Transportation Research Part B 43 (2009) 97–107

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

Hyperstar: A multi-path Astar algorithm for risk averse vehicle navigation

Michael G.H. Bell *

Department of Civil and Environmental Engineering, Imperial College London, London SW7 2BU, United Kingdom

ARTICLE INFO

Article history:
Received 13 January 2008
Received in revised form 30 May 2008
Accepted 30 May 2008

Keywords:
Robust route guidance
Vehicle navigation
Uncertain networks

ABSTRACT

The Astar algorithm, which forms the backbone of vehicle navigation systems, is capable of producing only one path. Given uncertainty about link travel times, there is interest in algorithms that can deliver all the paths that may be optimal, termed collectively a hyperpath, to improve travel time reliability. The actual path taken within the hyperpath will typically be determined by on-trip events, like incidences of congestion leading to delay. In this paper, the Spiess and Florian algorithm for generating hyperpaths in transit networks is adapted to road networks by assuming that drivers follow a risk averse strategy whenever a choice of path arises. To improve the efficiency of the resulting algorithm for vehicle navigation applications, the Astar approach to link selection is incorporated, leading to the Hyperstar algorithm. Proof of the optimality of the algorithm is provided, followed by numerical examples.

1. Introduction

Rapid shortest path methods, along with digital maps and satellite locationing, have made possible the development of low cost vehicle navigation systems, which are now spreading rapidly through the vehicle fleets of the developed and developing worlds. Shortest path methods have been extended to generate alternative routes, perhaps to avoid sites of congestion or specified areas (Tomtom One, for example, allows specified roads or the congestion charging zone in London to be avoided). In Europe, the traffic message channel (TMC) has been used to broad cast congestion warning messages, giving the location and type of each incidence of congestion. These messages can be received by the guidance unit, leading to on-trip rerouting. If the driver deviates from the recommended path, on-trip rerouting will also be required. Hence, the speed of the shortest path algorithm is important.

Since its introduction by Hart et al. (1968), the Astar shortest path algorithm has played a pivotal role in cybernetics and more recently in satellite navigation systems for road vehicles. The algorithm constitutes a speed up of Dijkstra's classic shortest path algorithm (Dijkstra, 1959), as described later.

In summary, Dijkstra's algorithm builds a shortest path tree from the *source* node to all other nodes. The algorithm maintains two lists, an *open list* from which the next node for expansion is selected and a closed list of nodes that have been expanded. Initially, the source is assigned to the *open list*. At each step, the node with highest priority is selected from the open list, the links leaving the selected node are added to the tree, the nodes reached thereby are added to the open list, and the selected node is transferred from the open list to the closed list. Priority for selection is determined by distance from the source by the best route discovered so far. The algorithm terminates when the open list is empty or when the *target* node is transferred to the closed list. A succinct description of Dijkstra's algorithm is to be found in Wagner and Willhalm (2006).

Various attempts to speed up the shortest path search for the case of a single source and single target are reviewed in Wagner and Willhalm (2006) and compared empirically in Bauer et al. (2007). Wagner and Willhalm (2006) identify

* Tel.: +44 2075946091; fax: +44 2075946102.
E-mail address: m.g.h.bell@imperial.ac.uk

0191-2615/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.trb.2008.05.010

bidirectional search and *goal-directed search* as the two classical speed-up techniques. With bidirectional search, two searches are performed, one "normal" or forwards search from the source and the other "reverse" or backwards search from the target. The algorithm terminates when one node is designated closed by both searches. Goal-directed search, more widely known as Astar search, introduces node *potentials* to modify the priority of the nodes in Dijkstra's open list, thereby orienting the search toward the target. A suitable potential is an underestimate of the remaining distance to the target.

The accuracy of the estimate of the remaining distance to the target governs the efficiency of the Astar algorithm. If the estimate is accurate, the algorithm expands only those nodes that lie on an optimal path. If the estimate is a constant (for example, zero), the Astar algorithm is the same as the unmodified Dijkstra algorithm. If the estimate is not less than the actual remaining distance, the Astar algorithm may terminate before finding the optimal path. Hart et al. (1968) provide the optimality conditions for the Astar algorithm, together with proofs.

Upon termination, Dijkstra's algorithm, whether sped up or otherwise, delivers not only the shortest path, but also the least distance from the source to every node expanded in the search (if the search is from the source). This opens up the possibility of searching initially from the target rather than the source. Then, should the vehicle deviate from the recommended route, a new shortest path to the target may usually be calculated very quickly searching from the current location as the remaining distance to the target will be known accurately for those nodes within the envelope of nodes expanded by the initial search. It is conjectured that this technique is used by current vehicle navigation systems, accounting for their ability to reroute rapidly.

In reality, there is uncertainty about link travel times which should be taken into account during path generation. While congestion warning messages leading to on-trip rerouting can avoid the worst consequences of congestion, the driver may still end up on a sub-optimal route. There is increasing interest in the generation of sets of paths offering good alternatives should congestion materialise after the commencement of the trip so as to improve travel time reliability. This problem has been examined by Chen et al. (2007) using penalty methods.

Where there is uncertainty about link travel times, there will be a set of paths that may be optimal, referred to collectively as a *hyperpath*. Choice of a particular path from the hyperpath is then governed by on-trip events, like the incidence of congestion, or by other factors. There are a number of other reasons why a set of paths are useful. With the rapid spread of vehicle navigation systems, there is an increasing risk that frequently recommended routes become overloaded, so traffic should be spread across a set of paths. Moreover, drivers have diverse preferences which may not be fully captured by the fastest path. Park et al. (2007) studied methods for learning driver preferences for a range of link and path attributes by analysing path acceptance/rejection in real time.

The next section reviews previous work on path set generation. This is followed by an introduction to the hyperpath concept in its original setting, namely transit assignment. An adaptation of the Spiess and Florian algorithm to find hyperpaths in traffic networks is then presented, starting with an equivalent optimisation problem. The equivalence between the risk averse choice strategy and minmax exposure strategy is proven. Potentials are then introduced into the adapted Spiess and Florian algorithm to yield the Hyperstar algorithm, and the optimality of the solution produced by the algorithm is proven via two propositions. The paper culminates with numerical examples and conclusions.

