can help to solve the problem.
Computer programs to calculate routes for motorists have been available for several years. Microsoft AutoRoute [http://www.Microsoft.com/AutoRoute] comes in versions with route data for both Great Britain and Europe, and Vicinity Corporation's MapBlast [http://www.mapblast.com/] offers a similar online service in the United States.
However, both programs assume the capacity of roads remains static. The algorithms calculate a vehicle's speed solely on the basis of the class of road on which it is traveling. No attention is paid to variations in the traffic density with time.
A possible solution relies on the data offered by the Trafficmaster system which was developed in England. Trafficmaster [http://www.trafficmaster.co.uk/] has installed sensors every few kilometers that measure the speed of vehicles moving on the road. The information returns over a very high frequency (VHF) packet radio system to a control center for rebroadcast to users [Fawcett J. et. Al. (2000)].
An online service that offers information about traffic conditions, road conditions and accidents is the NAVIGATOR [http://www.georgianavigator.com], a website implemented by the Georgia Department of Transportation's.
However, special equipment and sometimes complicated infrastructure is necessary in order to use all these systems. Our approach presents a simple method to find the shortest route to a destination avoiding the congested roads.

3. ITS Technologies used to measure and manage congestion

The ITS technologies that local transportation agencies deploy to manage traffic in congested areas typically are ones that have gone through research and development and are readily available.
Some examples of ITS technologies that are used to address congestion is the monitoring of traffic conditions with closed-circuit television, sensors, automatic vehicle identification or video image processing, the metering of traffic onto freeways, the implementation of electronic money transactions on toll roads, the faster and anticipatory responses to traffic incidents, or the supply of information on travel conditions as well as alternative routes to travelers (dynamic message signs, traveler information phone number, web-based internet sites).
More specifically, the usage of wireless communications facilitates the communication between the vehicles and traffic management centers. Short-range communications can be accomplished using IEEE 802.11 protocols. The Global

System for Mobile Communications (GSM) can be used for longer range communications.
In addition, another technology is the floating cellular data or floating car data (FCD) [Lorkowski S. et. Al. (2003)], that means that the signals from the cellular phones that may be contained in a car and transmit their location information to the network, are converted into accurate traffic flow information. No infrastructure needs to be built along the road. The FCD technology provides great advantages over existing methods of traffic measurement, much less expensive than sensors or cameras.
Sensing technologies can also be used. Sensing systems for ITS can be either infrastructure based or vehicle based systems, or both. Infrastructure sensors are devices that are installed or embedded on the road, or surrounding the road. These sensing technologies may be installed during road construction and maintenance.
Inductive loop detection is another method for measuring the traffic. Inductive loops can be placed in a roadbed to detect vehicles as they pass over the loop by measuring the vehicle's magnetic field. The simplest detectors simply count the number of vehicles during a unit of time that pass over the loop, while more sophisticated sensors estimate the speed, length and weight of vehicles and the distance between them.
Traffic flow measurement using video cameras is another form of vehicle detection. Video from cameras is processed and the changing characteristics of the video image as vehicles pass are analyzed. As these technologies are not affordable, we decided to simulate the traffic data.

4. The Navigation System

4.1 Road Network Modeling

The modeling of road networks is strongly connected with graph theory [Hofmann et. Al. (2003)]. A road network can be represented as a directed graph. So, it is possible to design a graph that corresponds to that road network using the digital GIS data that represent the map of the city's road network.

4.1.1 Fundamental Definitions from Graph Theory

A graph [Gibbons (1985)], more specifically a directed graph, it can represent a map that depicts the road network of a city or of a country. A directed graph or digraph G is an ordered pair G: = (N, E) with a set of nodes N with |N| = n and a set of ordered pairs of nodes E with |E| = m, called directed edges. Each node of the graph represents an intersection or a terminal point of the roads. Each edge from one node to another represents a directed link between two adjacent intersections or between an intersection and a terminal point. A road of single direction is represented with a directed edge to the particular direction while a road of double direction is represented with two edges to both directions (Fig.2).

