Reliable Autonomous Route Guidance by a Constrained A* Search Considering Intersection Delays

TRAIL Research School, Delft, May 2005

Authors
Prof. Yanyan Chen
Transportation Research Center, Beijing University of Technology, China
Ioannis Kaparias and Prof. Michael G.H. Bell
Centre of Transport Studies, Imperial College London, U.K.
Dr. Klaus Bogenberger
Department of Science and Traffic Policy/Traffic Technology, BMW Group, Germany

Contents

Abstract

An efficient algorithm for finding a time-reliable path subject to a maximum duration constraint in autonomous in-vehicle navigation systems is proposed. When no dynamic traffic information is available, a reliable path can be found by a constrained A* search based on default link and turning movement data profiles stored on an on-board disk. Intersection turning movement penalty and delay are considered in the algorithm design. The robustness of navigation is improved by taking travel time uncertainty into account for both links and turning movements. By making use of data from previous path searches, a learning version of the constrained A* algorithm is proposed which reduces computation time. An in-depth experimental performance analysis of the proposed methods on grid graphs demonstrates their efficiency.

1 Introduction

Advanced traveler information systems are an integral component of Intelligent Transportation Systems (ITS). Among these, in-vehicle information systems and especially route guidance systems (RGS), are spreading gradually through the vehicle fleet. Route guidance may be provided to a driver by means of two strategies; *autonomous route guidance* and *supported route guidance*. In autonomous route guidance systems, routes are suggested to drivers on the basis of default estimates of link travel times, possibly complemented by broadcast dynamic traffic information about incidents. Network data is accessed by the guided vehicle from a static database available on an on-board disk. In contrast to that, supported route guidance has a more centralized architecture with a two-way exchange of data between the guided vehicle and a Traffic Information Centre (TIC); as a result, the range and accuracy of the information used in providing guidance is increased.

Compared to the supported route guidance scheme, autonomous route guidance has the following advantages; firstly, the suggested path is based on the static traffic database which is preloaded on the vehicle. Therefore, the response time is not affected by bandwidth limitations of the radio channel through which dynamic data is transferred and the calculation is not subject to possible interruptions or inaccuracies of the dynamic data being transmitted. Furthermore, users usually need to pay a lower (or no) subscription fee for static information than for dynamic. However, the main drawback of autonomous route guidance can be easily identified; the suggested path is based on average traffic conditions, which may deviate considerably from the actual traffic conditions, especially when non-recurrent incidents occur. In seriously congested urban road networks, such a situation is often encountered.

Therefore, there is a need to develop an autonomous route guidance system, which will be more reliable under dynamic traffic conditions in a congested urban area without being strongly dependent on transmitted data. The outcome of such a system will therefore be a set of route recommendations that prove *ex post* to have been good. A good route in this context is one that, although possibly not the best (fastest), is both acceptable to the driver and reliable. We call such a system "reliable autonomous route guidance". As the path calculation has to be performed by the in-vehicle unit which usually has limited computational power and memory, the network has to be represented in the form of an efficient data structure and the search algorithm has to be fast enough to meet real time requirements.

In our autonomous navigation strategy, the network features, the intersection movements and possible movement prohibitions are represented and stored by an extended forward star structure. The autonomously guided vehicle will be provided with the capability of accessing default travel time information, which includes not only static travel time predictions but also travel time uncertainty information. When no dynamic traffic information is available, the default travel time predictions will be used for navigation. The reliability of the navigation is improved by taking travel time uncertainty into account. A constrained A* algorithm is proposed to search for a reliable path whose duration is not too great. The efficiency of the algorithm is increased by taking advantage of previously calculated information, thereby decreasing the time required for subsequent A* searches.

This paper is organized as follows: Section 2 contains the review of past work; Section 3 presents the construction method for the default link profiles required for reliable autonomous navigation; Section 4 prescribes a modified A* algorithm which accounts for turning delay at intersections; Section 5 describes the constrained A* algorithm for reliable navigation; Section 6 presents the real time implementation of the constrained A* algorithm; and Section 7 presents some numerical experiments. Finally Section 8 presents the conclusions thus far and suggestions for future work.

2 Literature review

2.1 A* Algorithm

A sub-procedure that is frequently used in navigation systems is finding the shortest path in a graph. Although various algorithms exist for finding the shortest path, their performance tends to deteriorate as the network size increases. Since we work on very large networks and need to call this subroutine quite often in an iterative procedure when searching for a most reliable path subject to a maximum duration constraint, it is important to have shortest path algorithms that are efficient to meet real-time operational requirements.

A leading algorithm for finding the shortest path from one node to another is the A* algorithm (Hart and Nilsson, et. al. 1968), which belongs to the label correction category. Unlike the breadth-first search algorithm, the A* algorithm uses an estimate of the minimum distance from each node to the destination node in order to determine how likely it is that any node lies on the best route. The cost associated with a node *n* is $f(n)=g(n)+h(n)$, where $g(n)$ is the minimum cost of the path from the start node to node *n*, and $h(n)$ is an estimate of the minimum cost to the destination node from node *n*. Thus $f(n)$ estimates the minimum total cost of any solution path passing through node *n*. Two lists, the open list and the closed list, are maintained during path searching. The open list contains candidate nodes that may or may not be part of the selected path. In other words, the nodes on the open list are eligible to be expanded, but have not been expanded yet. The closed list contains nodes that have already been expanded. The shortest path is generated by repeatedly going through the open list and choosing the node with the lowest $f(n)$ to be expanded next. Provided $h(n)$ is an under-estimate of the minimum cost from *n* to the destination node (like the cost based on the Euclidean distance), the A* algorithm is guaranteed to find the true optimal solution. When $h(n)$ is replaced by 0, the algorithm reduces to Dijkstra's algorithm (Dijkstra, 1959). The efficiency of searching increases as $h(n)$ approaches the optimal value, so that if $h(n)$ is the minimum cost only nodes on the minimum cost path will be expanded.