2. Review of path set generation work

Chen et al. (2007) review methods for generating multiple paths in the context of route guidance. Traditionally, alternative paths have been calculated by two categories of algorithm in graph theory; the k-shortest path algorithms proposed by Eppstein (1998), Martins et al. (1999) and Jimenez and Marzal (1999), and the totally disjoint path algorithms proposed by Dinic (1970) and Torrieri (1992). However, these algorithms have some drawbacks for route guidance. For the k-shortest path algorithms, after the paths are sorted, an additional path checking procedure has to be performed to choose paths that satisfy acceptability constraints. Therefore, when the number of paths from a source to a target is large, the k-shortest path algorithms become inefficient for finding alternative paths that are acceptable. Moreover, alternative paths may overlap excessively. On the other hand, for the disjoint path algorithms, the primary shortest path of the network may not be included or the length of alternative paths may be unacceptable.

Recently, literature has appeared on searching for k partly disjoint paths subject to some constraints and the link penalty method presented in Chen et al. (2007) falls into this category. Akgun et al. (2000) find a dissimilar path set by measuring the spatial dissimilarity between any two paths in the path set. Paths are chosen to maximise dissimilarity, which is measured by the minimum distance between it any other path in the dissimilar path set. Rouphail (1995) develops an algorithm to provide tourists with alternative routes by increasing link impedance by 20%, 50% and 100% of its original value. Pu et al. (2001) extends Rouphail's algorithm to seek an alternate path that has the minimum number of double- and triple-shared links, using Dijkstra's shortest path algorithm and an incremental logarithmic link penalty procedure. Such incremental link penalty algorithms can generate paths which share as few links as possible, avoid unreliable links and conform to reasonability criteria (see Chen et al., 2007).

The problem of generating multiple paths also arises in route set generation for the analysis of path choice behaviour (see Prato and Bekhor, 2006, for a review). The generated route set should exclude unrealistic paths that no traveller would consider and highly similar paths that no traveller would ever differentiate between. Deterministic and probabilistic approaches to route generation have been proposed.

Van der Zijpp and Fiorenzo-Catalano (2005) employ a *k*-shortest path method together with a range of constraints to generate a path set. Ben-Akiva et al. (1984) assume that travellers have different objective functions and each objective yields a different route. Dial (2000) constructs a set of efficient paths. Azevedo et al. (1993) remove all the shortest path links from the network to find the next best path. However, link elimination runs the risk of network disconnection. This problem can be avoided by eliminating individual links or combinations of links from the shortest path. Barra (1993) present a link penalty approach, whereby the impedances of the shortest path links are increased in order to find the next best path. Park and Rilett (1997) modify this approach by not increasing the impedance on links within a certain distance of the source or the target. Scott et al. (1997) determine penalties for shortest path links in order to generate a next best path that overlaps with the shortest path by no more than a given number of links.

Simulation offers an alternative approach to path set generation. Assuming that travellers erroneously perceive link attributes, Sheffi and Powell (1982) combine random draws from a distribution representing drivers' perceptions with shortest path calculations. Bekhor et al. (2001) verify the suitability of simulation methods to produce paths similar to those observed for a real case study in Boston.

3. Concept of the hyperpath

The concept of the hyperpath, namely a set of paths any one of which may be optimal, emanates from the field of transit assignment (Spiess and Florian, 1989; Nguyen and Pallotino, 1988) and is associated with the concept of *common lines*. When arriving at a bus stop or a train station it is often the case that there are a number of attractive paths and the choice of which to take depends on which line happens to arrive next. Under the "take whichever attractive line arrives next" rule, and assuming that lines arrive randomly with given frequencies, it is possible to determine which paths (and therefore which lines) could be optimal and are therefore attractive. The lines that are attractive at a given stop or station hence constitute the common lines referred to earlier.

Spiess and Florian (1989) show that under the "take whichever attractive line arrives next" rule the hyperpath may be found by minimising the expected travel time and that the resulting problem (looked at in more detail in the next section) is a linear program. They also provide an algorithm resembling Dijkstra's algorithm to solve the linear program which works outwards from the target. In this paper, we note the parallel between link frequency and link delay, and extend the Spiess and Florian algorithm by adding node *potentials* into the link selection step, yielding an algorithm that resembles the Astar algorithm, but which generates a hyperpath. This we refer to as the Hyperstar algorithm. For completeness, we present proofs of optimality via two propositions.

4. Adaptation of the Spiess and Florian algorithm

Consider a road network consisting of a set of links A and a set of nodes I. In Spiess and Florian (1989), each link a has a service frequency the inverse of which defines the expected link waiting time. In our case, road links also provide a service that may be subject to a delay. We interpret the inverse of this service frequency as defining the maximum link delay. A high service frequency therefore corresponds to a link with a low maximum delay and hence a reliable travel time, and vice versa. Drivers are assumed to be interested in all potentially optimal paths when adopting a minmax exposure to delay strategy.

Define the following sets and variables:

A	Set of links
I	Set of nodes
H	Set of links containing the hyperpath between source r and target s
A_i^+	Set of links out of node i
A_i^-	Set of links into node i
d_a	Maximum delay on link a
p_a	Probability link a is used
c_a	Undelayed travel time on link a
u_i	Least travel time from node i to target s expected by a pessimist who minimises maximum exposure to delay
w_i	Exposure to maximum delay at node i
h_i	Potential at node i with respect to source r
N	A large number whose size depends on the precision of computation

Following Spiess and Florian (1989), the hyperpath from source r to target s is identified by the following linear program:

$$\mathrm{Min}_{\mathbf{p},\mathbf{w}} \sum_{a\in A} c_a p_a + \sum_{i\in I} w_i \tag{1}$$

subject to

$$\sum_{a\in A_i^+} p_a - \sum_{a\in A_i^-} p_a = g_i \quad i \in I \tag{2}$$

$$p_a d_a \leqslant w_i \quad a \in A_i^+, i \in I \tag{3}$$

$$p_a \geqslant 0 \quad a \in A \tag{4}$$

The objective function (1) is the expected travel time. Note that expected node delay is interpreted as that which would be expected by a pessimistic driver, namely his exposure to maximum link delay. If link a is the only link out of node i included in the hyperpath, then the driver is fully exposed to maximum delay on that link and $w_i = d_a$. To mitigate this exposure, the driver requires alternative ways out of node i.