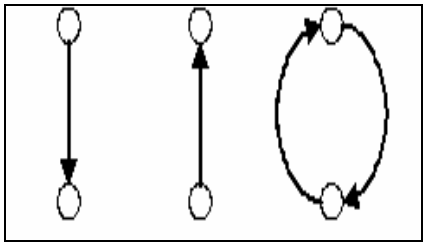

Figure 2. *Single and double direction road representation*

A graph can also be weighted. In this case each edge of the graph has a real number that corresponds to the transition cost from one node to the other through one edge. This value is called cost or weight. The cost describes how difficult is to transit from node to node. The cost can be translated as the traffic that exists on a road or as the distance between two intersections. In certain practical problems, cost can be negative, e.g. a vehicle that is moving losing or gaining money.
A path from a starting node to a destination node is a sequence from adjacent nodes. Adding up the costs of all edges of the path, results the total cost of the path. The path with minimal total cost from all the likely paths between the starting and final node is called shortest path.

4.1.2 Graph and Road Network Representation

Apart from the obvious geometrical representation of the graphs, an alternative representation is needed, that can be used by computers and algorithms. There are two very common and widely acceptable representations: adjacency matrix and adjacency lists.

- ***Adjacency Matrix:*** In this representation, the graph is stored in an $n \times n$ matrix M, where every row and column corresponds to a vertex in G. Let $M_{i,j}$ denote the (I,j) entry of M. If the edge (I,j) is an edge of the graph, then $M_{i,j} = 1$, otherwise $M_{i,j} = 0$. In case that G is a weighted graph, we construct an additional $n \times n$ matrix C to store the weights (costs) of the edges. The adjacency matrix has n^2 elements, out of which only few are non-zero. Hence, this representation is suitable only if G is very dense.
- ***Adjacency Lists:*** The adjacency list of a node n in a graph G = (N,E) is defined as the set of nodes that can be visited from the node n. There is an additional list that contains the transition costs of the edges and one array of length n for storing the head of the list. Hence, it consists of $\leq n + 2m$ elements and it is therefore a more efficient solution, especially when the graphs are sparse.

We have chosen the second graph representation which is more efficient in the case of road network representation [Gibbons et. Al. (1985)].

4.2 Traffic Simulation

Something that must be further explained is the method of traffic simulation. ITS that take into account traffic level on the roads, use data that may derive from sensors that are installed along the roads, or from radars, or from information that drivers provide. As we didn't have in our disposal this information and the cost of the appropriate infrastructure is high, we decide to simulate the traffic that is appearing on the road at a certain time of the day using a very simple concept. The main advantage of this concept is that there is no need for any specific equipment, as simulation is only based on the capabilities that the programming language offers. This choice allows the later use of data that derive from the mentioned above traffic measurement technologies.

There are many ways for measuring the congestion on a road such as he number of vehicles per mile on a road divided by the number the road was designed to bear, or the speed of vehicles compared to the speed limit for that road. Further options include the length of queues of traffic traveling at similar speeds. In practice, most drivers want to know how much longer a journey is likely to take under the prevailing conditions, compared with traveling on an empty road [Fawcett J. et. Al. (2000)].

We prefer to use as a weight of the graph's edges a measure that combines both traffic's value and roads' length. Assuming that traffic determines the speed of vehicles and knowing the length of the roads, we use as transition cost from one node of the graph to the other the total time that a vehicle needs to travel all this distance. Regarding also that the speed of the vehicle is constant during the transition from one node to the other and the vehicle is moving in a straight line during a time interval *t*, from the fundamental law of physics, the necessary time for traveling from one node to the other is described by the formula: t=s/v. The length *s* of the road can be easily calculated by the formula

$$s = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2},$$

which gives the distance between two points. The point with coordinates (x_1, y_1) corresponds to the starting node of an edge and the point with coordinates (x_2, y_2) corresponds to the final node of this edge.

For the simulation of vehicle's average speed we used a PHP function which returns random values between an interval (a, b). As the permissible speed in a city varies from 5 to 50 Km/h, which is from 1m/sec to 14m/sec, the function returns values between the interval (1, 14).

4.3 In-Vehicle Routing Algorithms: Dijkstra's algorithm

ITS use a variety of algorithms in order to find the optimal route from a starting point to a destination that means the shortest path connecting two specific nodes of a directed graph.