2.2 Shortest path search considering intersection delays

Most of the available shortest path algorithms assume that there are no costs or prohibitions associated with intersection movements. In urban areas however, the magnitude of these movement delays may even be comparable to link travel times and ignoring them may lead to the computation of sub-optimal or illogical paths. The main obstacle in computing a path considering intersection delays is the difficulty in representing the network in an economical, compact and manageable way. An extensively used approach in transportation applications is to analyze each intersection as a sub-network by introducing dummy nodes and links to represent all the possible movements and then to use conventional shortest path algorithms (Yagar, 1971). This expanded network nolonger has turning delays and prohibitions, and can be solved with any standard shortest path algorithm. Nevertheless, such a method considerably increases the computation time and computer memory requirements. Furthermore, extensive coding work is needed to alter the structure of the network rendering it inflexible to modifications. Another scheme (Kirby and Potts, 1969) is a link-based variant of conventional shortest path algorithms, in which every possible turn is represented as a chain. While it does not require transformation of the original network, it is still uses computer memory inefficiently. Ziliaskopoulos and Mahmassani (1996) make further improvements in this field by taking an extended forward star structure to represent the turning movements and by modifying the label correcting shortest path algorithm to calculate the fastest paths.

2.3 Travel time uncertainty and multiple objective routing

In congested urban road networks, link travel times have a high degree of uncertainty (Fu et al., 1998). The representation of travel time uncertainty can be divided into two fields; travel time variance and travel time reliability. Some literature estimates travel time variance by using simulation (Noland et al., 1998), other literature proposes analytic methods for estimating the probability distribution of total network travel time in the light of normal day-to-day variations in the travel demand matrix (Clark and Watling, 2004; Pattanamekar et al., 2003). As travel time reliability plays an important role in route choice behavior (Abdel-Aty et al., 1995; Bates et al., 2001; Lam and Small, 2001; Liu et. al, 2004), some researchers have begun to consider travel time uncertainty in route planning. For example, Miller-Hooks and Mahmassani (2000) find the least expected time paths in graphs with random edge weights, while Bell and Cassir (2002) use game theory to derive reliable routing strategies. From a mathematical point of view, considering travel time uncertainty in path searching can also be dealt with by formulating a multi-objective problem.

A multi-objective routing can be found by considering one dominant criterion as the objective function while taking a threshold for the other criterion as a constraint. This generates a constrained shortest path problem. However, such a modification causes the route guidance problem to become NP-hard, even in the simpler case of constant link travel times. Finding the K shortest paths instead of one path (Climaco and Martin, 1998; Eppstein, 1998) is another approach to this problem. One reason to do so is that certain constraints may be difficult to decide or hard to optimize. A common strategy is to compute several paths and then choose among them by considering the other criteria.

3 Construction of default traffic information profiles for autonomous navigation

What we are interested in this paper is to find the most reliable path from an origin node s to a destination node d in a graph subject to an acceptable duration constraint. We let a network $N(V, E)$ denote a city road network, where the node set V corresponds to the intersections including the origin s and the destination d, and the link set E to the road segments connecting the nodes in the city. As intersection movement delays can be significant in congested urban road networks, they must be taken into account in a reliable autonomous navigation system. The default link travel times, as well as the probabilities of links and intersections performing normally, are assumed to reside on a disk in the guided vehicle. These will be called the *default link and intersection profiles*.

3.1 Default link and intersection travel times

As is well known, road travel times vary between different days of the week and different times of the day. However, a strong similarity can be detected for the same time segment between different days (Sen et al. 1998, 1999), if no abnormal incidents or events occur. Therefore, travel time fluctuations can be divided into two categories; normal fluctuations and abnormal fluctuations. The former reflect travel time variance under recurrent (normal) conditions, whereas the latter represent the corresponding situation under non-recurrent (abnormal) conditions. The degree of normal fluctuations is relatively small compared to that of abnormal fluctuations.

For the links for which historic data over a certain period (over a month or longer) is available, we will carry out a statistical analysis and compute the mean normal speed and travel time for each link during different time segments in a day. Speed here is regarded as *normal* if it is not lower than the defined threshold for abnormal delay. The mean normal travel times for each link during different time segments constitute the default link travel times for that link.

For links for which no historic data is available over the relevant period, we can perform an Origin-Destination (O-D) assignment for different times of the day, based on historical O-D data, and assume that the resulting link travel times are close to the mean normal travel times of the corresponding links. Of course, the cost or delay functions used in the assignment should be correct for each particular type of link. Table 1 indicates the contents of the link travel time data profile.

Table 1: Mean normal speed (travel time) of different road links at different times of the day

Time No. of Link	7:00—9:00 (morning peak)	9:00—16:00 (slack time)	16:00—19:00 (afternoon peak)	19:00—22:00 (slack time)	22:00—5:00 (low time)	5:00—7:00 (slack time)

For intersections, in order to save storage space and computation time, we can just consider the ones for which the turning delay is comparable to that of links and which can therefore not be omitted. The method for mean normal delay estimation for intersections is similar to that for links.

3.2 Default link and intersection reliability

To improve the veracity of the navigation based on the static data profile in a dynamic traffic environment, travel time variation or uncertainty needs to be considered. As a result of the large expected running times involved in stochastic route planning, appropriate criteria and models are needed to handle uncertainties, along with efficient solution techniques.