Constraint two enforces flow conservation. Note that $g_i = 1$ if $i = r$, $g_i = -1$ if $i = s$, and $g_i = 0$ otherwise. Constraint (3) ensures that the usage of link a is inversely proportional to its maximum delay, if link a is used ($p_a > 0$). If link a is not used it will not be in the hyperpath; if it is used it is in the hyperpath.

The following algorithm both solves the above problem and provides insight into the properties of the solution:

A_0	Spiess and Florian algorithm
0. Initialisation	$u_i \leftarrow \infty$, $i \in I - \{s\}$, $u_s \leftarrow 0$; $f_a \leftarrow 1/d_a$ if $d_a > 0$, otherwise $f_a \leftarrow N$; $f_i \leftarrow 0$, $i \in I$; $y_i \leftarrow 0$, $i \in I - \{r\}$, $y_r \leftarrow 1$; $L \leftarrow A$; $H \leftarrow \varnothing$.
1. Select link a	Find $a = (i, j) \in L$ with minimum $u_j + c_a$; $L \leftarrow L - \{a\}$.
2. Update node i	If $u_i \geqslant u_j + c_a$ then if $u_i = \infty$ and $f_i = 0$ then $\beta \leftarrow 1$ else $\beta \leftarrow f_i u_i$, $u_i \leftarrow (\beta + f_a(u_j + c_a))/(f_i + f_a)$, $f_i \leftarrow f_i + f_a$ and $H \leftarrow H + \{a\}$; if $L = \varnothing$ or $u_j + c_a > u_r$ go to Step 3 else go to Step 1.
3. Loading	For every link $a \in A$ in decreasing order of $u_j + c_a$, if $a \in H$ then $p_a \leftarrow (f_a/f_i)\, y_i$ and $y_j \leftarrow y_j + p_a$ else $p_a \leftarrow 0$.

Variable u_i captures the expected travel time from node i to the target s. In the original Spiess and Florian algorithm, f_i captures the combined frequency of services leaving node i on attractive links. After initialisation, the algorithm begins at the target and finds the node closest to the target (Step 1). Suppose the link from this node to the target is a. Link a is removed from set L. Expected travel time from this node to the target is then updated (Step 2) to $1/f_a + c_a = c_a + d_a$, which is the undelayed link travel time plus the maximum delay, or the maximum link travel time. Link a is added to the hyperpath. The justification for the convention that $f_i u_i = 1$ when $u_i = \infty$ and $f_i = 0$, and therefore that $u_i = c_a + d_a$, is that the driver is fully exposed to the risk of a delay of up to d_a on link a until another link leaving node i is added to the hyperpath. Being risk averse, and therefore pessimistic, the driver expects the worst.

The algorithm then returns to Step 1 and selects a new link. The selection criterion ensures that the selected link $a' = (i', j')$ will lead to a node that was previously updated or the target. If use of this link would enable the mean travel time from the entrance node to the link to the target to be reduced, then $u_{i'}$ is updated and a' is added to the hyperpath. In the Spiess and Florian (1989) algorithm, a weighted mean travel time is calculated in Step 2, where the relevant link service frequencies are chosen as the weights.

The algorithm continues to return to Step 1 until all links have been selected and L is empty or $u_j + c_a$ for selected link a is larger than u_r. The second termination criterion has been added, because we seek only one hyperpath, from r to s. Proof of this termination criterion follows from Proposition 3 below, which in turn follows from Proposition 2 below when $h_i = 0\ \forall i \in I$.

The expected travel time from each node i to the target s, u_i, is calculated in Step 2 on the basis that the probability of link usage is inversely proportional to the maximum link delay. Link usage is calculated in Step 3.

4.1. Usage of attractive links

The link usage implicit in the proposed adaptation of the Spiess and Florian (1989) algorithm is justified if drivers are risk averse. To illustrate this, suppose that two attractive links exit node i, one with a maximum delay of d_a and the other with a maximum delay of $d_{a'}$, where $a = (i,j)$ and $a' = (i,j')$. Suppose further that p_a is the probability that a given driver chooses link a. Then the maximum exposure to delay on link a is $p_a d_a$ and the maximum exposure to delay on link a' is $p_{a'} d_{a'} = (1 - p_a) d_{a'}$. The cautious driver would choose p_a to minimise his maximum exposure to delay on the two links.

Proposition 1. Minmax exposure implies that $p_a d_a = w_i > 0$ if $p_a > 0$ and $d_a > 0$ for i such that $a \in A_i^+$.

Proof 1. The Lagrangian function for problem (1)–(4) is

$$L_{\mathbf{p},\mathbf{w},\mathbf{q},\mathbf{v}} = \sum_{a \in A} c_a p_a + \sum_{i \in I} w_i + \sum_{i \in I} \sum_{a \in A_i^+} q_a (p_a d_a - w_i) + \sum_{i \in I} v_i \left(\sum_{a \in A_i^+} p_a - \sum_{a \in A_i^-} p_a - g_i \right) \tag{5}$$

where **q** and **v** are the Lagrange multipliers. According to strong duality theory, the solution to problem (1)–(4) can be found by minimising $L_{\mathbf{p},\mathbf{w},\mathbf{q},\mathbf{v}}$ with respect to $\mathbf{p} \geqslant 0$ and **w** while maximising it with respect to $\mathbf{q} \geqslant 0$ and **v**. If $p_a > 0$, then from (3) we know that $w_i > 0$ for i such that $a \in A_i^+$, since $d_a > 0$. Note that according to the complementary slackness conditions:

$$\frac{\partial L_{\mathbf{p},\mathbf{w},\mathbf{q},\mathbf{v}}}{\partial w_i} = 1 - \sum_{a \in A_i^+} q_a = 0 \Rightarrow \sum_{a \in A_i^+} q_a = 1$$

at the solution, because $w_i > 0$. If $q_a = 0$, p_a could be reduced to 0 at the solution, violating the assumption that $p_a > 0$. So, by the complementary slackness conditions, $q_a > 0$ implies that $p_a d_a = w_i > 0$ for i such that $a \in A_i^+$. □

According to Proposition 1, objective (1) together with constraints (3) ensure that $p_a$will be inversely proportional to d_a if $p_a > 0$ and $d_a > 0$. Note that if there is only one link a out of node i then the delay expected by the risk averse driver is by definition d_a, as the risk averse driver expects the worst and is fully exposed to maximum delay on link a. He/she can reduce his exposure to maximum delay on one link by having choices.