Dijkstra's algorithm [Dijkstra (1959)] solves the single-source shortest path problem for a directed graph with nonnegative edge weights. In graph theory, the single-source shortest path problem is the problem of finding a path between two vertices such that the sum of the weights of its constituent edges is minimized.
The algorithm works by keeping for each vertex *v* the cost *d[v]* of the shortest path found so far between s and v. Initially, this value is 0 for the source vertex s, and infinity for all other vertices, representing the fact that we do not know any path leading to those vertices.
The algorithm guarantees to terminate in all circumstances. The asymptotic cost is $O(n^2)$ for a graph with n nodes.
Bellman-Ford, A*, Lee, Soukup are some other algorithms that solve the shortest path problem.
Among the existing shortest path algorithms we have chosen the Dijkstra, which is often used in such problems, and its pseudocode is presented below. The results of this algorithm are satisfactory enough and implementing it with the optimizations mentioned before, the improvement of running time and memory space is possible [Noto et. Al. (2000), Eklund et. Al. (1996)]. The Dijkstra approximate bucket implementation is best when the maximum edge weight is less than or equal to 1,500. Both Dijkstra approximate bucket and Dijkstra double bucket are best when the maximum the maximum is greater than or equal to 1,500 [Miller et. Al. (2001]. The pseudocode of original Dijkstra algorithm is presented below.

```
V : το σύνολο όλων των κορυφών
T  : το σύνολο των κορυφών με προσωρινή ετικέτα
P  : το σύνολο των κορυφών με μόνιμη ετικέτα
N : το σύνολο των γειτονικών κορυφών μιας κορυφής
us : αρχική κορυφή
lj  : ετικέτα της κορυφής uj
pj  : πρόγονος μιας κορυφής uj
cij  : κόστος μιας ακμής eij

ls :=0;  T ← { us };  P ← { };
FOR uj ∈ V \{ us } DO  lj := ∞  END FOR
WHILE  T ≠ { }  DO
   ui := uj ∈ T with min lj;
    P ← P + {ui};  T ← T \ {ui};
   FOR  uj ∈ N(ui)  DO
      IF (uj ∉ T)  ∧  (uj ∉ P)  THEN
        lj := li + cij ; pj := ui ;
        T ← T + {uj};  END IF
      IF (uj ∈ T)  ∧  (li + cij < lj) THEN
        lj = li + cij ; pj := ui ;  END IF
   END FOR
END WHILE
```

4.4 Technical Details

For the development of this application we have used PHP (version 5.0.4) as a network programming language with the GD (Graphics Draw) library enabled. GD is a library that offers the possibility to draw graphics easily and quickly. We have used this library's functions in order to draw the map images. We decided to use raster images, when drawing the map image and more specifically the GIF format because of the smaller image size produced in the case of our implementation.
The graphs that model the map of a city's road network are stored in a database and then they are available to the user whenever he wants to find his route in this city. Apart from the adjacency lists that represent the graphs, in the database are also stored the coordinates of the nodes. For this purpose we have used MySQL (version 4.1.10a-nt).
In the part of the application, the user needs a browser, for example Mozilla Firefox or Internet Explorer, to access it, as the content of the application is HTML.
With reference to the geographic data that we had in our disposal, the file type of the maps was the MIF (MapInfo Data Interchange) type, supported by MapInfo. MIF has a very comprehensive structure that facilitates the development of the map file parser.
Finally, we have used JavaScript for drawing the nodes and the vertices of the graph on the city's selected map. For this purpose we have used an available graphics library [JavaScript Library for Graphics http://www.walterzorn.com] from the web that contains functions for drawing lines and circles.

4.5 Map Drawing

As we have already mentioned, the MIF files contain the coordinates of the points that corresponds to the roads. In order to draw the map, we have developed a parser that reads the coordinates from MIF files and stores them in PHP arrays. Then, the appropriate mathematical transformations must be applied, in order to transform the coordinates from the original coordinate system to the reference system that corresponds to the image displayed in the screen.

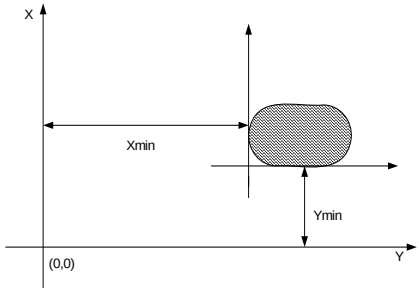

Figure 3: *The first transformation of the coordinates*

In figure 3, the shadowed area corresponds to the region that the geographic data cover in the original coordinates system. As shown, the transformation is a tranfer of the start of the axis. If x and y are the coordinates of a point in the shadowed area, x_{min} the minimum abscissa and y_{min} the minimum ordinate of the shadowed area, then after the transformation the new coordinates are:

