ICNavS: A Tool for Reliable Dynamic Route Guidance

Journal:	*IET Intelligent Transport Systems*
Manuscript ID:	ITS-2006-0066.R2
Manuscript Type:	Research Paper
Date Submitted by the Author:	17-Apr-2007

Complete List of Authors:	Kaparias, Ioannis; Imperial College London, Centre for Transport Studies, Civil and Environmental Engineering Bell, Michael; Imperial College London, Centre for Transport Studies, Civil and Environmental Engineering Chen, Yanyan; Beijing University of Technology, Transportation Research Center Bogenberger, Klaus; BMW AG, Corporate Quality ☐ Product Monitoring

Keyword:	HUMAN MACHINE INTERACTION, IN-VEHICLE, TRAVELER INFORMATION

ICNavS: A Tool for Reliable Dynamic Route Guidance

Ioannis Kaparias[1], Michael G.H. Bell[2], Yanyan Chen[3], Klaus Bogenberger[4]

1. Centre for Transport Studies, Imperial College London, Skempton Building, London SW7 2BU, UK
Email: ik00@imperial.ac.uk

2. Centre for Transport Studies, Imperial College London, Skempton Building, London SW7 2BU, UK
Email: mghbell@imperial.ac.uk

3. Transportation Research Center, Beijing Univ. of Technology, 100 Ping Le Yuan, Beijing 100022, China
Email: cdyan@bjut.edu.cn

4. Department of Corporate Quality – Product Monitoring, BMW Group, AQ-11, Munich 80788, Germany
Email: klaus.bogenberger@bmw.de

Abstract

The aim of this paper is to devise a new reliable dynamic route guidance approach by integrating the A* algorithm, the concept of reliability and an existing route guidance method into a single package. A new purpose-developed software tool, the Imperial College Navigation Software (ICNavS), is presented, so as to implement and demonstrate the new approach on a real road network, using simulated data. A summary of the background of the program is given, followed by a procedure developed in order to model the features of real road networks, as well as missing data. Then, a simulation experiment on a part of West London's road network is carried out and the results are presented.

1. Introduction

As the usefulness of in-vehicle information systems is increasingly appreciated, more drivers install them in their vehicles. Having started off with luxury makes and models, they are gradually spreading through the entire vehicle fleet, proving that their future is promising. An important feature, offered by more sophisticated systems, is route guidance. Route guidance is called dynamic when the computed route takes into account the current traffic conditions.

The objective of conventional dynamic route guidance is to provide participating drivers a fast route to their chosen destination. This study aims at taking this objective a step further and incorporate reliability into route guidance, so as to develop an advanced system, which does not only give simple route directions to the driver, but also avoids possible sources of delay. Thus, reliable dynamic route guidance is defined as the feature, whose objective it is to provide participating drivers a reliable (risk-averse) route to their destination. It should be noted, that risk in this context is defined as the probability of encountering congestion and does not refer to road safety. Consequently, a reliable (risk-averse) route is one which, under travel time uncertainty (the rule in real road networks), minimises the probability of experiencing delays, by avoiding the potential sources of delay. These are for example right turns in the UK (left turns in Continental Europe and North America), where the turning vehicle usually has to cross one or more opposing streams, resulting in increased waiting times at junctions.

The research work presented here is carried out as part of the OFFENSIVE project, in collaboration with BMW, and involves providing reliable dynamic route guidance to the driver in two forms, namely autonomous route guidance and supported route guidance, each requiring the corresponding system architecture. More specifically, autonomous route guidance includes a computation of a set of alternative routes based on static (historic) link travel time profiles, complemented by information on traffic incidents broadcast to the vehicle using the Traffic Message Channel (TMC), which is part of the Radio Data System (RDS) for broadcasting digital data on FM channels. Network data is provided in the form of a DVD and the route computation takes place in the vehicle itself. As opposed to that, supported route guidance involves continuous communication with a Traffic Information Centre (TIC). The route computation takes current network conditions into account and is carried out in the TIC, with the set of alternative routes computed being transmitted to the vehicle either as a whole, or in parts.

The aim of this research study, which focuses on the autonomous route guidance scheme, is to integrate a number of methods into a single package, such that a novel reliable dynamic route guidance strategy can be devised, implemented and demonstrated. The new strategy is based on three methods: the A* path finding algorithm, the concept of reliability and Chen's link penalty method. For its implementation, a software tool called ICNavS (acronym for Imperial College Navigation Software) is developed; its demonstration occurs through an experiment on a real road network, using simulated dynamic traffic data.

The present paper is structured as follows: Section 2 presents the background of the research reported here, while Section 3 describes the reliable dynamic route guidance approach used. Section 4 describes the modelling of some features of real road networks, so as to enable the application of the proposed approach. The structure, user interface and functions of ICNavS are presented in Section 5, while Section 6 introduces the data simulation procedure carried out in this study, for the purpose of conducting a preliminary experiment. The experiment carried out and the results obtained are given in Section 7, while Section 8 concludes the paper.

2. Background

This section gives a brief review of previous relevant research and describes the background of the work presented in this paper. The A* shortest path algorithm and the concept of travel time reliability are reviewed.

2.1 The A* shortest path algorithm

Finding the shortest path in a network is one of the most frequently encountered problems, not only in transportation engineering, but also in computer science and operations research. Although various algorithms exist for finding the shortest path, their performance tends to significantly deteriorate with increasing network size. In route guidance, finding the shortest path is a subroutine that needs to be called very often. Due to the fact that the size of transportation networks is usually large, it is of vital

importance to have a shortest path algorithm, which is efficient enough to produce accurate results in little computation time.

It should be noted here, that the term shortest path does not necessarily refer to the distance, but to the variable to be optimised. In the case of route guidance, one is usually not interested in the least distance path, but is more concerned with the least travel time path. The link weights used for this computation are travel time values, rather than distance values. Therefore, in the problem formulation, the variable to optimise (minimise) is the total travel time; the path achieving this is referred to as the shortest path in the rest of the paper.

While a large number of methods for finding the shortest path between two points in a network exist and are comprehensively appraised by Ahuja et al [1], the leading algorithm for finding the shortest path from an origin to a destination is the A* algorithm [2]. The advantage of A* compared with other shortest path algorithms, such as Dijkstra's algorithm [3], is that A* is much more efficient, due to its ability to convert an uninformed search into an informed search. The A* algorithm uses a heuristic function to estimate the cost (travel time) from any point to the destination node of the network. This is often the airline distance to the destination. The shortest path is eventually found, under the condition that the heuristic does not overestimate the actual distance to the destination. The number of nodes expanded is generally much smaller than for other algorithms, making A* more efficient.

The concept of A* is summarised as follows; the algorithm holds two lists, the open list and the closed list. The closed list contains all the nodes of the network that have been expanded, whereas the open list contains all the nodes that may be expanded at the next step. At each step, one node is expanded and moved from the open list to the closed list, while its successors are placed into the open list. For every node n, $f(n) = g(n) + h(n)$ is calculated, where $g(n)$ is the travel time from the origin to n and $h(n)$ is the estimate of the travel time from n to the destination. The node to be expanded at each step is the node with the lowest $f(n)$ value among the nodes in the open list. When $h(n) = 0$ for all nodes n, A*

reduces to Dijkstra's algorithm. Moreover, the efficiency of the algorithm increases, the closer $h(n)$ is to the actual value.

