TEMPORALLY ADAPTIVE A* ALGORITHM ON TIME-DEPENDENT TRANSPORTATION NETWORK

N. B. Zheng *, F. Lu

Institute of Geographic Sciences and Natural Resources Research, CAS, 11A Datun Road, Chaoyang District, Beijing, 100101, China – (zhengnb, luf)@lreis.ac.cn

KEY WORDS: Shortest Path; A* algorithm; Time-Dependent Network; FIFO condition; Traffic Information

ABSTRACT: Traditional solutions to shortest path problems on time-varying transportation networks only use traffic information at definite moment so as to ignore the fact that the travel time through a link is dependent on the time to enter it. In this paper, the travel speed instead of the travel time on each link of road networks was modelled as a time-interval dependent variable, and a FIFO-satisfied computational function of the link travel time was then deduced. At last, a temporally adaptive A* shortest path algorithm on this FIFO network was presented, where the time factor was introduced into the evaluation function, and the Euclidean distance divided by the maximum possible travel speed was used as heuristic evaluator. An experiment on the real road network shows that the proposed algorithm is capable of foreseeing and bypassing those forthcoming traffic congestions, only with a cost of about 10 percent more computational time than the traditional algorithm. Furthermore, frequent path reoptimization caused by the traditional algorithm gets avoided effectively.

1. INTRODUCTION

It is increasingly necessary for a vehicle navigation system or a map search website to calculate fastest paths by using real-time, historical, or predicted traffic information (Fawcett & Robinson, 2000; Yamane *et al*., 2004; Ishikawa, 2005). Traditional solutions to this problem only consider traffic information at definite moment, and calculate optimal paths by either classic or heuristic shortest path algorithms including Dijkstra, A*, branch pruning, hierarchical search, etc (Lu & Guan, 2004; Fu *et al.*, 2006; Klunder *et al.*, 2006; Cho & Lan, 2009). As a result, while travelling, the planned path will have to be re-optimized frequently to respond the periodical renewal of that near-real-time traffic information (e.g., every five minutes). Obviously, this process will be considerably time-consuming, even if some practical accelerating techniques, such as dynamic window scheme (Kim & Jung, 2002) and incremental search approach (Huang & Wu, 2007), are introduced into it. Besides, this type of algorithms is incapable of computing the path and evaluating the travel time for a trip from an overall perspective. The reason is that they have overlooked the fact that the travel time through a link is dependent on the time to enter it. Therefore, the time-dependent shortest path algorithms are needed for the dynamic route planning.

The time-dependent shortest path problem (TDSPP) models the travel time through a link as an arrival-time-dependent variable, and can be resolved by some modified labelling algorithms (i.e., Dijkstra, A*, etc), only if the First-In-First-Out (FIFO) premise is satisfied (Kaufman & Smith, 1993; Sung *et al.*, 2000; Horn, 2000; Chabini & Lan, 2002; Kanoulas *et al*., 2006). In practice, the travel time through a link is usually taken as time-interval dependent in a nutshell, which ignores the fact that the travel speed may vary with time intervals during a trip along this link, and at the same time, violates the FIFO condition. In view of this, this paper defines the travel speed rather than the travel time on a link as a time-interval dependent variable, and then computes the FIFO-satisfied time-dependent link travel times, which will be used for the temporal adaption of the A* shortest path algorithm at last.

The remains of this paper are organized as follows. Section 2 defines a time-dependent network with time-interval dependent link travel speeds. Section 3 derives a computational function of FIFO-satisfied link travel time. Section 4 presents a temporally adaptive A* algorithm. Section 5 tests the proposed algorithm on a real road network. Conclusions are drawn in section 6.