$$x' = x - x_{min}$$
$$y' = y - y_{min}$$

Afterwards a new transformation must be applied, so that the point coordinates are transformed to image coordinates that is pixel coordinates. In this point, we have to mention that for computer screens and for digital images, as start of the axis is considered the upper left corner and not the lower left corner. As shown in figure 4, the maximum abscissa for the area of interest is x_{max}, the maximum ordinate is y_{max} and the start of the axis is the point (x_{min}, y_{min}). The width of the image must be $x_{max} - x_{min}$ and its height $y_{max} - y_{min}$. Considering as start of the axis the upper left corner the new coordinates of a point that correspond to the image are:

$$x'' = x' = x - x_{min}$$
$$y'' = (y_{max} - y_{min}) - y'' = y_{max} - y$$

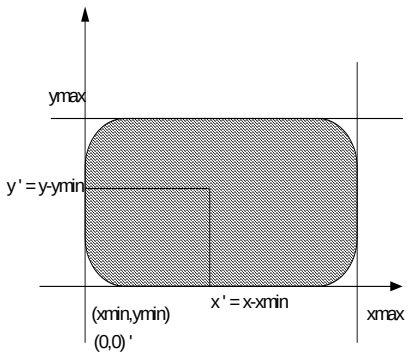

Figure 4: *The second transformation of the coordinates*

These coordinates are multiplied or divided with a factor according to user's preference for larger or smaller image.

Having the transformed coordinates in our disposal, we then use them to generate the image of the map, using the functions of GD library. We should also mention that the names of the roads are retrieved from MID files, which are files that contain attributes related to the coordinates of MIF files.

4.6 Functionalities of the Smart Navigation System

The application we haωε developed offers to the user two possibilities. First of all, the user can use a graphic tool and design the roads and their directions and furthermore can proceeds in corrections each time that changes occur in the road network, such as pedestrianizations or changes in the direction of certain roads. Secondly, there is a tool that allows the user to find the shortest way to a desired destination.

4.6.1 Graphic Tool application

First of all, a graph of the road network must exist, which later will constitute the input of the shortest path algorithm. In the page of graph construction, as shown in figure 5, the map of the city is displayed and the user can easily place the nodes where he wants. Every second click that the user makes on the image, a directed line is designed on the image. An arrow is also designed that indicates the direction of the road. If the road is of double direction the user must draw two arrows to both directions.
The coordinates of the points (nodes) where the user has made a click and their serial number are stored in a JavaScript array. The coordinates of the ends of the edges are also stored in another array.

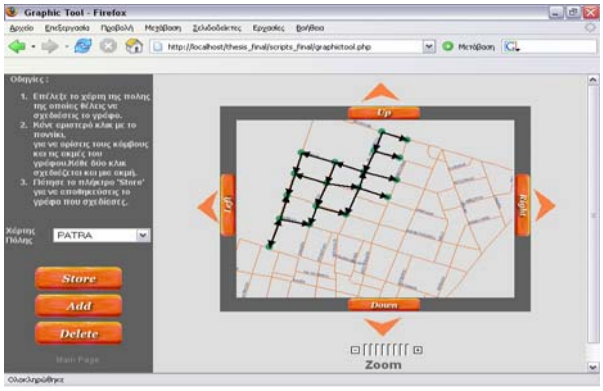

Figure 5: *Graph construction*

When the user finishes with the graph's construction, he is then able to store the graph in order to be available for later use. Before storing the graph in the database, the adjacency list is constructed using the arrays of nodes and edges.
There is no need for designing the graph for the entire road network from the beginning. The user can leave the graph incomplete and extend it with additional edges whenever he wants (figure 6). He can also proceed to modifications in case of changes in the road network deleting and then adding new edges.

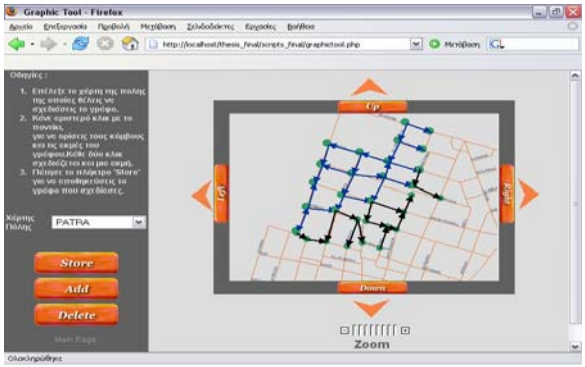

Figure 6: *Graph extension*

4.6.2 Route Planner application

The Route Planner application computes the shortest path in terms of travelling time. The travelling time depends on the length of the roads and the traffic level on the roads and it is computed with Dijkstra's algorithm.