Usually, drivers are not so sensitive to normal travel time fluctuations and are pleased to be guided along the least mean travel time path when travel time fluctuations are within acceptable bounds. This is why we divide the variation of travel time into two parts; normal fluctuation and abnormal fluctuation. Accordingly we assume each component (link or turning movement) has only two states; normal and abnormal. The threshold separating the normal from the abnormal state is a pre-defined value of link speed, above which the link is considered to be performing normally. For intersections on the other hand, the threshold can be a pre-defined value of turning delay. Links or turning movements performing abnormally relatively frequently are unreliable. Under this binary assumption, the reliability of a link or turning movement could be defined as the probability of the link or turning movement not incurring an abnormal delay (or travel time going beyond a threshold) in a given time interval, which is a value between 0 and 1. For component *i*, the reliability can be obtained from historical data as follows:

$$r_i^j = \frac{N_{r,i}^j}{N_i^j} \tag{1}$$

where r_i^j is the reliability of link *i* in a given time interval *j*, e.g. 7:00—9:00; $N_{r,i}^j$ is the numbers of samples whose value is larger than a critical travel time threshold in interval *j*; N_i^j is the total number of samples of historical travel time data for link *i* in interval *j*. Note that for interval *j* in one day, several travel time measurements are taken and the sample value is the mean travel time of the counted vehicles.

The reliability of an intersection can be estimated by a similar method to the one mentioned above. Different turns in an intersection may have different reliabilities. Usually, under the left-hand-drive rule, left turns have a higher reliability, whereas under the right-hand-drive rule, right turns are more reliable.

To save storage space and memory, only the reliabilities of high-risk links and turning movements of some intersections are estimated and stored. High-risk in this context means that there is a relatively high probability of encountering abnormal congestion, leading to large delays which are difficult to estimate.

Usually, the probability of abnormal delay to occur on a path increases as the flow increases. Hence, when historical data over a long period is not available, a reliability index r_i based on the degree of saturation v/c (Ensor, 2004) can be used. This describes the relative reliability of link *i* not incurring serious congestion that will induce abnormal travel time fluctuations in a given time interval *j* of a day and can be estimated approximately by the following expression:

$$r_i = \begin{cases} 1 & v_i / c_i < 0.7 \\ 1 - v_i / c_i & 0.7 \le v_i / c_i < 1.1 \\ 0.1 & v_i / c_i \ge 1.1 \end{cases} \tag{2}$$

where c_i is the capacity of component *i* and v_i is the traffic flow on component *i* which can be obtained through static traffic assignment.

3.3 Representation of road network and storage of default traffic profiles

The computer memory needs to be able to store the network structure and traffic information including link travel costs, the intersection turning penalties and their corresponding reliabilities. The manner in which a network is represented (in a computer) directly affects the performance of algorithms applied to it. Ziliaskopoulos and Mahmassani (1996) used an effective extended forward star structure (EFSS) to consider intersection delays and prohibited turns in the shortest path algorithm by a slight modification. In our paper, we will use the same structure to store the default traffic information of links and intersections.

In the forward star structure (FSS) a pointer is kept for each node of the network, indicating the position of links beginning at this node in the link list. Fig.1b illustrates this pointer representation for the network sketched in Fig. 1a.

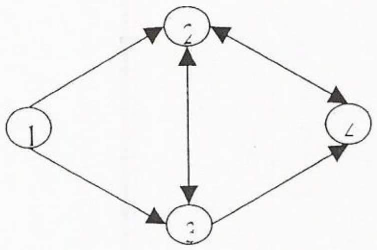

Figure 1a

Node	Forward Pointer	Pointed Nodes	t_{ij}	r_{ij}	t_{ijk}		r_{ijk}	
1	1	2	40	0.93	3	7	0.95	0.85
2	3	3	42	0.85	5	6	0.90	0.60
3	5	4	54	0.55	23		0.50	
4	7	3	73	0.90	17	12	0.90	0.99
		2	42	0.75	4	16	0.70	0.60
		4	64	1.0	11		1.0	
		2	82	0.95	19	9	0.90	0.99

Figure 1b

In the Fig.1b, t_{ij} is the mean normal travel time required to go through link (i,j) , i, $j \in V$ and r_{ij} is the corresponding travel time reliability. t_{ijk} is the delay associated with a turning movement at intersection j for a vehicle coming from the predecessor node i and leaving for node k. The movement is described as i->j->k in the paper. r_{ijk} is the corresponding turning travel time reliability. Some of these delays may be infinite, meaning that the corresponding movement is prohibited.

4 Modified A* algorithm accounting for turning delays

Since we work on very large networks and need to call this subroutine quite often in searching for a reliable path, it is important to have shortest path algorithms that are practically efficient to meet real-time operational requirements.

In this paper, we use the extended forward star structure to represent the network and make some modification to the A* shortest path algorithm, taking into account the movement delay at the nodes. For any node i, $(|\Gamma(i)|+2)$ labels are maintained, where $\Gamma(i)$ is the set of the successor nodes to node i. $|\Gamma(i)|$ is the out degree of node i, i.e. the number of successor nodes of node i. At any stage of the computation, the $|\Gamma(i)|$ labels are upper bounds to the least travel times from the origin node to the beginning of the links emanating from node i, while the first additional label holds the value $T_{i,0}$ which is the upper bound to the least travel time from the origin node s to node i (no further movement). The second additional label holds the estimate value $f(i)$ for the path going through node i from the origin node s to the destination d, which is the sum of $h(i)$ and $T_{i,0}$. $h(i)$ is the estimate of the travel time from node i to the destination d. Fig. 2 shows the relationship of adjacent nodes, the travel times of links, the turning delays of nodes and the labels of nodes.