5. Hyperstar algorithm

In this algorithm, we introduce a node *potential* to transform undelayed link travel times as follows:

$$c'_a = c_a + h_i - h_j \quad \forall a = (i,j) \in A \tag{6}$$

In contrast to Wagner and Willhalm (2006), we define the potential with respect to the source r rather than the target s. With a suitable potential, the search can be pushed towards the target thereby reducing the running time without upsetting the optimality of the algorithm. In the words of Wagner and Willhalm (2006) "one can compare a path in a traffic network with a walk in a landscape. If you add a potential, the affected region is raised. If the added potential is small next to the target, you create a valley around the target. As walking downhill is easier than uphill, you are likely to hit the target sooner than without the potential added". In our case, the search is in reverse from the target back to the source, so we use the node potentials to orient the search back to the source.

The expected travel time from j to s, using the transformed link travel times, is

$$u'_i = u_i + h_i - h_s \quad \forall j \in I \tag{7}$$

as the potentials at intermediate nodes on all paths from node j to target s cancel out. Henceforth, we set $h_r = 0$ (without this condition, h_i is not necessarily a lower bound on the travel time from r to i), so the expected trip travel time is modified by a constant as follows:

$$u'_r = u_r - h_s \quad \forall j \in I \tag{8}$$

Hence, the transformation of undelayed link costs by means of node potentials is equivalent to subtracting a constant from the objective function, leaving the solution unaltered. To show this more rigorously, consider the dual to problem (1)–(4).

Rearranging the Lagrangian function (5) and setting $a = (i,j)$ we obtain:

$$L_{\mathbf{p},\mathbf{w},\mathbf{q},\mathbf{v}} = -v_r + v_s + \sum_{(i,j) \in A} p_{(i,j)}(c_{(i,j)} + q_{(i,j)} d_{(i,j)} + v_i - v_j) + \sum_{i \in I} w_i \left(1 - \sum_{a \in A_i^+} q_{(i,j)}\right) \tag{9}$$

This leads to the dual to problem (1)–(4), namely:

$$\mathrm{Max}_{\mathbf{q},\mathbf{v}} - v_r + v_s \tag{10}$$

subject to

$$c_{(i,j)} + q_{(i,j)} d_{(i,j)} + v_i - v_j \geqslant 0 \quad (i,j) \in A \tag{11}$$

$$q_{(i,j)} \leqslant 1 \quad (i,j) \in A \tag{12}$$

$$q_{(i,j)} \geqslant 0 \quad (i,j) \in A \tag{13}$$

where the Lagrange multipliers are now **p**, for constraints (11), and **w**, for constraints (12). For $p_{(i,j)} > 0$ (in which case, link $(i,j) \in H$),

$$c_{(i,j)} + q_{(i,j)} d_{(i,j)} + v_i = v_j \quad (i,j) \in H \tag{14}$$

Now replace $c_{(i,j)}$ with $c'_{(i,j)} - h_i + h_j$. This leads to

$$c'_{(i,j)} + q_{(i,j)} d_{(i,j)} + v_i - h_i = v_j - h_j \quad (i,j) \in H \tag{15}$$

Hence, the transformation of link costs causes **h** to be subtracted from **v**, and since $h_r = 0$ the dual objective function becomes

$$-v_r + v_s - h_s \tag{16}$$

At the optimum, the result of the transformation is simply to subtract h_s from the primal objective function, leaving the solution otherwise unaltered.

The chosen potentials should satisfy two conditions; they should be feasible and they should be less than the actual expected travel time from the respective node to source r.

Assumption 1. The potentials satisfy the following triangular inequality principle:

$$h_j \leqslant h_i + c_a \ \forall a = (i,j) \in A \tag{17}$$

In this case the potentials are said to be *feasible* (Wagner and Willhalm, 2006). Without this assumption, the modified link costs may be negative. Only if the potentials are feasible is the following algorithm sure to deliver the hyperpath. This is clear from Proposition 2. If the potentials exceed the actual respective travel times, the algorithm may terminate too soon and so not deliver the hyperpath, as is clear from Proposition 3.

A_1	Hyperstar algorithm
0. Initialisation	$u_p \leftarrow \infty$, $i \in I - \{s\}$, $u_s \leftarrow 0$; $f_a \leftarrow 1/d_a$ if $d_a > 0$, otherwise $f_a \leftarrow N$; $f_i \leftarrow 0$, $i \in I$; $y_i \leftarrow 0$, $i \in I - \{r\}$, $y_r \leftarrow 1$; $L \leftarrow A$; $H \leftarrow \varnothing$.
1. Select link a	Find $a = (i, j) \in L$ with minimum $h_i + u_j + c_a$; $L \leftarrow L - \{a\}$.
2. Update node i	If $u_i \geqslant u_j + c_a$ then if $u_i = \infty$ and $f_i = 0$ then $\beta \leftarrow 1$ else $\beta \leftarrow f_i u_i$, $u_i \leftarrow \leftarrow (\beta + f_a(u_j + c_a))/(f_i + f_a)$, $f_i \leftarrow f_i + f_a$ and $H \leftarrow H + \{a\}$; if $L = \varnothing$ or $h_i + u_j + c_a > u_r$ go to Step 3 else go to Step 1.
3. Loading	For every link $a \in A$ in decreasing order of $h_i + u_j + c_a$, if $a \in H$ then $p_a \leftarrow (f_a/f_i)\, y_i$ and $y_j \leftarrow y_j + p_a$ else $p_a \leftarrow 0$.

When $h_i = 0$ for all $i \in I$, the algorithm is the same as the Spiess and Florian algorithm.