2.2 Travel time reliability

Travel time uncertainty has been identified as one of the important factors affecting travellers' decisions. Travellers are interested in how long it will take them to reach their destination, but are even more concerned with the reliability of their prediction of total travel time. A wrong travel time prediction results in either an early arrival at the destination or in a delay. None of these situations is appreciated by the traveller, with delays usually having more severe consequences for him/her (e.g. late arrival at the workplace) and therefore not being tolerated.

Both empirical and analytical studies have been conducted, all of them demonstrating the importance of travel time uncertainty to travellers [4-9]. Models have been devised to measure the uncertainty encountered by a traveller during his/her trip and express it as reliability. Comprehensive reviews of this topic are given in [10] and [11].

Much research has focussed on defining adequate measures for quantifying travel time reliability. Most measures use the characteristics of the travel time distribution, such as measures attempting to quantify the amount of delay that may be experienced along a link and measures indicating the probability that a link may be unusable due to congestion.

In many studies, measures of the first category are suggested and implemented [7,12,13]. A range of similar measures are presented and appraised in [14], where most of the measures described involve quantities such as the mean, median, standard deviation and 90^{th} percentile of the travel time distribution, to define upper and lower bounds for the travel time experienced on a link. Other measures developed in other studies also consider the skew of the distribution [15].

However, measures of the second category have been more widely used in transportation applications so far. Several probability based reliability measures having been developed over the years, such as the one defined by Bell and Iida [16], which is expressed as "the probability of a link to be uncongested". Bell and Iida assume that the condition of traffic flow on a link is binary, "normal" or "abnormal", and that the link is either congested or uncongested. They then consider two aspects of transport network reliability, namely connectivity reliability and travel time reliability. The former indicates the probability that traffic can reach its destination while the latter depicts the probability that traffic can reach its destination within a given time (or cost, etc.).

In this study, the reliability measure introduced by Bell and Iida is used, so reliability is defined as the probability that the travel time on a link is greater than a preset threshold value. As a consequence of the definition, the reliability of a link can only take values between 0 and 1. Extending the definition of the reliability of a link, the reliability of a path is simply the product of the reliabilities of the series of links constituting the path in the absence of "failure dependence" (inter-link abnormality correlations).

3. Methodology

In studies [17-19] and further in [20], a new reliable dynamic route guidance strategy is developed, making use of the concept of reliability, as defined by Bell and Iida, and of the A* algorithm. The objective of the new method is to compute and provide to the driver a set of routes, which, while not necessarily being the fastest or the shortest, are acceptable to him/her and are equivalently reliable. This is ensured by applying a number of constraints, namely a maximum path duration constraint and a maximum path length constraint. Every new path that is computed is checked against these and it is only accepted if it satisfies them. If the constraints are not met, the path is discarded and a new path is sought.

The constraints introduced are enforced by the application of link weight (travel time) penalties. The

main idea lies in performing an initial search in order to find the quickest path without considering reliability, penalising unreliable links by applying travel time penalties and running subsequent searches, each time reducing the penalties, until the computed path satisfies the constraints imposed. More specifically, unreliable links are initially excluded from subsequent path computations, provided this is possible; however, if a path satisfying the acceptability constraints cannot be found, unreliable links are progressively re-included in the search. The sequence in which this occurs depends on their reliability.

The idea of link penalties is also introduced as an attempt to develop an efficient and acceptable method for finding alternate paths in a network. Routing strategies suggesting multiple alternative paths to the driver have been proven to be superior to single path solutions, especially when the market penetration (proportion of equipped vehicles in the entire vehicle fleet) of the route guidance system is high [21]. The reason for this is that suggesting only one path to all participating drivers increases the risk of congestion feedback, which results in all vehicles being guided along a single path and thus causing congestion on a previously uncongested road.

However, where more than one path is computed, it has to be guaranteed that they will share as few links as possible (will be maximally disjoint), in order to reduce the probability of joint path failure as much as possible. This is enforced by additionally applying penalties to links already used, i.e. links that are already included in a path that was previously computed and accepted, ensuring that used links will be initially excluded from subsequent path computations; however if no acceptable path then exists, they will be progressively re-included by having their penalties reduced.

A third constraint, associated with the partially disjoint paths is introduced, and this is the maximum path overlapping ratio constraint, ensuring that a path can only be accepted if it does not overlap too much with any previously computed path. Finally, to ensure that the algorithm will terminate and will not carry on computing an infinite number of paths, a maximum number of paths to be computed is

imposed as a constraint. Of course the algorithm terminates earlier if only a smaller number of acceptable paths are found.

The following travel time penalty is used in Chen's link penalty method, for links having a reliability value lower than a preset threshold and for links that are already included in one of the computed paths:

$$\Delta t_i = \alpha^m (1 - r_i)^q W_0$$

with $0 \leq \alpha < 1$, m = number of iterations, $q = 0$ if $m = 0$ otherwise $q = 1$, and W_0 = a value large enough to bring about link exclusion.

The fact that travel time is not constant at different times of the day and on different days is also considered in the original presentation of the method [18], as well as in a modification made to it [20]. An interesting feature of both the original and the modified approach is the fact that in the multiple runs of the A* algorithm carried out, the search occurs in different directions (origin to destination and destination to origin), such that previously computed information can be used in subsequent runs, thus significantly reducing the overall computation time. While an initial reverse A* run followed by subsequent forward runs is executed in the original approach, in the modified approach, which is the one used in this paper, the initial run is carried out forwards, followed by a series of reverse runs. An outline of the modified link penalty method is shown in Figure 1.

[Figure 1 goes here]

The method adopted has the main advantage that it can yield good heuristic solutions, whilst at the same time being flexible, simple and efficient. The ability of the A* algorithm to convert an uninformed search into a partly informed one, while at the same time making use of previously computed information, reduces the total running time of the algorithm significantly. By finding the maximum penalty that can be applied to an unreliable link while still delivering an acceptable route

using the above expression, the algorithm is seeking the *maximum feasible disutility (or, more colloquially, "dislike") that can be applied to link unreliability*. While this is not the same as seeking the most reliable route, the conjecture is that this is a useful heuristic in a context of limited information about travel time distributions and limited computing power. The task of finding the most reliable route would require knowledge of link travel time distributions and their correlations, which must then be convoluted to produce route travel time distributions, an extremely computationally demanding task [22-25].

The penalties applied to unreliable links are reduced by a fixed amount between two iterations, which may result in better paths than the one found being missed out. An exact method would require the penalties to be alternately increased and reduced by smaller amounts each time, so as to converge to optimal paths. However, performing such a procedure would significantly increase the running time of the algorithm, which thus implies that there is a trade-off between efficiency and accuracy.

In spite of this trade-off, the method is still a very efficient way of providing the driver with an acceptable route that avoids unreliable links as much as possible. Further research is currently being undertaken on the topic of learning user preferences so as to incorporate these in the approach and so devise a personalised reliable dynamic route guidance strategy [26,27].

In the following section, some features of real road networks are modelled, so as to enable the application of the approach described here.