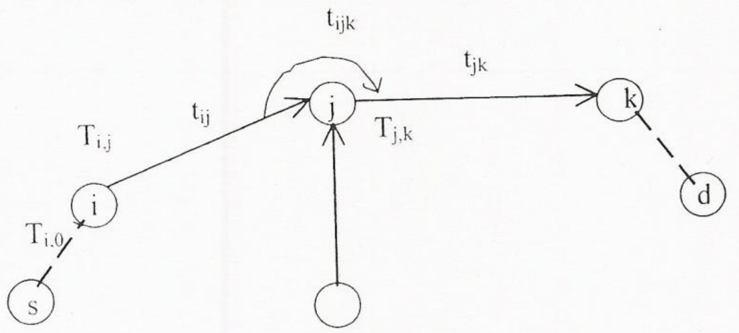

Figure 2

As in the case of the traditional label correcting algorithm, in the algorithm accounting for turning delays, a node i can be scanned (expanded) by updating the labels of the nodes $j \in \Gamma(i)$ simultaneously in every iteration by using the following equation:

$$T_{j,k} = \min_{\forall i \in \Gamma^{-1}(j)} \{t_{ijk} + t_{ij} + T_{i,j}, T_{j,k}\}, \forall j \in V \quad and \quad k \in \Gamma(j) \tag{3}$$

$$T_{j,0} = \min_{\forall i \in \Gamma^{-1}(j)} \{t_{ij} + T_{i,j}, T_{j,0}\}, \forall j \in V \tag{4}$$

where $\Gamma^{-1}(j)$ is the set of predecessor nodes to node j. $T_{j,k}$ is an upper bound to the least travel time from the origin node to the beginning of the link (j,k) emanating from node j. $T_{j,0}$ is an upper bound to the least travel time from the origin node to the node j.

As in the "best first" label correcting algorithm, the node with the lowest function cost is expanded first and in the A* algorithm the function cost associated with a node is denoted by *f*, therefore the node with the lowest value of *f* is expanded first.

The algorithm can be described as follows:

Algorithm

Step 0: Create a closed list *S*=null. Create an open list and initialize it by inserting the origin node *s*. Initialize the labels as follows:

$T_{i,0} = \infty\ T_{i,j} = \infty, \forall i \in V \quad and \quad j \in \Gamma(i)$

$T_{s,0} = 0,\ T_{s,p} = 0, \forall p \in \Gamma(s)$

Step 1: Repeat the following:

a) Look for the lowest *f* cost node *i* on the open list. Name it as the current node.

b) Switch it to the closed list.

c) For each of the nodes $j \in \Gamma(i)$

- If it is unreachable (banned turn) or if it is on the closed list, ignore it. Otherwise do the following:
    - If it is not on the open list, add it to the open list. Calculate and record $T_{j,0}$, $T_{j,k}$ and $f(j)=T_{j,0}+h(j)$; $k \in \Gamma(j)$. Record node *i* and movement *i->j->k* as predecessor node and movement respectively.
    - If it is in the open list already, examine and improve the labels of each node $j \in \Gamma(i)$. Specifically, do the following:

      1) For every possible movement *i->j->k* at node *j*, $k \in \Gamma(j)$, check if $T_{j,k}$ is greater than $t_{ijk} + t_{ij} + T_{i,j}$. If so, replace $T_{j,k}$ by the smaller value and record node *i* and movement *i->j->k* as the predecessor node and movement respectively; otherwise do nothing and continue with the next movement.

      2) For the no movement case at node j, check if $T_{j,0}$ is greater than $t_{ij} + T_{i,j}$. If it is, replace $T_{j,0}$ by the smaller value and record node *i* and "no movement" as predecessor node and movement statements respectively; recalculate $f(j)=T_{j,0}+h(j)$ of the node and resort the open list by *f* value to account for the change. Otherwise, do nothing.

d) Repeat Step 1 until:

- Destination node is added to the closed list, in which case the path has been found, or
- The destination node has not been found and the open list is empty, in which case there is no path.

Step 2. Save the path. First trace the predecessor node *m* recorded in the label $T_{d,0}$ for the destination node *d* and then keep tracing back from node *m* by using the pointers to the predecessor nodes recorded in the label $T_{i,j}$ until the origin node s is reached. This is the shortest path.

The proof of the algorithm follows the same principles as the proof of the standard label correcting shortest path algorithms (Moore, 1957; Tarjan, 1983).

5 Constrained reliable A*Algorithm

A good route in our paper is one that, although possibly not the best (fastest), is both acceptable to the driver and reliable. The suggested route is acceptable if its duration is not significantly greater than that of the optimal path under normal conditions and it is reliable if the probability of encountering abnormal delay during the trip is low. Determining the most reliable acceptable path is a bi-objective routing problem.

As we discussed above, bi-objective routing is very computationally demanding. Therefore we propose a heuristic in the form of a constrained reliable A* algorithm to find a practical solution. The idea of our algorithm arises from the following fact: in a shortest path algorithm, such as Dijkstra's algorithm or the A* algorithm, a link (i, j) with a heavier weight $w(i, j)$ will have a smaller probability to be included in the shortest path P_{st} from node s to node t. If we set $w(i, j)$ = INFINITY, then link (i, j) will never be included in P_{st}, provided alternate paths with less than infinite length are available. This can also be achieved by deleting link (i, j) from the network. Some literature uses this idea to calculate alternative paths that are disjoint to the shortest path. Rouphail (1995) developed a pre-trip path planning scheme for providing tourists with alternate routes by increasing each link weight in P_0 (the shortest path) by 20%, 50% and 100% of its original value. Pu (2001) extended Rouphail's algorithm to seek an alternate path that has the minimum number of double- and triple-shared edges with other paths obtained thus far, using Dijkstra's shortest path algorithm and a logarithmic link weight increment procedure.