Proposition 2. At the point at which $a = (i, j)$ is selected, u_j has been reduced to its final value.

Proof 2. The proof is by induction. For all nodes $i \in I - \{s\}$, u_i is initially set to ∞ (in practice a large number) in Step 0 and then either left unchanged or reduced in Step 2. Suppose u_j is greater than its final value. There must be a path to j not so far considered offering the possibility of a shorter travel time. Without loss of generality, suppose this path is via $a' = (j,j') \in A$ where $f_{a'} > 0$ (if $f_{a'} = 0$ link a' has an infinite maximum delay and will not be used) and $a' \notin H$ (the set of links from which the hyperpath from r to s will be constructed). Hence, $c_{a'} + u_{j'} < u_j$. If $u_{j'}$ is reduced to its final value and $h_j \leqslant h_i + c_a$ (the potentials are feasible), then $h_j + c_{a'} + u_{j'} \leqslant h_i + c_a + c_{a'} + u_{j'} < h_i + c_a + u_j$ and a' would have been selected in Step 1 instead of a, which is a contradiction. Therefore, u_j' is also greater than its final value. This argument can be repeated until we reach s and conclude that $u_s = 0$ is greater than its final value, which is clearly untrue. □

The Hyperstar algorithm works backward from the target s. When a link $a = (i,j)$ is selected in Step 1, u_j must have already been reduced to its final value (the expected travel time to the target), otherwise a link not already in the hyperpath, but closer to the target would have been selected and added to the hyperpath. Note that for a pessimistic driver, u_i is the expected travel time from i to s, and so u_r is the expected travel time for the trip.

Proposition 3. If, at the point at which $a = (i,j)$ is selected, $h_i + c_a + u_j > u_r$, the algorithm should terminate.

Proof 3. If $h_i + c_a + u_j > u_r$, then the expected value of the travel time from r to s via link a is greater than the current expect travel time from r to s, as u_j is reduced to its final value (2) and h_i is less than or equal to the expected travel time from r to i (by the feasibility condition). As using link a would increase u_r, link a should not be added to H. Since link a is not added to H, $h_{i'} + c_{a'} + u_{j'} > u_r$ will also be true for any subsequently selected link $a' = (i',j') \in A \backslash H$, so the algorithm should terminate at this point. □.

With each successive link selection, $h_i + c_a + u_j$ increases, so when $h_i + c_a + u_j > u_r$ there is no possibility of further reductions in u_r provided u_j has achieved its final value, the expected travel time from j to s, and h_i is less than the expected travel time from r to i. If h_i were greater than the expected travel time from r to i, there is a risk that the algorithm may terminate too soon.

If $d_a = 0$ for all links $a \in A$ then the Hyperstar algorithm finds only the path(s) with least undelayed travel time provided the large number chosen for f_a is sufficiently large, as proven by the following proposition. In this sense, the solution to the Hyperstar algorithm tends to the solution to the Astar algorithm as d_a tends to 0 for all $a \in A$.

Proposition 4. If $d_a = 0$ for all links $a \in A$ then the Hyperstar algorithm finds only the path(s) with least undelayed travel time.

Proof 4. If $d_a = 0$ then in Hyperstar $f_a = N$ (a large number). Suppose that when link $a = (i,j)$ is selected in Step 1, node $i \neq s$ is reached for the first time. According to Step 2 of Hyperstar:

Table 1
Undelayed link travel times (all cases) and the corresponding random variable R from which maximum link delays are calculated in cases 2 and 3

i	j	$c_{(i,j)}$	R	i	j	$c_{(i,j)}$	R
r	2	1.5000	0.5313	r	9	1.8413	0.3190
2	3	1.3610	0.7027	2	10	1.3170	0.6413
3	4	1.7729	0.1717	3	11	1.5935	0.5670
4	5	1.8187	0.6617	4	12	1.5603	0.6940
5	6	1.8722	0.9257	5	13	1.1730	0.7273
6	7	1.5820	0.9747	6	14	1.0704	0.7543
7	8	1.5706	0.3410	7	15	1.8659	0.3370
9	10	1.5597	0.7933	8	16	1.8146	0.8150
10	11	1.4173	0.8680	9	17	1.1564	0.1923
11	12	1.2747	0.8280	10	18	1.4030	0.8187
12	13	1.1977	0.3443	11	19	1.9787	0.7963
13	14	1.2180	0.8067	12	20	1.2667	0.2407
14	15	1.6626	0.8293	13	21	1.0520	0.2237
15	16	1.9660	0.7467	14	22	1.9117	0.0420
17	18	1.3257	0.7790	15	23	1.0958	0.3807
18	19	1.9818	0.6493	16	24	1.1689	0.6123
19	20	1.3743	0.7873	17	25	1.9717	0.9037
20	21	1.2728	0.4763	18	26	1.6148	0.5540
21	22	1.9925	0.3880	19	27	1.0486	0.1733
22	23	1.3273	0.5203	20	28	1.5807	0.4217
23	24	1.6411	0.6437	21	29	1.3387	0.1813
25	26	1.0008	0.5217	22	30	1.4270	0.7773
26	27	1.0929	0.5097	23	31	1.2009	0.2343
27	28	1.2745	0.1507	24	32	2.0000	0.1637
28	29	1.2705	0.2080	25	33	1.6582	0.7300
29	30	1.0575	0.7060	26	34	1.4465	0.8650
30	31	1.1811	0.0930	27	35	1.6267	0.4870
31	32	1.4555	0.1440	28	36	1.8063	0.5543
33	34	1.4984	0.7437	29	s	1.6019	0.3667
34	35	1.1762	0.7957	30	38	1.0544	0.1970
35	36	1.5859	0.2723	31	39	1.6007	0.1690
36	s	1.9953	0.3817	32	40	1.5166	0.8100
s	38	1.0100	0.0733	33	41	1.4043	0.0353
38	39	1.4532	0.1850	34	42	1.8270	0.9423
39	40	1.8387	0.5077	35	43	1.4006	0.8473
41	42	1.2054	0.4890	36	44	1.8538	0.9877
42	43	1.2740	0.2497	s	45	1.5716	0.3507
43	44	1.6977	0.6437	38	46	1.2603	0.6803
44	45	1.4136	0.2843	39	47	1.4228	0.9937
45	46	1.0893	0.3647	40	48	1.6276	0.3923
46	47	1.1894	0.7877	41	49	1.9032	0.7077
47	48	1.5936	0.1290	42	50	1.3071	0.0873
49	50	1.7800	0.3773	43	51	1.6098	0.1233
50	51	1.6612	0.7733	44	52	1.4277	0.0407
51	52	1.0437	0.4607	45	53	1.8261	0.9483
52	53	1.6744	0.6570	46	54	1.7268	0.6847
53	54	1.7681	0.6773	47	55	1.1905	0.4960
54	55	1.0768	0.9980	48	56	1.1297	0.1917
55	56	1.7852	0.6560	49	57	1.0407	0.5973
57	58	1.1098	0.2397	50	58	1.2472	0.6487
58	59	1.2464	0.5803	51	59	1.7329	0.3803
59	60	1.5763	0.5597	52	60	1.2138	0.9390
60	61	1.6065	0.2957	53	61	1.4803	0.6543
61	62	1.3059	0.8817	54	62	1.5394	0.6843
62	63	1.1120	0.9367	55	63	1.5097	0.8790
63	64	1.2408	0.9940	56	64	1.8949	0.8640