4. Route guidance in real road networks

Artificial grid networks are very convenient to carry out initial tests, however they are too idealised and cannot be used to model real road networks. In order to run simulation experiments on real networks, a software tool (ICNavS) is developed and used. The software tool is described in detail in

Section 5 of this paper. This section presents the methods involved in modelling the features of real road networks.

The features of real road networks that are discussed here are the following:

1. One-way roads and dead-ends
2. Turn restrictions
3. Positioning of the vehicle

4.1. One-way roads and dead ends

Artificial grid networks are usually undirected graphs, meaning that all links can be traversed in any direction. However, this is not the case for real road networks, where due to the presence of one-way streets, all links have to be directed. Additionally, it is very common that a two-way road has different travel times in opposing directions. Therefore, separate links are introduced for every possible connection from one node to another; two-way roads are represented by two opposing links.

Dead-ends are another feature of real road networks which do not exist in grid networks. Taking into account the fact that dead-ends are always two-way roads, as otherwise one would not be able to drive in and then out, it has to be ensured that they are represented by two links of opposite directions.

4.2. Turn restrictions and intersection representation

The main feature, which differentiates real road networks from artificial grid networks, is the existence of turn restrictions. Right turns in the UK (left turns in Continental Europe and in the US) are very often banned, due to the fact that there is not adequate space to accommodate the formation of the resulting queue of turning vehicles.

Several attempts to model turn prohibitions have been made in the past. The first method was

proposed by Wattleworth and Shuldiner [28]. The approach adopted was to substitute every junction by a smaller sub-network of dummy nodes and links. Despite the fact that this approach alters the network structure and reduces efficiency, it is the most widely used method in transportation applications so far.

An important contribution to the field was made by Kirby and Potts [29], where a review of existing techniques was carried out and a new method was proposed, according to which penalties are applied to turning movements. For prohibited turns, the corresponding penalty value is set to infinity. Ziliaskopoulos and Mahmassani [30] introduced the Extended Forward Star Structure, according to which there is a list with all allowed movements in the network and every node holds as many labels as links emanating from it. Shortest path algorithms can then be executed on this modified network structure, while all labels are updated gradually.

Similarly to the Extended Forward Star Structure, the approach used here involves holding a list of all allowed intersection movements in the network database, each one of which is represented by three nodes: start-node – middle-node – end-node. Nevertheless, instead of placing multiple labels on each node, the labels are placed on the links, such that each link holds two labels, one at its start and one at its end. The A* algorithm is modified accordingly, such that it keeps two open lists (open-start and open-end) and two closed lists (closed-start and closed-end) containing links instead of nodes. Thus, the algorithm proceeds in a link-based manner, updating the start- and end-labels of every link visited and finally tracing back the shortest path. The advantage of this method is that it enables the application of label-correcting algorithms on a network, not only eliminating the need for creation of extra network elements, but even involving just one type of elements (links) instead of two (links and nodes), thus improving efficiency significantly.

4.3. Positioning of the vehicle

In artificial grid networks, the origin and destination of a route, as well as the position of the vehicle, are identified as nodes. To be exact, any journey always starts from an origin node and ends at a

destination node. However, in real road networks, where the nodes correspond to junctions, this approach has a serious drawback, and that is the fact that not only the position of the vehicle is required, but also its direction. It is possible, that a vehicle is located on a road between two junctions, facing towards one of them and not being able to make a U-turn. However, if the position of the vehicle is expressed as a node, it is very likely that two alternative routes would guide the vehicle to make illegal or impossible movements.

A solution to the problem presented here is to slightly modify the network structure and make it link-based rather than node-based. Namely, when specifying the origin and destination, an origin-link and a destination-link are specified, considering that the origin or the destination respectively are located somewhere on this link. Using this technique, the setting of the origin link or the current position link immediately limits the next allowed movements. Similarly, the setting of the destination link results in the direction of approach of the destination being fixed, which is very important in some cases. In any case, when the user is prompted to enter the destination of his/her trip, it is most likely that this will be a street name, therefore only referring to a link.

5. Description of the software

ICNavS (Imperial College Navigation Software) is a software tool enabling the execution of route guidance algorithms on real road networks. It is written in Visual C#.NET object-oriented programming language. A description of the structure, the user interface and the functions of ICNavS is given next.

5.1. Structure

There are two classes, one for nodes (called “Nodes”) and one for links (called “Links”), each one of them holding a set of properties. Additionally, there is a class for movements (called “Movements”) and a class for paths (called “Paths”) likewise holding a number of properties. Properties can be categorised into static and runtime. Static properties are the properties of a network element, which are

not modified during the execution of a route guidance algorithm. Such are, for example, properties relating to network topology and geometry, such as location. As opposed to that, properties that are altered during the execution of a route guidance algorithm are called runtime properties.

Every object of class “Nodes” (i.e. every node in the network) only holds the following static properties: location (X and Y co-ordinates with respect to a given origin), name and ID number. Its main purpose is to define the start and end locations of “Links” objects. Also, every “Nodes” object is associated with a circle image, representing the node in the visual interface of the software. All the nodes of the network are stored in a list and are recalled when needed using their ID number.

On the other hand, every “Links” (i.e. every link in the network) object holds the following static properties: name, ID number, start node, end node, road type, length, speed, travel time and reliability. Additionally, it holds a number of runtime properties, namely; six labels for the A* algorithm – f, g and h for the start and end of the link, a property holding the updated value of the travel time following the penalty application and a pointer indicating the link where the search came from in case the link has been visited by the algorithm. Every “Links” object is also associated with the image of a line, representing the link in the visual interface of the software. All the links of the network are stored in a list and are recalled when needed using their ID number.

To keep track of the allowed movements in the network, the “Movements” class is created. Each “Movements” object holds the following static properties: start-node, middle-node, end-node, type (right, left or straight-on), delay and reliability. It also possesses a runtime property holding the updated value of the delay following the application of penalties. Similarly to nodes and links, movements are stored in a list in the network database and are recalled when needed using their start-, middle- and end-nodes.

In order to store the computed paths, a class called “Paths” is formed. A “Paths” object only holds the following properties: ID number, list of member links, total travel time, total reliability, total length

and total number of right and left turns. These are all runtime properties, as they can only be modified by the route guidance algorithm.

5.2. User interface

Describing the user interface of ICNavS, which is shown in Figure 2, it consists of a main window with a canvas, where the network is inserted, buttons for each application, a box showing the properties of the selected item, a list box displaying the set of the computed paths and another box, displaying the properties of the selected path in the list box. By selecting the appropriate mode using the buttons, the user can draw a network on the canvas by adding nodes and links, edit their properties, edit the allowed movements, remove elements and zoom in and out to obtain the most desirable view of the network.

[Figure 2 goes here]

The user can additionally place a road map as a background and draw the network on top of it, so that the created network matches the real network. For memory requirement minimisation purposes, the background can be enabled and disabled at any time. The background can also be scaled in order to obtain the right distances. The created network can be saved to a file, through which it can be loaded at any time.