2. TIME-DEPENDENT NETWORK

Suppose the time T is segmented into the following intervals: $[t_0, t_1), [t_1, t_2), ..., [t_k, t_{k+1})$ ($k = 0, 1, ..., m–1$, $t_k < t_{k+1}$), then the time-dependent network is defined as $G = (N, A, L, V, W)$, where $N = \{0, 1, ..., n–1\}$ denotes the node set; $A \subseteq \{(i, j)|(i, j) \in N \times N\}$ denotes the directed link set; $L = \{l_{ij}|(i, j) \in A, l_{ij} > 0\}$ denotes the link length set; $V = \{f_{ij}(t)|(i, j) \in A, t \in T\}$ denotes the set of the time-interval dependent link travel speeds, and for each $t \in [t_k, t_{k+1})$, have $f_{ij}(t) = v_{(i,j)k}$; $W = \{w_{ij}(t)|(i, j) \in A, t \in T\}$ denotes the set of the time-dependent link travel times, where $w_{ij}(t)$ denotes the travel time along link (i, j) when departing from node i at time t.

Let $T_{ij}(t)$ denote the arrival time to node j with departure time t from node i along link (i, j), have $T_{ij}(t) = t + w_{ij}(t)$. Let $p = \{i_1, i_2, ..., i_u\}$ denote a path from node i_1 to node i_u, then the path arrival time function of path p is given by the composition of the link arrival time functions along p:

$$T_p(t) = T_{i_{u-1}i_u}(T_{i_{u-2}i_{u-1}}(T_{i_{u-2}i_{u-3}} \cdots \ T_{i_1i_2}(t))) \ .$$

Let $P(o, d)$ denote the path set from origin node o to destination node d, $ET_{od}(t)$ denote the earliest arrival time while leaving from origin node o at time t to destination node d, then:

$$ET_{od}(t) = \min\{T_p(t) : p \in P(o,d)\} \ .$$

* Corresponding author. Email: zhengnb@lreis.ac.cn.

Obviously, the time-dependent shortest path problem is just a problem of computating the earliest arrival time $ET_{od}(t)$.

(FIFO conditon) For each link $(i, j) \in A$, if the following inequality is satisfied, we call this link FIFO-satisfied:

$$T_{ij}(s) \le T_{ij}(t) \quad \forall s \le t .$$

If every link of a time-dependent network is FIFO satisfied, we call this network a FIFO network. Kaufman & Smith (1993) has proved: the shortest path problem on the FIFO network can be well dealt with by the traditional labelling algorithms.

3. COMPUTATION OF LINK TRAVEL TIME

Consider a link $(i, j) \in A$ of length l (for simplicity, we drop the subscript (i, j) in this section) with travel speed v_k changing with the time intervals $[t_k, t_{k+1})$ ($k = 0, 1, ..., m–1$). If a vehicle sets off from node i at time t and reaches node j at time $T(t)$, then its travel time along this link will be $T(t) - t$. The computational procedure of $T(t)$ is as follow:

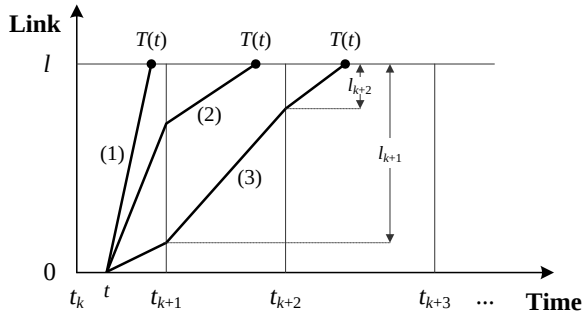


Fig. 1 A trip along a link with time-interval-varying speeds

1) if $l_k - v_k \times (t_{k+1} - t) < 0$ (See Fig. 1(1)), then
$T(t) = t + l_k / v_k$,
else
2) if $l_{k+1} - v_{k+1} \times (t_{k+2} - t_{k+1}) < 0$ (See Fig. 1(2)), then
$T(t) = t_{k+1} + l_{k+1} / v_{k+1}$,
else
3) if $l_{k+2} - v_{k+2} \times (t_{k+3} - t_{k+2}) < 0$ (See Fig. 1(3)), then
$T(t) = t_{k+2} + l_{k+2} / v_{k+2}$,
else
⋮
i) if $l_{k+i-1} - v_{k+i-1} \times (t_{k+i} - t_{k+i-1}) < 0$, then
$T(t) = t_{k+i-1} + l_{k+i-1} / v_{k+i-1} \qquad \forall i = 2,3...$,
else
⋮