As the driver is usually more sensitive to abnormal delay, a good path should be one that reduces the probability of encountering abnormal delay as much as possible, subject to a path length constraint that prevents the path from being excessively long. Taking advantage of the idea mentioned above, we suggest a heuristic search algorithm to find a reliable path efficiently by avoiding high-risk links and turning movements as much as possible, without violating the duration constraint. This algorithm is coincident with observed driver behavior, and is therefore judged acceptable to guided drivers. In fact, driver preferences can be further met by revising the reliability of links according to the road category and the reliability of turning movements according to their direction of movement. In the algorithm, we use the A* algorithm as the basic sub-procedure to find the solution. It involves two stages; the first is to compute a reliable path by the weight increment procedure in conjunction with the A* algorithm and the second is to check the path to see if the path satisfies the duration constraint. The algorithm ends when the most reliable path that meets the duration constraint has been found. The following is a description of the algorithm:

Notation

t_{ij}	Mean normal travel time of link (i,j), $i, j \in V$
t_{ij}'	Incremented weight of link (i,j), $i, j \in V$
t_{ijk}	Normal delay of turning movement $i\text{->}j\text{->}k$, $i, j, k \in V$
t'_{ijk}	Incremented weight of turning movement $i\text{->}j\text{->}k$, $i, j, k \in V$
W_0	Large value to be added to t_{ij} or t_{ijk}
r_{ij}	Reliability of link(i,j), $i, j \in V$
r_{ijk}	Reliability of turning movement $i\text{->}j\text{->}k$, $i, j, k \in V$
$P_{s,0}$	Shortest mean normal travel time path
$L_{s,0}$	Duration of $P_{s,0}$
$\beta L_{s,0}$	Duration constraint, where β is the permission parameter (β>1)
$P_{s,n}$	Shortest path at the nth iteration
$L_{s,n}$	Duration of path $P_{s,n}$

Algorithm

***Step 1*: Set up**

Set n=0. Taking t_{ij} as the weight of link (i,j), and taking t_{ijk} as the weight of the turning movement $i\text{->}j\text{->}k$, find the path $P_{s,0}$ with the least weight using the modified A* algorithm. The duration of $P_{s,0}$ is $L_{s,0}$. If no high-risk link and turning movement is included on path $P_{s,0}$, go to Step 5.

***Step 2*: Link weight increment**

Every high-risk link and turning movement on the road network has its weight increased by Δt_{ij} and Δt_{ijk} respectively,

$$t_{ij}' = t_{ij} + \Delta t_{ij} = t_{ij} + \alpha^n (1 - r_{ij})^q W_0 ,\ 0 < \alpha \le 1,\ q=0 \text{ when } n=0,\ q=1 \text{ when } n>=1$$

$$t_{ijk}' = t_{ijk} + \Delta t_{ijk} = t_{ijj} + \alpha^n (1 - r_{ijk}) W_0 ,\ 0 < \alpha < 1,\ q=0 \text{ when } n=0,\ q=1 \text{ when } n>=1$$

$$n := n + 1$$

***Step 3*: Reliable path computation**

Taking t_{ij}' and t_{ijk}' as the weights of link (i, j) and turning movement $i\text{->}j\text{->}k$ respectively, recalculate the shortest path $P_{s,k}$ with the A* algorithm. Then restore the link weight to its original value and compute $L_{s,n}$, which is the sum of t_{ij} and t_{ijk} on the path $P_{s,n}$.

***Step 4:* Path checking**

If $P_{s,n}$ meets the duration constraint , i.e. $L_{s,n}<\beta L_{s,0}$, go to Step 5.
Else go to Step 2.

***Step 5:* Termination**

$P_{s,n}$ is the most reliable path that meets the duration constraint.

In our algorithm, the weights of every high-risk link and turning movement are initially increased separately by W_0 (when $n = 0$), which ensures that the new shortest path totally avoids them. However, if the duration constraint is violated, meaning that the path is too circuitous, then some high-risk links and turning movements currently avoided may need to be included. The weight increments Δt_{ij} and Δt_{ijk} are reduced and the least weight path is recalculated.

When $n >= 1$, the functions $\alpha^n(1-r_{ij})$ and $\alpha^n(1-r_{ijk})$ ensure that Δt_{ij} and Δt_{ijk} are reduced (because $0 < \alpha < 1$) with increasing iterations; nevertheless, the weights of higher risk links and turning movements are reduced less than those of links and turning movements of relatively lower risk, so as to ensure that they have a lower probability of appearing in the recommended path.

In the algorithm, W_0 should be large enough to ensure than all the high-risk components (links and turning movements) are avoided in the first iteration. However, the efficiency of the algorithm can be improved by reducing W_0 is as much as possible. We suggest $W_0 = \gamma L_{s,0}$, where the value of γ can be between 1.5 and 3. Generally, the denser and less congested the road network, the smaller γ can be.

To get a more exact solution, Δt_{ij} and Δt_{ijk} should be "dither changed", which means that as soon as the constraint is met after a reduction of Δt_{ij} and Δt_{ijk}, they should be increased again to a value between the value of the last iteration and the value of the present iteration, as there may exist a path, avoiding a higher risk component but still meeting the duration constraint. However, such a dither procedure would result in a heavy computation, which may be not be suitable for real time path search; therefore, a monotone weight reduction strategy is used in this paper. Although it may not be optimum, it is clearly faster and yields acceptable results. In the implementation of this algorithm α can be adjusted to balance the number of iterations and the accuracy of the solution. The larger α is, the larger the number of iterations and the greater the likelihood that the optimum path is found.

The permission parameter β can take different values for drivers with different preferences. The more risk-averse the driver, the larger the value of β as more risky links and turning movements are avoided at the expense of longer paths.