When a calculation has taken place, the computed paths are shown in the list box and are displayed on the canvas by different colours and patterns. The selected path from the list box is shown more opaque than the other paths and its properties are displayed in the box below the list box. The user must then choose one of the computed paths to reach the specified destination. Once a path is chosen, all the other paths disappear from the canvas, while the list box is disabled. However, the paths are not deleted from the memory and can always be recalled if there is a need for them.

5.3. Function

The main function of ICNavS is to compute a set of maximally disjoint paths between a chosen origin and destination, using the modified link penalty method, as described in Section 2.3. Thus, when an origin and destination pair has been selected, a forward run of the A* algorithm is executed, outputting the fastest path, not making any reliability considerations. Following that, link penalties are applied on links with a reliability value lower than a specified threshold and a reverse A* search is run, using the computed actual costs of the visited nodes from the destination as the heuristic estimates of the remaining distance. As soon as a path is obtained, it is checked against the imposed constraints (path duration, path length), and if these are met, the path is added to the path set.

Applying link penalties on all unreliable links and all used links (links that are already in a path), a subsequent reverse A* search is executed to yield another alternative path. For this path to be accepted, apart from the path duration and path length constraints, a maximum path overlap ratio constraint is also applied. The process continues until the fourth constraint is also met, i.e. a sufficient number of paths have been computed, or until no more paths satisfying the imposed constraints are available.

6. Traffic data simulation

In order to carry out the experiments in Section 7, traffic data was simulated. This includes link speed data, from which link travel time data is extracted, and junction delay data. How each is simulated is described next.

6.1. Link speed data

A road network is usually hierarchical. This means that there are different types of road, each one having different properties, such as traffic flow speeds and number of lanes. Generally, all roads can be placed in one of the following five categories:

1. Motorway

2. Major A-road
3. A-road
4. B-road
5. Minor road

For longer distances, drivers prefer higher ranked roads, as they enable higher speeds. This is why motorways and major A-roads are usually preferable for inter-urban trips. However, each driver's origin and destination are usually only accessible by roads of a lower category.

In order to simulate link speeds and to obtain travel time data using the hierarchy of roads, an approach based on the speed limits of each road type is adopted. Namely, it is assumed that vehicles travel at a constant speed at a given time interval of the day and that the highest value that this speed can take is the speed limit, which occurs during off-peak hours. Link speed profiles are thus derived using 15-minute intervals and are shown on Figure 3 for all five road types for weekdays (a) and weekends (b).

[Figure 3 goes here]

Using the rule that time is given by space over speed, the travel time of every link in the network at any time of the day can be calculated. Thus, travel time data can be generated for an entire network.

6.2. Junction delays

The delays experienced at junctions are, together with the delays encountered on links, the main sources of loss of travel time along a route. A good route should therefore ensure that they are minimal. Junction delays are strongly dependent on the type of movement that takes place. Right turns (assuming left-hand driving) usually cause higher delays than straight-on movements and left turn movements.

Chen et al [17] assume a delay value of 0 for left turns. On the other hand, the delay of a straight-on movement or right turn is a randomly generated value between 0 and 2 minutes, with right turns having generally higher delay values than straight-on movements. In this paper, more specific values are given, bearing in mind that these values have to be relatively low for a road network, in order to be realistic.

The estimated values assigned to junction delays in this paper also depend on the types of roads meeting at the junction and on the time and day of occurrence. Left turns are still assumed to have a delay of 0, as a left turning movement does not usually come into conflict with other traffic streams.

The estimated junction delay for a right turn movement depends on the road type of the link, from which the movement starts and on the road type of the link, to which the movement is directed. Example right turn delay profiles for right turns starting from a “Major A Road” link are shown in Figure 4 for weekdays (a) and weekends (b).

[Figure 4 goes here]

The estimated junction delay for a straight-on movement depends on the road types of the links that are being crossed by the movement, and more specifically, on the road type of the link with the highest road type crossing the movement in question. Example delay profiles for straight-on movements starting from a “Major A Road” link are shown in Figure 5 for weekdays (a) and weekends (b).

[Figure 5 goes here]

It should be noted here, that the above simulated values are only used in the experiments presented in the next section, for the purpose of demonstration of the reliable dynamic route guidance approach described in Section 3 and the software tool introduced in Section 5. When it comes to using the approach and the software tool in a real route guidance application, speed and junction delay profiles

will be obtained from floating car data. This will also enable the acquisition of link speed and junction delay distributions, so that reliability values, according to the definition of Section 2.2, can be obtained.

7. Experimental results

In order to demonstrate ICNavS, a simulation experiment is carried out on a real road network. This section gives an insight to the experiment carried out and presents the results obtained.

7.1. Description of the experiment

The network chosen for the execution of the experiments is a part of the West London area of Kensington, shown in Figure 6, containing 384 nodes and 956 links and spreading over a length of approximately 2.5 km and a width of 1.5 km. It is modelled according to the methods described in Section 3 and then imported into ICNavS.

[Figure 6 goes here]

The parameters of the simulation experiments are given next. First of all, parameter α, which is used in the link penalty application function, determines the rate of reduction of the link penalties. As α takes values between 0 and 1 and as it is exposed to the power of the number of iterations, the smaller it is the faster the link penalties are reduced from the one iteration to the next. Consequently, a very small value of α will significantly reduce computation time, as the link penalties will become zero in very few iterations and the procedure will stop, as no more paths will be found. However, this will result is some acceptable paths to be missed out, because of the small number of iterations. On the other hand, a large value of α will guarantee that, if available, more paths will be found, however it will significantly increase computation time, as the rate of reduction of the link penalties will be small and more iterations will be required. For the present experiments, the value of α is set to 0.7.

A similar concept lies behind the setting of parameter W_0 in the link penalty application function. It has to be ensured that W_0 is a large value, in any case significantly larger than the link travel time. A sensible choice in this case would be a multiple of the total travel time of the fastest route, T_0, calculated in the first step of Chen's link penalty method. W_0 is thus set to be equal to γT_0, where γ takes values between 1.5 and 3. As can be observed from the link penalty application function, γ determines the magnitude of the first (and largest) link penalty. Consequently, small values of γ mean that the link penalties applied will also be small and that inclusion of some excluded links in the computed paths will be prevented, but not totally avoided. The larger the value of γ, the greater the probability of inclusion of the excluded links is reduced. For the present experiment, γ is set to 1.9.

Having determined the parameters of the link penalty application function, the parameters of the constraints imposed are set next. Considering the maximum duration and maximum length constraints, the parameters β and ζ are introduced, such that the total duration of the computed paths may not be larger than βT_0 and the total length of the computed paths may not be greater than $\zeta \Lambda_0$, where Λ_0 is the total length of the fastest path. The values of β and ζ can vary depending on the size of the network. The present test network is fairly small, such that travel times do not exceed 7 minutes and distances are at most 3.5 km. Therefore, for the present experiment, $\beta = 2$ and $\zeta = 2$, which means that the computed paths may be twice as long as the fastest path, both in terms of travel time and distance, to be acceptable. Of course, in a larger network where travel times usually exceed 30 minutes and distances are often as great as 30 km, values of $\beta = 2$ and $\zeta = 2$ would result in routes of duration of 1 hour and of distance of 60 km to be acceptable, which is clearly not realistic. Values of 1.1 or 1.2 would be assigned to β and ζ in that case.