where

$$l_k = l$$
$$l_{k+1} = l_k - v_k \times (t_{k+1} - t)$$
$$l_{k+2} = l_{k+1} - v_{k+1} \times (t_{k+2} - t_{k+1})$$
$$\vdots$$
$$l_{k+i} = l_{k+i-1} - v_{k+i-1} \times (t_{k+i} - t_{k+i-1})$$
$$\vdots$$

In practice, if the time intervals are long enough, a travel along the link will cover no more than two time intervals, namely, $[t_k, t_{k+1})$, $[t_{k+1}, t_{k+2})$. In this case, the above computational procedure can be reduced to:

$$T(t) = \begin{cases} t + l/v_k , & t \in [t_k,\ t_{k+1} - l/v_k) \\ t_{k+1} + [l - v_k(t_{k+1} - t)]/v_{k+1} , & t \in [t_{k+1} - l/v_k,\ t_{k+1}) \end{cases} .$$

The FIFO satisfaction of $T(t)$ is illustrated by Fig. 2 and Fig. 3. Assume $T(t)$ violates the FIFO condition, that is, an overtaking happens: $T(s) > T(t)$ $\forall s \le t$, two trajectories are sure to intersect at some position, e.g., point C in Fig. 2. Thus, there will have two different speeds in the same time interval, e.g., $[t_{k+2}, t_{k+3})$ in Fig. 2. In following two figures, speed value is indicated by the slope of the trajectory. Apparently, this violates the definition of the time-dependent network in Section 2. Consequently, it can be surely concluded that $T(t)$ satisfies the FIFO condition. In fact, two travel trajectories departing at different times will be parallel in the FIFO network, as shown in Fig. 3.

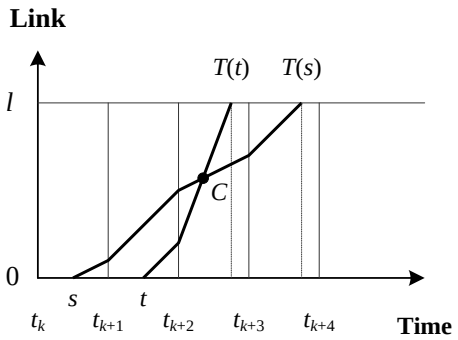


Fig. 2 FIFO condition violated

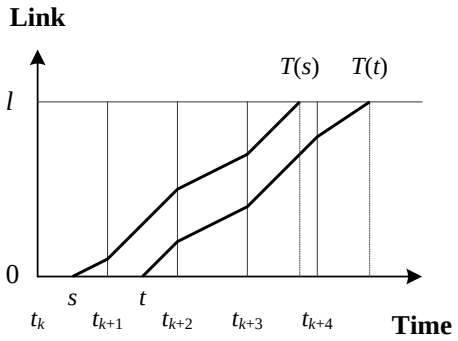


Fig. 3 FIFO condition satisfied

4. TEMPORALLY ADAPTIVE A* ALGORITHM

The A* shortest path algorithm was first proposed by Hart *et al*. (1968), and further proved applicable to road networks by Fu *et al*. (2006) and Zeng & Church (2009). The A* algorithm makes use of a heuristic evaluation function $F_i = L_i + e_{(i,d)}$ as a label for node i, where L_i is the travel time of the current evaluated path from the origin node to node i; $e_{(i,d)}$ is an estimated travel time from node i to node d. The sum of these two functions, F_i, is the weight of node i, and reflects the likelihood of node i being on the shortest path.