An assumption of component failure independence is made, so the reliability of a path is the product of the reliability of all the links and turning movements on the path. This can be taken as a criterion to assess path reliability.

6 Learning version of constrained reliable A* algorithm

As mentioned, two lists are used in the A* algorithm, namely the closed list and the open list. The nodes that have been explored and expanded are placed in closed list, while the nodes that have been explored but not expanded are placed in open list. For every node n that has been placed in the closed list, $g(n)$ is the shortest path cost from the start point to n. For every node n that is not in the closed list, $g(n)$ is greater than or equal to the shortest path cost from the start point to n.

We make use of the fact that $g(n)$ is the shortest path cost from the start point to node n, if n has been placed in the closed list, to improve the efficiency of subsequent A* searches. In the first step of the constrained reliable A* search described in Section 5, we first compute the normal shortest path from the destination node to the start node and then for the nodes n that have been put in the closed list, we store the corresponding value of $g(n)$ in the memory. Subsequent A* searches are performed from the start node to the destination node, making use of $g(n)$ stored in memory as the heuristic function $h(n)$. For the nodes that are not in the closed list but that are explored in subsequent searches, an estimate based of the Euclidean distance to the destination, divided by the maximum speed, is taken for $h(n)$. The calculation time is fast with a limited area being explored. Most nodes in the explored area have been placed in the closed list of the original A* search.

We should note that in the first step, although we scan and expand from the destination node, what we really want to get is the minimum cost from the origin node to the destination node as well as the minimum cost from node n to the destination. Therefore, the A* algorithm with turning penalties requires modifications, in which the open list is initialized by inserting the destination node d and by initializing the labels as follows:

$T_{i,0} = \infty$, $T_{i,j} = \infty, \forall i \in V$ *and* $j \in \Gamma^{-1}(i)$ in the original graph

$T_{d,0} = 0$, $T_{d,p} = 0, \forall p \in \Gamma^{-1}(d)$ in the original graph

Then, in the latter node scanning and checking procedure, let

$t_{ij} := t_{ji}$

$t_{ijk} := t_{kji}, \forall k \in \Gamma^{-1}(j)$ at original graph

Thus, the value of $T_{j,0}$ obtained after the A* search expanding from the destination node to the origin node in the first step of the constrained reliable A* search is actually an upper bound of the least travel time from the beginning of link (j, i) to the destination node, while the value of $T_{j,k}$ obtained is actually an upper bound of the least travel time from the end of link (k, j) to the destination d.

Therefore, in the following A* sub-procedure for reliable path search from the start node to the destination, at each weight increment iteration the estimation function of node j should be

$$f(j) = T_{j,0} + h(j) \qquad (5)$$

where $T_{j,0}$ is an upper bound of the least travel time from the origin node to node j, the recorded predecessor node is node k, $h(j)$ is the estimate of the travel time from the end of link(k, j) to the destination node. If node j has been put in the closed set in the first step of the constrained reliable A* search, then $h(j)$ could be substituted by $T_{j,k}$ obtained from the first step, which is the label of node j from the closed list, stored in memory. To further improve the efficiency of the algorithm, for j not put in the closed list an estimate based on the Euclidean distance divided by the maximum speed could be multiplied by a parameter (>=1) to reflect intersection delays.

7 Numerical experiment

The algorithms were tested for different randomly generated grid networks and conditions. At first the results for a small network with 36 nodes and 60 links under three conditions are discussed to illustrate the logic of reliable path search at the start of the trip. Then the results for a relatively large network with 2800 nodes are discussed in order to illustrate the effectiveness of the iterative constrained A* path search. The networks are subject to left-hand-drive rule. All the left turn delays are assumed to be 0. The links with a reliability lower than 0.9 are regarded as high-risk links.

The smaller network discussed in this paper is shown in Figure 3. A node is represented by a circle and the node number is shown in the circle. Links, represented by thin grey lines, are drawn to scale. The normal mean speed (kilometer/hour) and its reliability for the direction from right to left or top to the bottom are shown next to the corresponding link. Mean normal speeds lie between 30 and 60 (kilometers/hour) and reliability values lie between 0 and 1. The mean normal delays of right turns and straight forward movements are randomly generated values between 0 and 2 (min). The delay for right turns is nevertheless somewhat higher. High risk turning movements are 24->20->19, 24->20->16, 14->10->9 and 14->10->6 and their reliabilities are 0.6, 0.7, 0.8 and 0.85 respectively. The start and end nodes are represented by black rectangles. The number of the start node is 24 and the number of the destination node is 35. The duration permission parameter β is 1.1.

In Figure 3, the shortest path based on link mean normal travel times is represented by a thick black line, the cost of which is 36.7 (min) and the reliability of which is 0.22 under the link failure independence assumption. The reliable path without the duration constraint is represented by a thick grey line (or a green line in the colored graph), the cost of which is 41.1 (min) and the reliability of which is 0.69. The reliable path with the duration constraint is represented by a thick dashed line (or a red dashed line in the colored graph), the cost of which is 40.5 (min) and the reliability of which is 0.45. It is obvious that path reliability is improved by avoiding high-risk links and turning movements, for example in the reliable path without the duration constraint, the link from node 18 to node 19 (link reliability is 0.81), the link from node 10 to node 9 (link reliability is 0.73), the turning movement 24->20->29 and the turning movement 14->10->9 are avoided. However, in the reliable path with the duration constraint, some high-risk links and turning movements, like the one from node 10 to node 9 (link reliability is 0.73) and the turning movement 14->10->9, are included to meet the constraint. The reliability is nevertheless still higher than it is for the shortest path without considering component reliabilities at all.