For the third constraint, which ensures that the computed paths do not overlap too much, the dimensionless overlapping ratio ε is introduced, defined as $\varepsilon = \varepsilon_{ab} / \sqrt{(\varepsilon_a \times \varepsilon_b)}$, where for two paths a and b, ε_{ab} is the total shared length and ε_a and ε_b are the unshared lengths of the two paths respectively. Setting the value of ε determines how much overlapping is allowed, with values close to zero

indicating that very little or no overlapping is allowed and larger values permitting more overlapping. In this experiment, a value of $\varepsilon = 2$ is set. Finally, regarding the maximum number of computed paths, this number is set equal to 5.

The experiment is carried out as follows; St Mary Abbots Place, W8, at the western end of the network, is chosen as the origin of the trip, while the destination is set to be Pelham Crescent, SW3, at the eastern end of the network. The trip takes place during the afternoon peak on a weekday, knowing that a number of links on the eastbound branch of the A4 highway (Cromwell Road), are unreliable (have reliability values of 0.5-0.7), while, the links forming the A3220 road (Warwick Road/Pembroke Gardens) are also unreliable, with reliability values of 0.6-0.8. The results are presented next.

7.2. Results

The fastest path and five alternative paths to it are computed and are shown on Figure 7. The fastest path (a) is computed first, not taking into account the links and movements with low reliability values. It has a travel time of 5.4 minutes, a length of 2.1 km and goes through the unreliable A3220 and then onto the equally unreliable A4, thus resulting in a low path reliability value (0.05).

In the calculation of the first alternative path however (b), reliability is taken into account, and this is why both the A3220 and the A4 are avoided. Instead, a route along the A315 (Kensington High Street) is chosen. This route has a slightly higher travel time than the fastest route (7.7 minutes), is slightly longer (2.5 km), but has a much higher reliability value (0.43).

For the calculation of the second alternative path (c), not only unreliable links and movements are penalised, but also links already included in the first alternative path. Hence, the A315, used by the first alternative path, is avoided. The unreliable A4 is also avoided, and the path goes along the A3218 (Old Brompton Road), having a travel time of 7.3 minutes, a length of 2.4 km and a reliability value of 0.4.

The other three alternative paths that are calculated, (d), (e) and (f), avoid the links already included in the previously computed alternative paths as much as possible and therefore, despite sharing some parts of the main roads with each other, mainly include minor roads. Their travel times are 7.5, 8 and 6.3 minutes, their lengths are 2.3, 2.5 and 2.3 km and their reliabilities are 0.45, 0.48 and 0.21 respectively.

[Figure 7 goes here]

The results obtained seem to be plausible, as both the paths computed and the travel time and reliability values calculated are reasonable and correspond to the authors' choices, who have experience with the network. This suggests that the procedure, although at a very early stage, yields promising solutions.

8. Conclusion

In this paper, a novel reliable dynamic route guidance approach was presented, integrating the A* path finding algorithm, the concept of reliability and Chen's link penalty method. The approach was implemented and demonstrated through a simulation experiment on a part of West London's road network, using a new software tool, ICNavS. In order to use the software tool to compute reliable paths on real road networks, a procedure for modelling features of real road networks, as well as missing data was described. The preliminary results obtained from the experiment indicated that the approach is workable and has the potential of yielding promising results.

Based on the conclusions obtained from this study, further research will be conducted in this direction. Namely, it is intended to build a prototype reliable dynamic route guidance system using ICNavS as the platform. Issues such as reliability variability, inter-link dependences and congestion prediction will be considered. Further experiments using floating car data will be carried out on several different

road networks. The new system will then be field trialled in a real network and the results will be compared with those yielded by a conventional route guidance system.

Acknowledgement

The authors would like to thank BMW AG for funding this work.

References

[1] Ahuja RK, Magnanti TL and Orlin JB, "Network flows: Theory, Algorithms and Applications", Prentice Hall, 1993.

[2] Hart, PE, Nilsson, NJ and Raphael, B, "A formal basis for the heuristic determination of minimum cost paths", IEEE Transactions on Systems Science and Cybernetics 4, 1968, pp. 100-107.

[3] Dijkstra, EW, "A note on two problems in connexion with graphs", Numerische Mathematik 1, 1959, pp. 269-271.

[4] Jackson, WB and Jucker, JV, "An empirical study of travel time variability and travel choice behaviour", Transportation Science 16, 1981, pp. 460-475.

[5] Bates, J, Black, I, Fearon, J, Gilliam, C and Porter, S, "Supply models for use in modelling the variability of journey times on the highway network", AET - Proceedings of the European Transport Conference 2002 (Cambridge, UK, 2002).

[6] Liu, HX, Recker, W and Chen, A, "Uncovering the contribution of travel time reliability to dynamic route choice using real-time loop data", Transportation Research A 38, 2004, pp. 435-453.

[7] Lam, TC and Small, KA, "The value of time and reliability: measurement from a value pricing experiment", Transportation Research E 37, 2001, pp. 231-251.

[8] Abdel-Aty, A, Kitamura, R and Jovanis, PP, "Investigating the effect of travel time variability on route choice using repeated measurement stated preference data", Transportation Research Record 1493, 1995, pp. 39-45.

[9] Noland, RB and Small, KA, "Travel-time uncertainty, departure time choice, and the cost of morning commutes", Transportation Research Record 1493, 1995, pp. 150-158.

[10] Bates, J, Polak, JW, Jones, P and Cook, A, "The valuation of reliability for personal travel", Transportation Research E 37, 2001, pp. 191-229.

[11] Noland, RB and Polak, JW, "Travel time variability: a review of theoretical and empirical issues", Transport Reviews 22, 2002, pp. 39-54.

[12] Polus, A, "A study of travel time and reliability on arterial routes", Transportation 8, 1979, pp. 141-151.

[13] Dandy, GC and McBean, EA, "Variability of individual travel time components", ASCE Journal of Transportation Engineering 110, 1984, pp. 340-357.

[14] Lomax, T., Schrank, D., Turner, S., and Margiotta, R., "Selecting travel reliability measures", Texas Transportation Institute, 474360-1, 2003.

[15] van Lint, JWC and van Zuylen, HJ, "Monitoring and predicting freeway travel time reliability - Using width and skew of day-to-day travel time distribution", Transportation Research Record 1917, 2005, pp. 54-62.

[16] Bell, MGH and Iida, Y, "Network reliability" in Transportation network analysis, Wiley and Sons, Chichester, 1997.

[17] Chen, Y, Kaparias, I, Bell, MGH and Bogenberger, K, "Reliable autonomous route guidance by a constrained A* search considering intersection delays", The Reliability of Traveling and the Robustness of Transport Systems (Delft, The Netherlands, 2005).

[18] Chen, Y, Bell, MGH, Wang, D and Bogenberger, K, "Risk-averse time-dependent route guidance by constrained dynamic A* search in decentralized system architecture", Transportation Research Record 1944, 2006, pp. 51-57.

[19] Chen, Y, Bell, MGH and Bogenberger, K, "Reliable pre-trip multi-path planning and dynamic adaptation for a centralized road navigation system", ITSC 2005 - 8th International IEEE Conference on Intelligent Transportation Systems (Vienna, Austria, 2005).