As for the time-dependent network, it is necessary to introduce time factor into the evaluation function: $F_i(t) = T_i(t) + e_{(i,d)}(T_i(t))$, where t denotes the departure time from the origin node; $T_i(t)$ denotes the arrival time of the current path from the origin node at the time t to the node i; $e_{(i,d)}(T_i(t))$ is an evaluated travel time from the node i at time $T_i(t)$ to the destination node d. $e_{(i,d)}(T_i(t))$ controls the accuracy and efficiency of the algorithm. Chabini & Lan (2002) has proved that if $e_{(i,d)}(T_i(t))$ is not more than the real minimum travel time from node i to the destination node d with departure time t, the temporally adaptive A* algorithm will be strictly optimal. In view of this, let

$$e_{(i,d)}(T_i(t)) = \frac{D_{(i,d)}}{V_{\max}},$$

where $D_{(i,d)}$ denotes the Euclidean distance from node i to the destination node d; $V_{\max}$ denotes the maximum possible travel speed.

Let o denotes the origin node, d denotes the destination node, t denotes the departure time, P_i denotes the straight preceding node of node i along the shortest path, Q denotes the scan eligible node set, and R denotes the permanent labelled node set (That is, the shortest path from the origin node to any node in this set has been found), then the basic steps of the temporally adaptive A* (TAA*) algorithm are as follows:

Step 1: Initialization: Set $T_o = t$; $F_o = T_o + e_{(o,d)}(T_o)$; $F_j = T_j = \infty \ \forall j \neq o$; $P_o = -1$; $Q = \{o\}$; $R = \emptyset$;
Step 2: Node Selection: Select and remove the node i with the minimum label $F_i(t)$ from the scan eligible node set Q, and label it permanently:
$i = \text{argmin}_{j \in Q}\{F_j\}$; $Q = Q \backslash \{i\}$; $R = R \cup \{i\}$;
if $i == d$, the shortest path has been found, then
STOP;
else
Step 3: Node Expansion: Scan each link outgoing from node i.
For each link $(i, j) \in A$ & $j \notin R$, if
Arri_time((i, j), T_i) + $e_{(j,d)}$(Arri_time((i, j), T_i)) < F_j,
then
T_j = Arri_time((i, j), T_i); $F_j = T_j + e_{(j,d)}(T_j)$; $P_j = i$;
insert node j into Q: $Q = Q \cup \{j\}$;
where Arri_time((i, j), T_i) is a procedure of computing the arrival time of link (i, j) according to Section 3;
Step 4: Termination: If $Q = \emptyset$, then STOP;
else: goto step 2.

Tab.1 Default speed values of different road types (Speed: km/h)

	Expressway	Arterial	Sub-arterial	Minor	Overpass	Ramp
Monday morning peak (7:00-9:00) & Friday evening peak (17:00-19:00)	35	35	35	28	35	30
Other peaks (7:00-9:00; 17:00-19:00)	45	40	45	40	30	30
Nights (21:00-24:00; 0:00-6:00)	60	60	60	50	45	45
Others (6:00-7:00; 9:00-17:00; 19:00-21:00)	55	55	55	50	55	45

Tab 2 A naive model for time-dependent turn delays (Time: min)

	Straight	Right	Left	U-turn	Others
Peaks (7:00-9:00; 17:00-19:00)	1.0	0.5	1.0	0.8	0.3
Night (21:00-24:00; 0:00-6:00)	0.5	0.0	1.0	0.5	0.5
Others (6:00-7:00; 9:00-17:00; 19:00-21:00)	0.5	0.1	1.0	0.5	1.0

5. EXPERIMENT

This paper implements and tests the proposed algorithm in the self-developed Urban Public Travel Path Service System with runtime environment: dual-core CPU 1.6GHz, RAM 1.0GB, OS Windows XP professional. The experimental data include road network data within the Beijing's Fourth Ring and carriageway-based floating car traffic data of July, August, and September, 2007. As for the algorithm, the scan eligible node set is realized by quad-heap priority queue (Lu *et al*., 1999); the maximum possible travel speed is set to 60km/h. Besides, in order to deal with turn delays and prohibitions in the transportation network, an arc-labelling strategy is introduced (Gao & Lu, 2008). To be specific, the node and the link exchange their roles each other and the turn delays are accumulated into the path arrival time.