$$\begin{aligned} u_i &\leftarrow \frac{1}{N} + u_j + c_a \approx u_j + c_a \\ f_i &\leftarrow N \\ H &\leftarrow H + \{a\} \end{aligned} \tag{18}$$

If link $a' = (i,j')$ is subsequently added to H then $u_j + c_a + \varepsilon = u_i \geqslant u_{j'} + c_{a'}$ where $j \neq j'$ and $\varepsilon = 1/N$. Suppose that N is large enough for $\varepsilon = 0$ within the chosen limits of accuracy. Then either $u_j + c_a + h_i > u_{j'} + c_{a'} + h_i$, in which case link a' would have been selected before link a (a contradiction), or $u_j + c_a + h_i = u_{j'} + c_{a'} + h_i$. Since $u_i = u_{j'} + c_{a'}$ within the chosen limits of accuracy, u_i is not reduced by adding link a' to H as the new value of u_i emanating from Step 2 is a weighted average of the old value of u_i and the value of $u_{j'} + c_{a'}$. Hence, when N is sufficiently large, only least undelayed travel time paths will be included in the hyperpath found by Hyperstar. □

6. Numerical example

To demonstrate the Hyperstar algorithm, we take an 8 node by 8 node grid network. All arcs can be travelled in either direction, leading to 224 directional links. The undelayed travel time on each arc is set equal to 1 + R, where R is a random number in the range 0–1 inclusive, and is the same in either direction. The target s is set near the centre of the network to demonstrate the advantages of goal oriented search. The travel times on each arc are given in Table 1, and as it is the same in either direction, only one direction is given.

As the travel time on each link is at least 1 unit, it is straightforward to calculate a good value for h_i which underestimates the travel time from r to i. For example, $h_{20} = 5$ (2 links down and 3 across, or any other reasonable route, see Fig. 1). This is a form of what is sometimes referred to in the literature as the Manhattan metric (Wagner and Willhalm, 2006).

Three cases were considered. In the first case (see Fig. 1), all links are assigned a zero maximum delay, and therefore a high service frequency. As all links are reliable, the hyperpath contains only one elemental path (since link travel times were generated randomly to a large number of decimal places, the chance of two paths having the minimum travel time is vanishingly small). The introduction of h as defined above into the algorithm reduces the number of links selected before termination from 219 to 79, a significant saving in computation.

In the second case (see Fig. 2), maximum link delay is assigned a random value equal to 0.3R, where as before R is a random value in the range 0–1 inclusive. The values of R are given in Table 1. Some links are now significantly less reliable than others. The hyperpath now contains 2 elemental paths. The non-zero link usage probabilities are shown against the respective links in Fig. 2. When the risk averse driver reaches node 2, two links are attractive. He has a preference for link (2,10), but if he learns of congestion on links (2,10) or (10,11) en route, he knows he has a reasonable alternative route via links

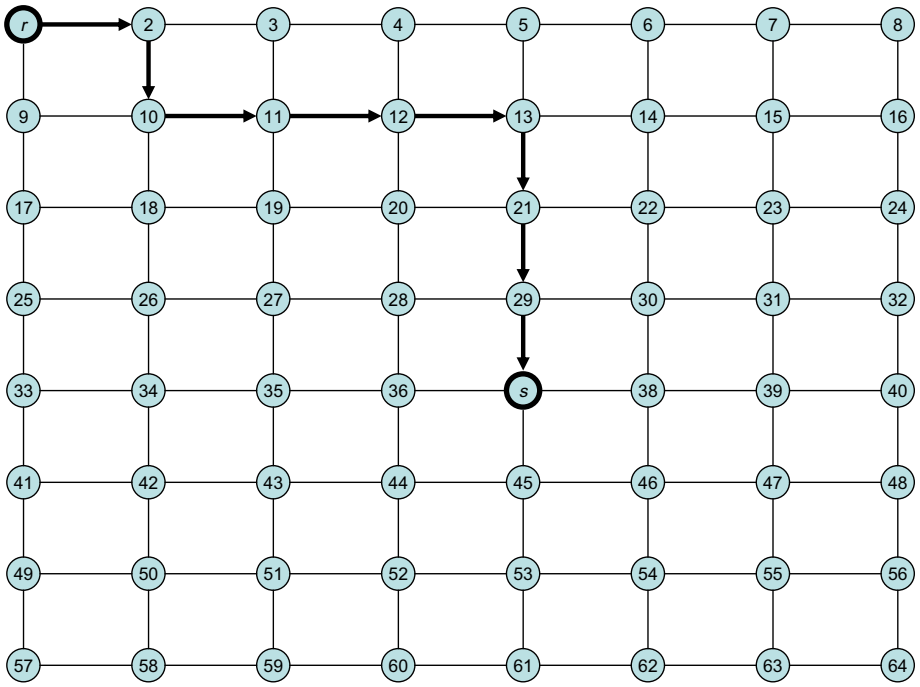

Fig. 1. Hyperpath for highly reliable links.