[20] Kaparias, I, Bell, MGH, Bogenberger, K and Chen, Y, "An approach to time-dependence and reliability in dynamic route guidance", TRB 2007 Annual Meeting CD-ROM (Washington DC, USA, 2007).

[21] Lee, C-K, "A multiple-path routing strategy for vehicle route guidance systems", Transportation Research C 2, 1994, pp. 185-195.

[22] Polenta T. Incorporating uncertainty in real-time route guidance systems. PhD Thesis, Nottingham Trent University, 2005.

[23] Miller-Hooks, E and Mahmassani, HS, "Least possible time paths in stochastic time-varying networks", Computers and Operations Research 25, 1998, pp. 1107-1125.

[24] Miller-Hooks, E and Mahmassani, HS, "Least expected time paths in stochastic, time-varying transportation networks", Transportation Science 34, 2000, pp. 198-215.

[25] Miller-Hooks, E, "Adaptive least-expected time paths in stochastic, time-varying transportation and data networks", Networks 37, 2001, pp. 35-52.

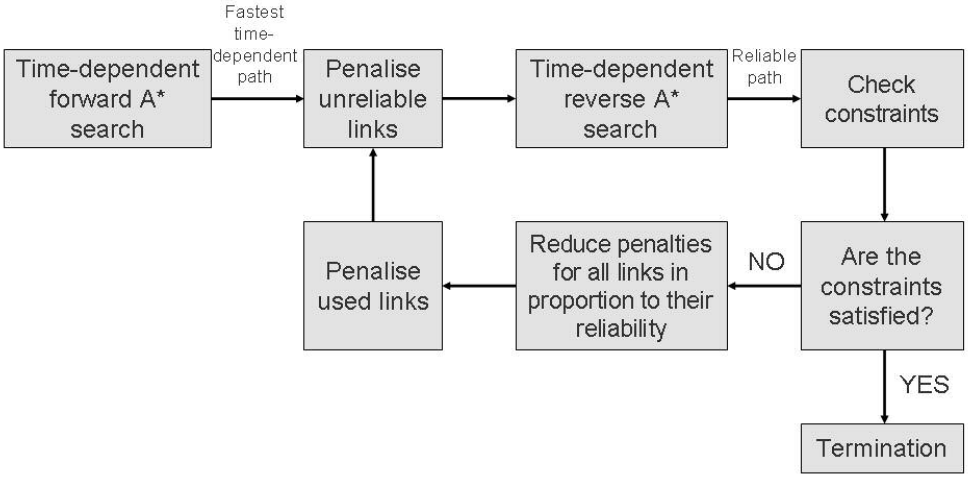
[26] Park, K, Bell, MGH, Kaparias, I and Bogenberger, K, "Learning user preferences of route choice behaviour for adaptive route guidance", 13th World Congress on Intelligent Transportation Systems (London, 2006).

[27] Park, K, Bell, MGH, Kaparias, I and Bogenberger, K, "Adaptive route choice model for intelligent route guidance", TRB 2007 Annual Meeting CD-ROM (Washington DC, USA, 2007).

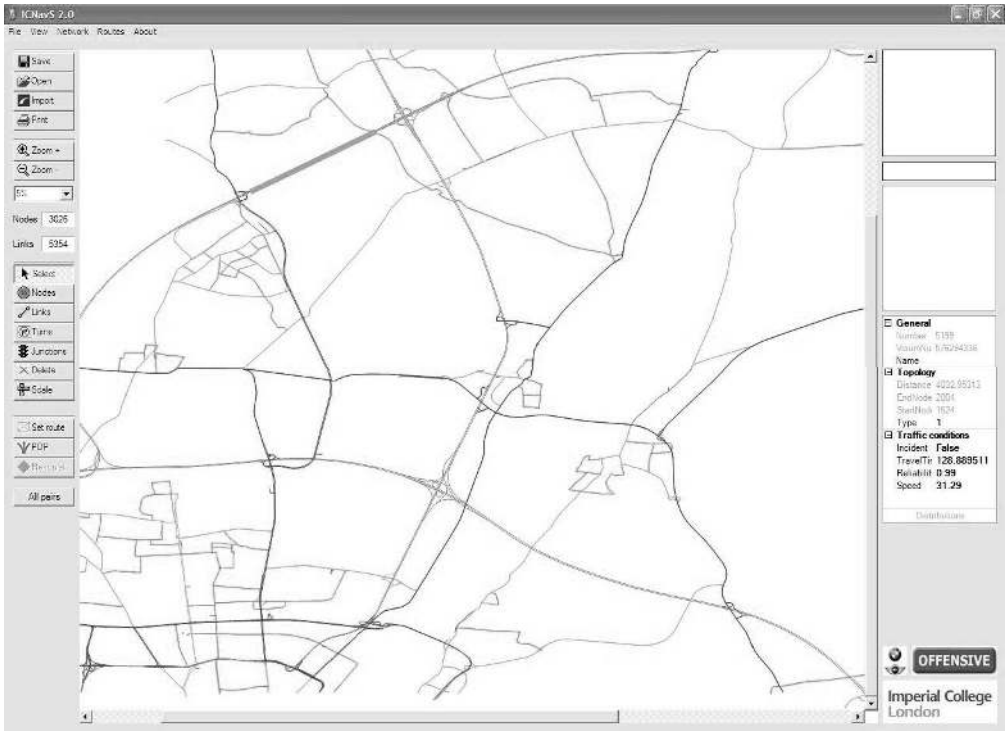
[28] Wattleworth, JA and Shuldiner, PW, "Analytical methods in transportation; left turn penalties in traffic assignment models", Journal of Engineering Mechanics of the American Society of Civil Engineers 89, 1963, pp. 97-126.

[29] Kirby, RF and Potts, RB, "The minimum route problem for networks with turn penalties and prohibitions", Transportation Research B 3, 1969, pp. 397-408.