The larger networks discussed in this paper are shown in Figure 4 and Figure 5. In these networks, the link mean normal travel speeds are randomly generated between 30 and 60 (kilometers/hour). The mean normal delays of turning movements are randomly generated and their value range is the same as in Figure 3. The reliability of a link is a randomly generated value between 0.5 and 1.0. The number of the start node is 1070, the number of the destination node is 2144, and the duration permission parameter β is 1.1.

In Figure 4, the normal shortest path is searched first from the destination node to the start node and the number of explored nodes is 425. In Figure 5, a new reliable path totally avoiding high-risk links, is searched using the closed list information from the A* shortest path search for the normal shortest path from the destination node to the start node. The reliable path is represented by a thick grey line (or a green line in the colored graph), the normal shortest path is still shown as a thick black line and the explored nodes during the reliable path search are expressed by the small black circles (red in colored graph). The number of nodes explored is 121. Hence, the reliable route search is far more efficient with than without prior information, when the number of explored nodes would have been 462. From Figure 4 we can also see that more nodes near the start node in the explored area are expanded than far from the start node. The reason is that in the normal shortest path search from the destination node to the start node, less nodes near the start node are expanded and placed in the closed list; therefore, no former information can be used for the $h(n)$ estimate for the node n near the start node in the reliable path search.

When the duration constraint is considered, several iterations have to be performed. As the iteration number increases, less weight is added to the unreliable links, so $h(n)$ approaches the actual value, taking advantage of previously computed information, reducing the number of expanded nodes further. In the above case, we take α=0.7 and γ=1.5 for the same OD pair. The reliable and acceptable path is obtained after 3 iterations and the number of expanded nodes is 121, 103 and 87 respectively.

More numerical results show that the expanded nodes number in each iteration can usually be reduced by between 1/5 and 1/3 compared to the case where no former information is used.

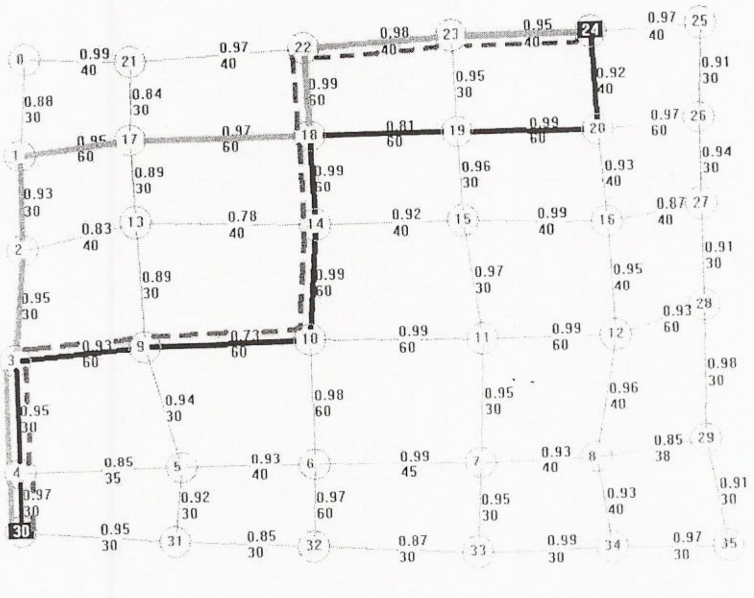

Figure 3: Comparison of shortest normal mean travel time path and the reliable path under a duration constraint

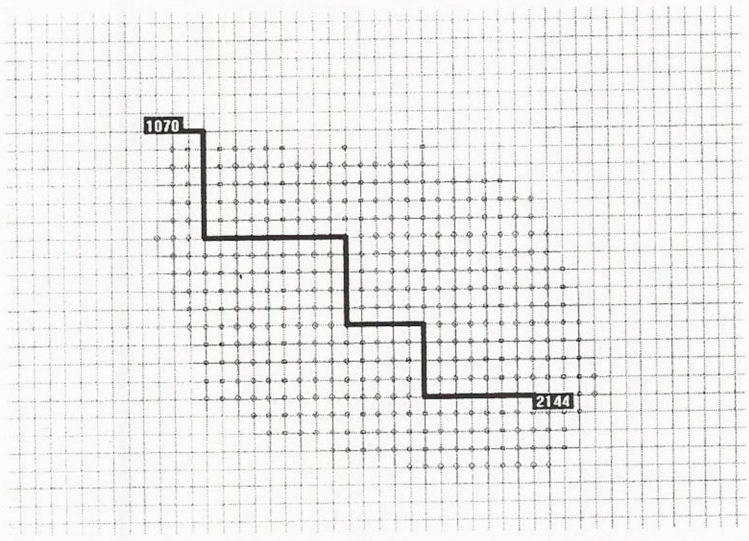

Figure 4: A* search for the normal shortest path from destination node to start node

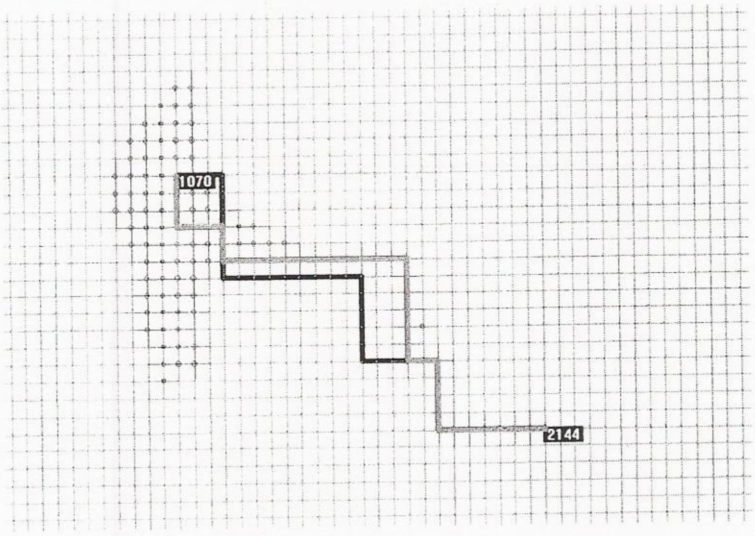

Figure 5: A*search for the reliable path avoiding all high-risk links using the former information

8 Conclusions and future work

In traditional autonomous navigation systems where no dynamic data transmission between the vehicle and a TIC takes place, route guidance is generally based on static average traffic information preloaded to the vehicle. Such a solution however sometimes deviates considerably from the actual fastest path in a dynamic traffic environment.