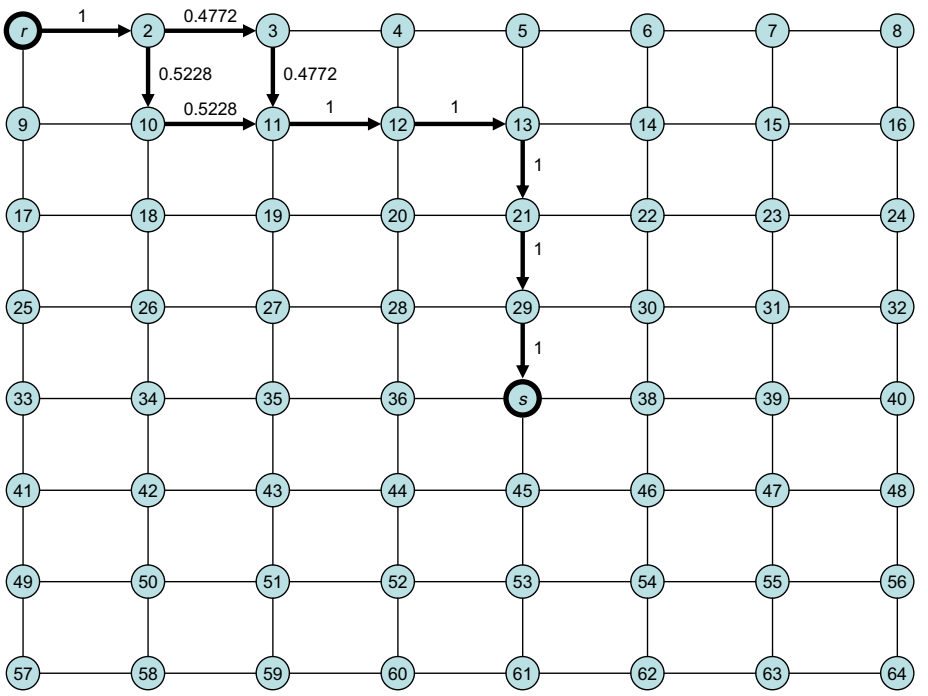

Fig. 2. Hyperpath for moderately reliable links.

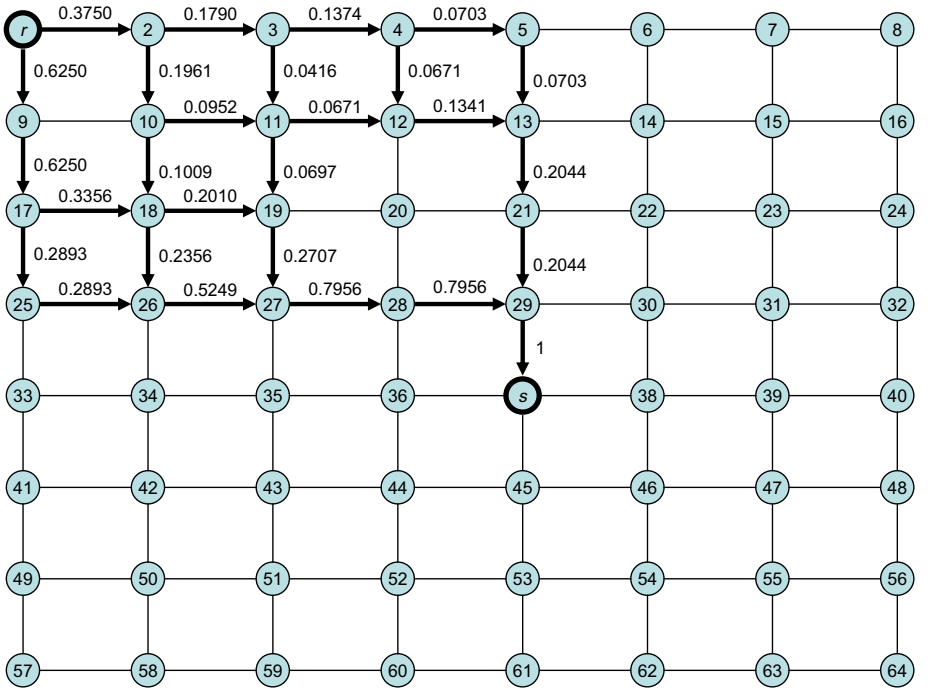

Fig. 3. Hyperpath for unreliable links.

(2,3) and (3,11). The advantage of the Hyperpath algorithm over single route algorithms is that reasonable detours are taken into account at the route generation stage. The best route in the assumed absence of delays may expose the driver to too high a risk of delay. On the other hand, assuming that every link suffers maximum delay would be too cautious as it neglects the

Table 2
Comparative performance of the Hyperstar algorithm

	Maximum delay	u_r	A_0 selected links	A_1 selected links
Case 1	$d = 0$	10.7001	219	79
Case 2	$d = 0.3R$	11.8649	222	111
Case 3	$d = R$	13.6226	223	148

benefits of detouring when the driver receives congestion information en route. The introduction of h in case 2 leads to a reduction in the number of links selected before termination from 221 to 111.

Finally, in the third case (see Fig. 3), maximum link delay is set equal to R as given in Table 1, so many links are now unreliable. This leads to a large number of elemental paths. The non-zero link usage probabilities are shown against the respective links in Fig. 3. The driver has a preference for the route $r \rightarrow 9 \rightarrow 17 \rightarrow 18 \rightarrow 26 \rightarrow 27 \rightarrow 28 \rightarrow 29 \rightarrow s$, in the sense that this is the route he will most likely choose, but has a large number of reasonable alternatives, should he learn of congestion en route. The introduction of h leads to a reduction in the number of links selected before termination of the algorithm from 223 to 148. The results are summarised in Table 2.

As the undelayed link travel times were the same in all three cases, we see that the size of potential delays can influence route preferences significantly and, as delays are increased, paths that were unattractive can become attractive.

7. Discussion

This paper offers an efficient way to generate a set of potentially optimal paths for use in car navigation systems. For links that may become congested after the commencement of the trip the driver would like to have good alternatives. Therefore, it is important to take into account the magnitude of possible link delays when paths are generated.