5.1 Data Preparation

The topology of the road network is carriageway-based, and one carriageway corresponds to one link of the time-dependent network in Section 2. After generalizing those virtual links and virtual nodes that represent a same road intersection into one topological node, the network contains 53997 links in all.

The traffic data of one day are discreted into 288 time intervals, and each interval is 5 minute duration. The traffic data items in each time interval include: carriageway ID, carriageway length, travel time, and congestion level. In order to support the TAA* algorithm, we convert these raw travel time data into the travel speed data by the division of carriageway length by travel time. Furthermore, in view of the cycling nature of the urban traffic flow, we average those speed values with the same day category and the same time interval, which results in 7 speed data files.

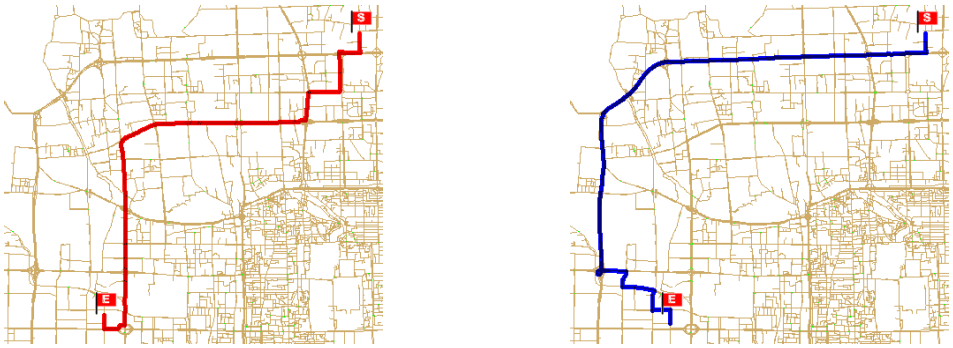
For the carriageways not covered by floating car traffic data, we give them default speed values according to the road type and the entrance time, as shown in Tab. 1. Moreover, we build a naive time-dependent turn delay model: $Turn = 0.5 \times f(time,$

turn_type), where 0.5 denotes the probability of waiting at intersections; *f*(*time*, *turn_type*) is evaluated as Tab. 2.

5.2 Experimental Results

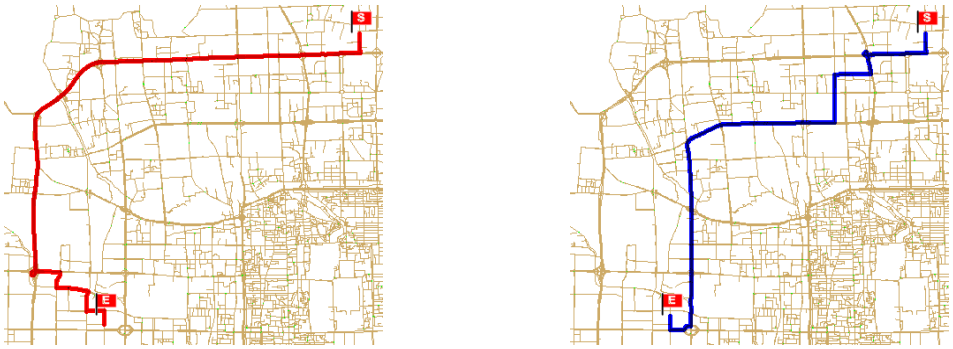
We choose the Beichendong Road as an origin road and the Cuiwei Road as an destination road, set 7:00 and 8:00 on Tuesday as two departure times, and then compute four least-time paths by the TAA* algorithm and the real-time A* (RTA*) algorithm respectively. Computational results are shown in Fig. 4 and Fig. 5. As compared with the TAA* algorithm, the only and significant distinct of the RTA* algorithm is that the RTA* algorithm merely uses the traffic data at the departure time.