The user must place two points on the image of the map. The first point is the point where the user is and the second point is the destination that he wants to reach. Then, the user can press a button and the shortest route will appear on the map. Before the shortest path algorithm runs, it is defined to which nodes of the graph correspond the starting and terminal point that user gives. The starting and terminal node together with the adjacency list of the graph and the list of the costs are passed as arguments to the algorithm and the shortest path is computed and presented to the user (figure 7).

Apart from the shortest path, it is also presented to the user a notification about the travelling time. This value is calculated by adding the costs of the edges that constitute the path and may change according to the traffic level.

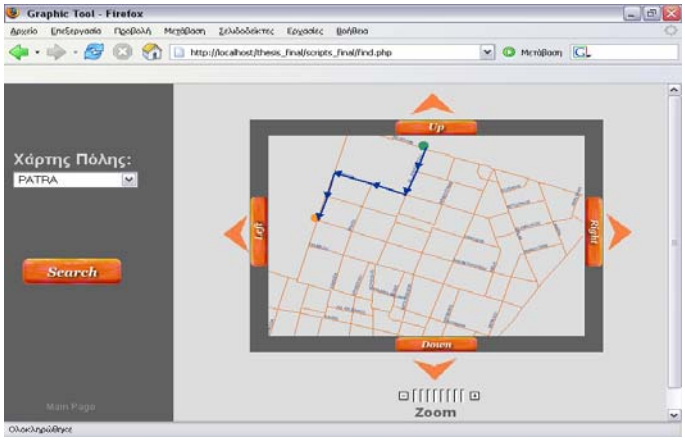

Figure 7: *A shortest path for the previously created graph*

5. Conclusions and Future Work

The application that was implemented is only one component of navigation software which can be used for finding the shortest route in a city. It empowers the user to design and modify the graph of the road network himself and computes the shortest route taking into account the traffic level on the roads. In this application, we simulated the real data. The more interesting is that, as the traffic's value is simulated, there is no need for any specific equipment and suitable infrastructure. However, one of our future purposes is to replace this value with real and more consistent traffic data.

This application can be improved in order to use data from a GPS receiver and to determine automatically the vehicle's position. It is also possible to collect historic data about the traffic on a city's roads and store them in a database. These data may depend on the day and the hour of the day and can be used by the shortest path algorithm instead of the values generated by the PHP function. Whilst not a commercial product, the software has demonstrated that adaptive computerized route generation could help drivers cope with the growing congestion on their city's roads and can be proved useful in minimizing the delays resulting from queues of traffic.

References

Bok S., Keon S. (2004). Introduction of Intelligent Transportation Systems.

Dijkstra E.W. (1959). A Note on Two Problems in Connexion with Graphs. Numerische Mathematik , p.269-271

Eklund P.W. Kirkby S., Pollitt S (1996). A dynamic multi-source Dijkstra's algorithm for vehicle routing. Intelligent Information Systems, Australian and New Zealand Conference.

Fawcett J., Robinson P. (2000). Adaptive Routing for Road Traffic. IEEE Computer Graphics and Applications.

Franzese O., Joshi S. (2002). Traffic Simulation Application to Plan Real Time Distribution Routes. Proceedings of the 34th conference on Winter simulation: exploring new frontiers San Diego, California

Garofalakis J., Michail T., Plessas A. (2006). The m-CHARTIS System. IW3C2, Edinburgh UK.

GD Graphics Library www.boutell.com/gd

Gibbons M. Alan (1985). Algorithmic Graph Theory. ISBN 0521288819

Harvey J. Miller, Shin-Lung Shaw (2001). Geographic Information Systems for Transportation, ISBN: 0-19-512394-8

Hofmann-Wellenhof, Legat, Wieser (2003). Navigation-Principles of Positioning and Guidance, ISBN: 3-211-00828-4

Javascript Library for Graphics http://www.walterzorn.com/

Lorkowski S., Mieth P., Schäfer R. (2003). New ITS applications for metropolitan areas based on Floating Car Data.
MapInfo Professional User's Guide, Mapinfo Corporation (2005)
Microsoft AutoRoute http://www.Microsoft.com/AutoRoute
NAVIGATOR, Georgia Department of Transportation – Intelligent Transportation Systems. http://www.georgianavigator.com/
Noto M., Sato H. (2000). A method for the shortest path search by extended Dijkstra algorithm. IEEE International Conference.