The method proposed in this paper does not require the iterative use of penalties, the k-shortest path algorithm or Monte Carlo simulation. Instead, the hyperpath algorithm of Spiess and Florian (1989), originally designed for transit networks, is adapted to traffic networks. In place of the "take next line" strategy, leading to a division of passengers between attractive links leaving a given node in proportion to their respective service frequencies, there is a "minmax exposure to delay" strategy, leading to a link use probability inversely proportional to maximum link delay for attractive links leaving a given node. This strategy is logical for drivers who place a high value on travel time reliability and so need to have good alternatives should links become congested after trip commencement.

As in the original Spiess and Florian (1989) algorithm, a link only becomes attractive if its use reduces the expected trip cost. However, in this case the interpretation of expectation is a little different. Because the driver is risk averse, he takes a pessimistic view, and his expectation of delay at any node is equal to his minmax exposure to delay. If there is only one attractive link leaving a given node, the expected delay is the maximum delay on that link. If there are alternative links, his exposure to delay is reduced, because he has escape possibilities. As delays are possible on the escape links as well, he adopts the minmax exposure to delay strategy.

As mentioned in the introduction, algorithm speed is critically important in vehicle navigation systems. Hence, node potentials have been introduced to speed up the hyperpath search, as in the Astar algorithm. Significant reductions in computation are demonstrated by numerical example. A proof of optimality is provided for the use of potentials in this context.

Acknowledgements

Fumitaka Kurauchi, Klaus Knoeckel and Kay Axhausen are gratefully acknowledged for comments on an early draft. The author is deeply indebted to reviewer one whose insightful comments led to a significant improvement of the paper.

References

Akgun, V., Erkut, E., Batta, R., 2000. On finding dissimilar paths. European Journal of Operational Research 121 (2), 232–246.
Azevedo, J.A., Santos Costa, M.E.O., Silvestre Madera, J.J.E.R., Vieira Martins, E.Q., 1993. An algorithm for the ranking of shortest paths. European Journal of Operational Research 69 (1), 97–106.
Barra, T., (1993). Multidimensional path search and assignment. In: Proceedings of the 21st PTRC Summer Annual Conference, pp. 167–172.
Bauer, R., Delling, D., Wagner, D., (2007). Experimental study on speed-up techniques for timetable information systems. ARRIVAL-TR-91 (<http://arrival.cti.gr>/).
Bekhor, S., Ben-Akiva, M.E., Ramming, S., (2001). Route choice: choice set generation and probabilistic choice models. In: Proceedings of the 4th TRISTAN Conference, Azores, Portugal.
Ben-Akiva, M., Bergman, M.J., Daly, A.J., Ramaswamy, R., 1984. Modeling inter-urban route choice behaviour. Proceedings of the Ninth International Symposium on Transportation and Traffic Theory. VNU Science Press, Utrecht. pp. 299–330.
Chen, Y.Y., Bell, M.G.H., Bogenberger, K., 2007. Reliable pretrip multipath planning and dynamic adaptation for a centralised road navigation system. IEEE Transactions on Intelligent Transportation Systems 8 (1), 14–20.
Dial, R.B., 2000. Bicriterion traffic equilibrium: T2 mode, algorithm, software overview. Transportation Research Record 1725, 54–62.
Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1 (1), 269–271.
Dinic, E.A., 1970. Algorithm for solution of a problem of maximum flow in a network with power estimation. Soviet Mathematics Doklady 11, 248–264.

Eppstein, D., 1998. Finding the *k*-shortest paths. SIAM Journal of Computing 28 (2), 652–673.
Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics SSC4 (2), 100–107.
Jimenez, V.M., Marzal, A., 1999. Computing the *k*-shortest paths: a new algorithm and an experimental comparison. Lecture Notes in Computer Science Series, vol. 1688. Springer-Verlag. pp. 15–29.
Martins, E.Q.V., Pascoal, M.M.B., Santos, J.L.E., 1999. Deviation algorithms for ranking shortest paths. International Journal of Foundations of Computer Science 10 (3), 247–261.
Nguyen, S., Pallotino, S., 1988. Equilibrium traffic assignment for large scale transit networks. European Journal of Operational Research 37 (2), 176–186.
Park, D., Rilett, L.R., 1997. Identifying multiple and reasonable paths in transportation networks: a heuristic approach. Transportation Research Record 1607, 31–37.
Park, K., Bell, M.G.H., Kaparias, I., Bogenberger, K., 2007. Learning user preferences of route choice behaviour for adaptive route guidance. IET Intelligent Transport Systems 1 (2), 159–166.
Prato, C.G., Bekhor, S., 2006. Applying branch-and-bound technique to route choice set generation. Transportation Research Record 1985, 19–28.
Pu, J., Manning, E.G., Shoja, G.C., Srinivasan, A. (2001). A new algorithm to compute alternate paths in reliable OSPF (ROSPF). In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2001), 1, pp. 299–304.
Rouphail, N.M., (1995). A decision support system for dynamic pre-trip route planning. In: Proceedings of the 4th ASCE International Conference on Application of Advanced Technologies in Transportation (AATT), New York, pp. 325–329.
Scott, K., Pabon-Jimenez, G., Bernstein, D., (1997). Finding alternatives to the best path. In: Proceedings of the 76th Annual Meeting of the Transportation Research Board, Washington, DC.
Sheffi, Y., Powell, W.B., 1982. An algorithm for the equilibrium assignment problem with random link times. Networks 12 (2), 191–207.
Spiess, H., Florian, M., 1989. Optimal strategies: a new assignment model for transit networks. Transportation Research Part B 23 (2), 83–102.
Torrieri, D., 1992. Algorithms for finding an optimal set of short disjoint paths in a communication network. IEEE Transactions on Communications 40 (11), 1698–1702.
Van der Zijpp, N.J., Fiorenzo-Catalano, S., 2005. Path enumeration by finding the constrained *k*-shortest paths. Transportation Research Part B 39 (6), 545–563.
Wagner, D., Willhalm, T., (2006). Speed-up techniques for shortest path computations. ARRIVAL-TR-0024 (<http://arrival.cti.gr/>).