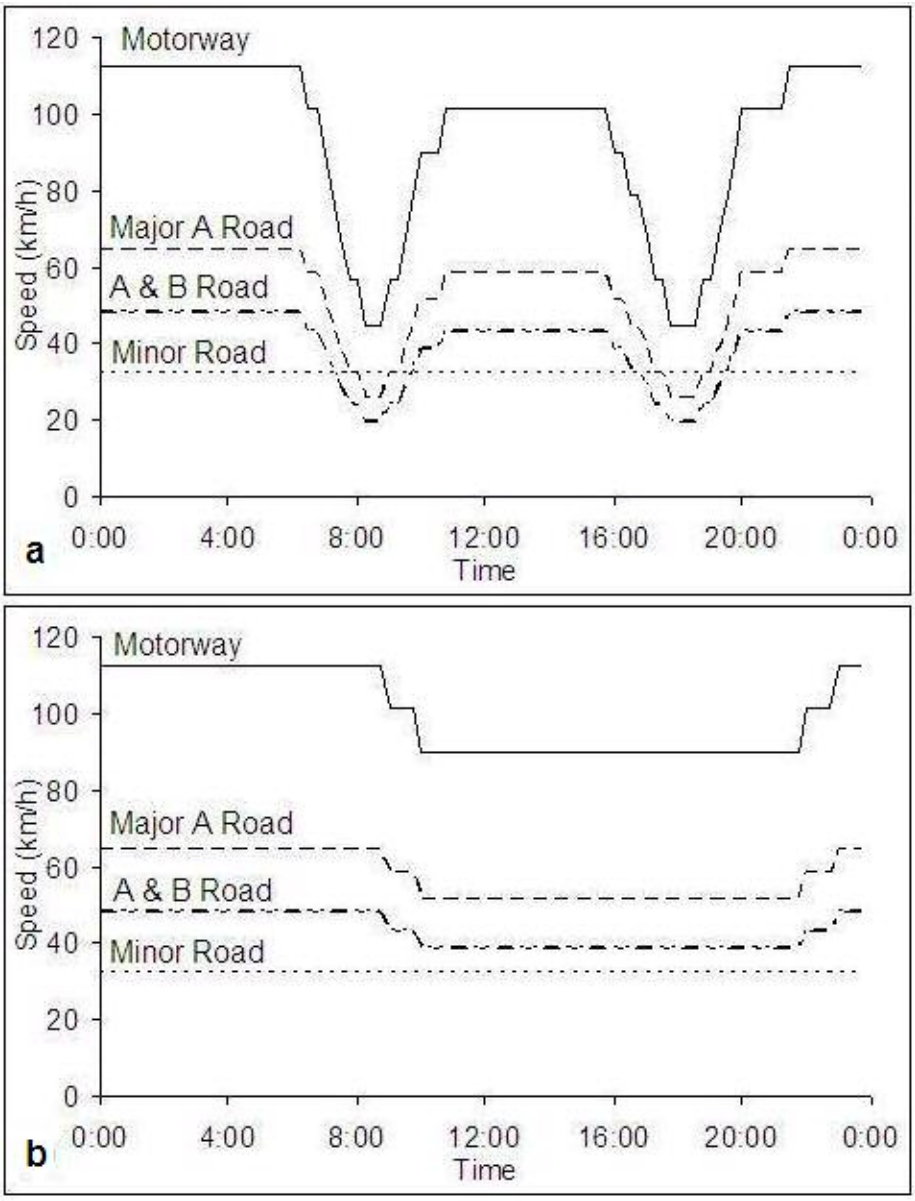
[30] Ziliaskopoulos, AK and Mahmassani, HS, "A note on least time path computation considering delays and prohibitions for intersection movements", Transportation Research B 30, 1996, pp. 359-367.