As seen from the Fig. 4 and Fig. 5, while departing at 7:00, the TAA* algorithm foresees and bypasses the forthcoming traffic congestion on the Third Ring Road, but the RTA* algorithm do not. While departing at 8:00, the TAA* algorithm predicts that the current traffic congestion on the Third Ring Road will die away at the entrance time, and therefore avoids an unwanted detour caused by the RTA* algorithm.



(a) RTA* (b) TAA*

Fig. 4 Comparison of computational results of TAA* and RTA* (Departure time: 7:00)



(a) RTA* (b) TAA*

Fig. 5 Comparison of computational results of TAA* and RTA* (Departure time: 8:00)

Furthermore, we select 30 pairs of Origin-Destination (O-D) carriageways, set 7:30 on Tuesday as departure time, and then calculate the least time path between each O-D pair by the TAA* algorithm, RTA* algorithm and RTA*_M algorithm separately. The comparisons between them are shown in Fig. 6, Fig. 7 and Fig. 8 (The 30 paths are sorted by length). Thereinto, the RTA*_M algorithm works as follow: i) At the beginning of the trip, compute an optimal path by the RTA* algorithm; ii) during the trip, re-optimize the path by the RTA* algorithm as long as new traffic information arrives (This is simulated by the historical traffic data); iii) repeat step ii) until the destination is reached. So the computational time of the RTA*_M algorithm is the sum of that of each RTA* algorithm, and the path of the RTA*_M algorithm is that actual travelled path.

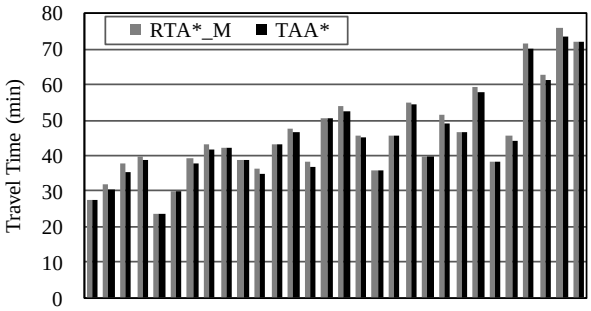


Fig. 6 Comparison of path travel times of TAA* and RTA*_M

As seen from the Fig. 6, although the low traffic information coverage and the immature of the traffic prediction model and the turn delay model make a few paths less differentiated, the path travel times of the TAA* algorithm are less than those of the RTA*_M algorithm on the whole.

It is concluded from the Fig. 7 that the TAA* algorithm will cost about 10 percent more computational time than the RTA* algorithm. The extra time is exhausted mainly by the procedure of link travel time computation. Its time complexity is O(*m*), where *m* denotes the number of the time intervals of the link.

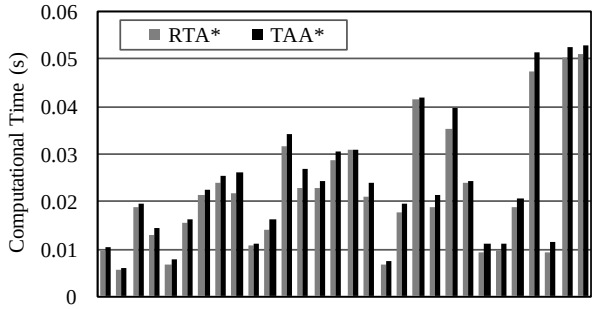


Fig. 7 Comparison of computational times of TAA* and RTA*

It can seen from the Fig. 8 that the TAA* algorithm cost much less computational time than the RTA*_M algorithm. The reason is that the TAA* algorithm only need to carry out the computational procedure once because it has taken the future traffic status into account already in advance, but comparatively, the RTA*_M algorithm must execute the computational procedure again and again because of its "short views".