Therefore we propose a more reliable strategy for autonomous navigation. In this strategy, default static travel information profiles, which include mean normal travel times and their reliabilities, are built and stored on disk in the vehicle. Reliability is taken into account in route guidance.

In this paper, a constrained A* algorithm is proposed to search for a reliable path subject to an acceptable duration constraint. Turning delays and their uncertainty at intersections are also considered in the reliable path search. By utilizing the previously computed shortest path information, the constrained reliable A* can be executed efficiently.

Given that a fast and reliable path is a key requirement for drivers in autonomous navigation systems, efficient memory use and real time search are important. We plan to continue research in this direction. In particular, we will perform more extensive studies to find effective ways of building static traffic data profiles. This will include simulation experiments to uncover travel time distributions and simplified methods of determining the reliabilities of links and intersections. We will also further improve the efficiency of the constrained reliable A* algorithm by choosing better parameters or functions for the weighting procedure.

Acknowledgements

The authors would like to thank BMW UK for their help in providing a vehicle for field trials.

References

Adler, J.L. (2001) Fuzzy adaptive path evaluation for intelligent route guidance systems, *Transportation Research C*, 9 (1), pp. 1-14.

Bates, J. Polak, P. Jones, A. Cook (2001) The valuation of reliability for personal travel. *Transportation Research E*, 37, pp. 191–229.

Bell, M. G. H., C. Cassir (2002) Risk-averse user equilibrium traffic assignment: an application of game theory, *Transportation Research B*, 36, pp. 671– 681.

Clark, S. and D. Watling (2004) Modelling network travel time reliability under stochastic demand, *Transportation Research B*, In Press, pp. 1-22.

Climaco, J.C.N. and E.Q.V. Martins (1982) A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11, pp. 399-404.

Dijkstra, E.W. (1959) A note on two problems in connexion with graphs, Numerische Mathematik, 1, pp. 269-271.

Eppstein, D. (1998) Finding the K shortest paths, *SIAM J. Computing*, 28(2), pp. 652-673.

Ensor, M. (2004) A procedure for evaluating the trip reliability benefits from individual roading project, *Proceedings of 2nd International Symposium on Transport Network Reliability*, New Zealand, pp. 168-173

Fu, L. P. and L.R. Rilett (1998) Expected shortest paths in dynamic and stochastic traffic networks, *Transportation Research B*, 32 (7), pp. 499-516.

Hart, E. P., N.J. Nilsson and B. Raphael (1968) A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Syst. Sci. Cybern.*, SSC-4(2), pp. 100–107.

Kirby R. F. and R. Potts (1969) The minimum route problem for networks with turn penalties and prohibitions. *Transportation Research*. 3. pp. 397-408.

Lam, T. C. and K.A. Small (2001) The value of time and reliability: measurement from a value pricing experiment. *Transportation Research E*, 37, pp. 231–251.

Liu, H.X., W. Recker and A. Chen (2004) Uncovering the contribution of travel time reliability to dynamic route choice using real-time loop data, *Transportation Research A*, 38 (6), pp. 435-453.

Miller-hooks E.D., H.S. Mahmassani (2000) Find least expected time paths in stochastic, time-varying transportation networks, *Transportation Science*, 34 (2), pp. 198-215

Moore E. F. (1957) The shortest path through a maze. *Proc. In/. Conf: on the Theory of Switching*. Harvard Univ. Cambridge, MA.

Noland, R. B. et al. (1998) Simulating travel reliability, *Regional Science and Urban Economics*, 28, pp. 535–564.

Pattanamekar, P. et al. (2003) Dynamic and stochastic shortest path in transportation networks with two components of travel time uncertainty, *Transportation Research C*, 11 (5), pp. 331–354.

Pu, J., E.G. Manning, G.C. Shoja and A. Srinivasan (2001) A new algorithm to compute alternate paths in reliable OSPF (ROSPF), *Proc.Int. Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA 2001)*, 1, pp. 299-304

Rouphail, N. M. et al. (1995) A decision support system for dynamic pre-trip route planning, *Proceedings of the 4th ASCE International Conference on Application of Advanced Technologies in Transportation (AATT)*, New York, pp. 325-329.

Sen, A. et al. (1998) Estimation of Static Travel Times in A Dynamic Route Guidance System—2, *Mathl. Comput. Modelling*, 27 (9-11), pp. 67-85.

Sen, A., et al. (1999) Variances of link travel time estimates: implications for optimal routes, *International Transactions in Operational Research*, *6*, pp. 75-87.

Tarjan R. E. (1983) Data structures and network algorithms. *SIAM Regionul Ser. Mafh. SIAM*, Philadelphia, PA.

Yagar S., (1971) Dynamic traffic assignment by individual path minimization and queuing. *Transportation Research*, 5, pp. 179-196.

Ziliaskopoulos A. K., S. Mahmassani (1996) A note on least time path computation considering prohibitions for intersection movements, Transpn. *Transportation Research B*, 30 (5), pp. 359-367