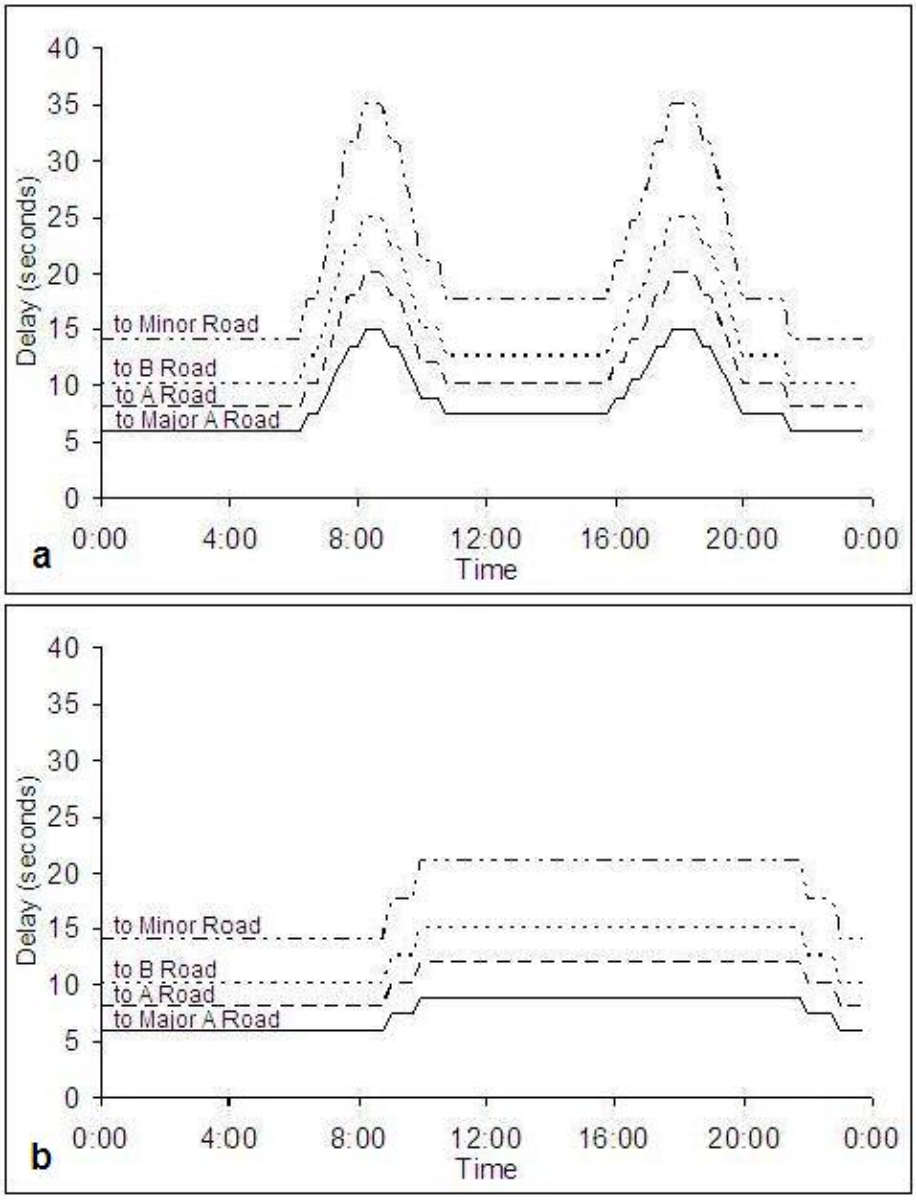
The modified link penalty method



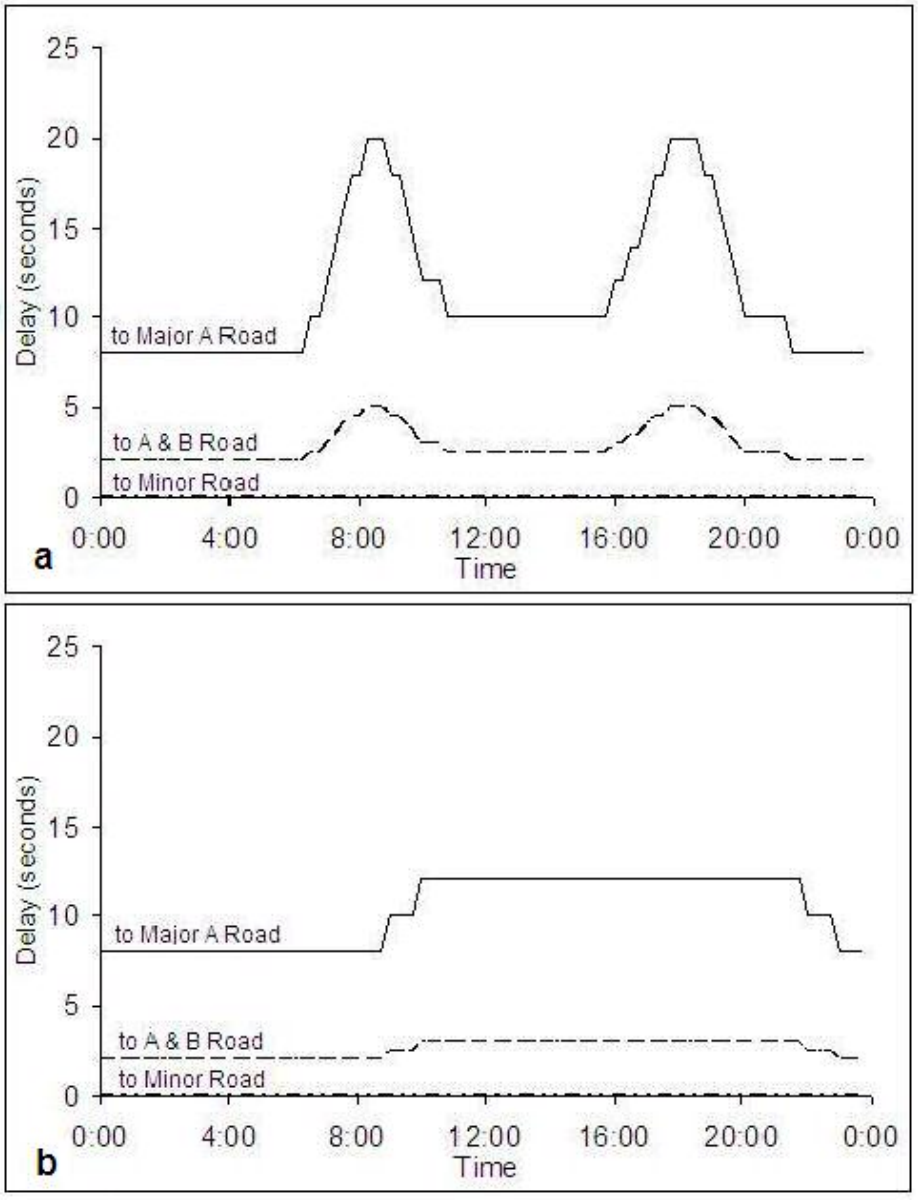
ICNavS user interface



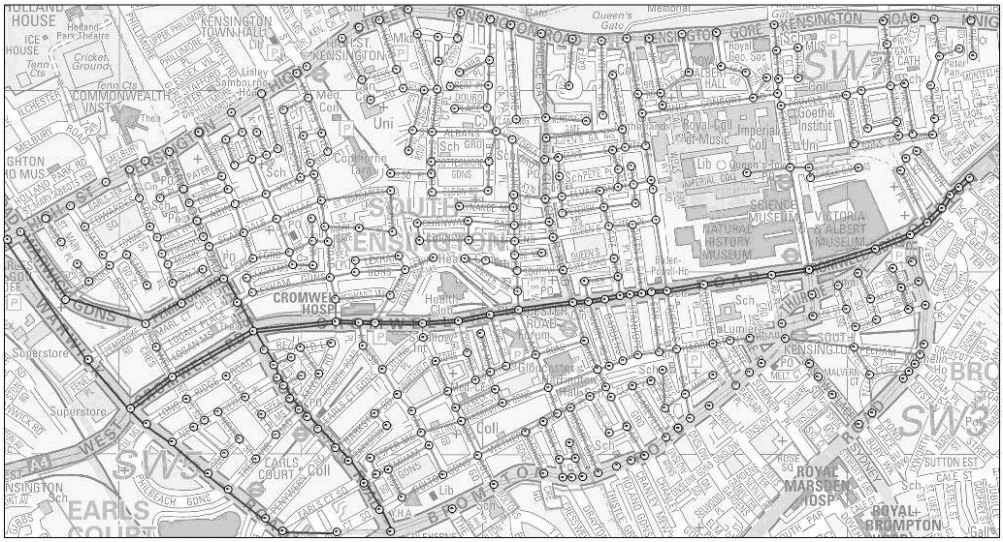
Link speed profiles for (a) weekdays and (b) weekends



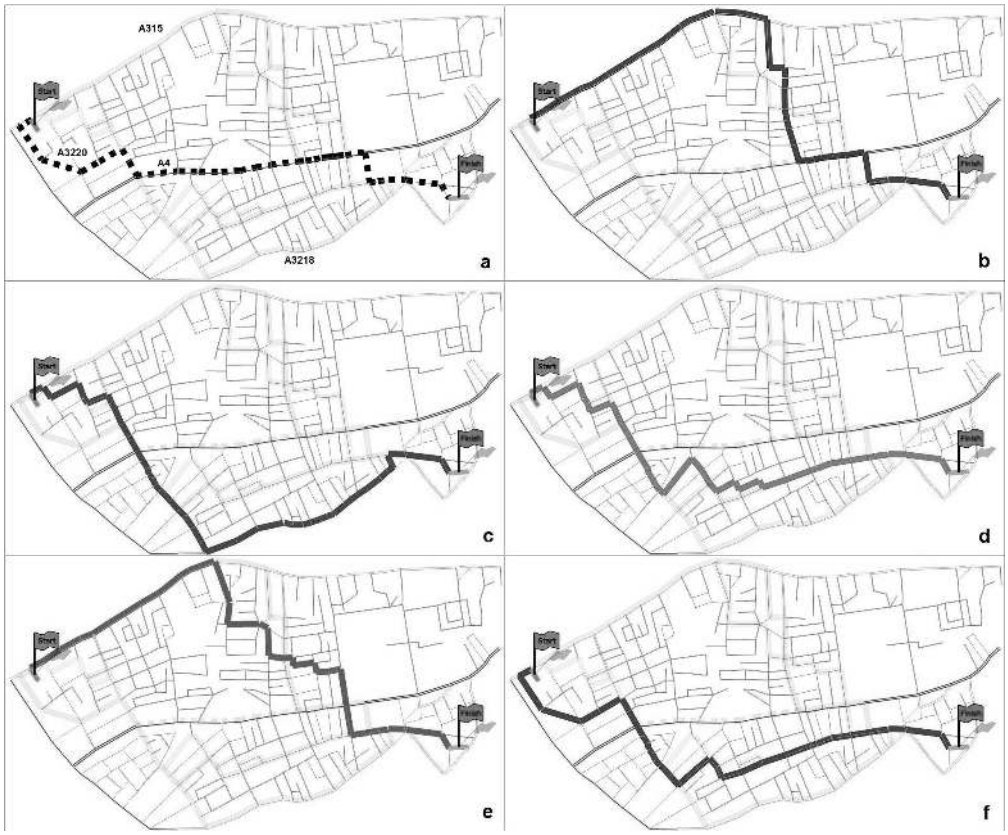
Delay profiles for right turns starting from "Major A Road" links for (a) weekdays and (b) weekends



Delay profiles for straight-on movements starting from "Major A Road" links for (a) weekdays and (b) weekends



The West London test network (Image source: www.multimap.com)



The fastest path and the five reliable alternatives output from ICNavS