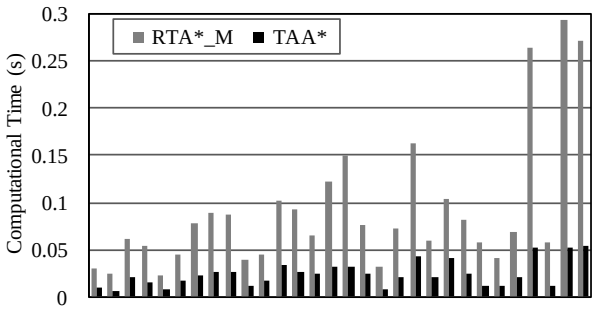


Fig. 8 Comparison of computational times of TAA* and RTA*_M

6. CONCLUSIONS

This paper suggests a novel time-dependent network model, where the travel speed instead of the travel time is regarded as changing with different time intervals, and from which a FIFO-satisfied computational function of link travel time is derived. Furthermore, this paper develops a temporally adaptive A* algorithm to calculate least-time paths on the defined time-dependent network, where the link-travel-time computational function was used to the evaluation function, and the Euclidean distance divided by the maximum possible travel speed was designed as the heuristic evaluator. An experiment on the real road network shows that the proposed algorithm is capable of foreseeing and bypassing those forthcoming traffic congestions, saving the travel time, and raising the overall efficiency of the navigation system.

The implementation of the TAA* algorithm needs the support of link travel speed data and turn delay data. This paper only takes the simple means of historical link travel speed data as future link travel speeds. Besides, turn delays between links are set empirically rather than theoretically. Therefore, we will keep on our research work on accurate and available link travel time prediction method and turn delay measure model in the near future.

Acknowledgements:
The work described in this paper is supported by the National Natural Science Foundation of China (40871184), the China Postdoctoral Science Foundation (20090450563) and the National High Technology Research and Develop Program of China (2007AA12Z241).

References:

Chabini, I., Lan, S., 2002. Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks. *IEEE Transactions on Intelligent Transportation Systems*, 3(1), pp. 60-74.

Cho, H.J., Lan, C.L., 2009. Hybrid shortest path algorithm for vehicle navigation. *The Journal of Supercomputing*, 49(2), pp. 234-247.

Fawcett, J., Robinson, P., 2000. Adaptive routing for road traffic. *IEEE Computer Graphics and Applications*, 20(3), pp. 46-53.

Fu, L., Sun, D., Rilett, L.R., 2006. Heuristic shortest path algorithms for transportation applications: state of the art. *Computers & Operation Research*, 33(11), pp: 3324-3343.

Gao, S., Lu, F., 2008. An arc-labeling shortest time path algorithm. *Geo-Information Science*, 10(5), pp. 604-610. (In Chinese)

Hart, E.P., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transaction of System Science and Cybernetics*, SSC-4(2), pp. 100-107.

Horn, M., 2000. Efficient modelling of travel in networks with time-varying link speeds. *Networks*, 36(2), pp. 80-90.

Huang, B., Wu, Q., Zhan, F.B., 2007. A shortest path algorithm with novel heuristics for dynamic transportation networks. *International Journal of Geographical Information Science*, 21(6), pp. 625-644.

Ishikawa, H., 2005. Development of a local storage type traffic prediction for car navigation system. In: *The 12th World Congress on Intelligent Transportation System*, San Francisco, USA.

Lu, F., Guan, Y., 2004. An optimum vehicular path solution with multi-heuristics. M. Bubak *et al.* (Eds.): ICCS 2004, *Lecture Notes in Computer Science*, 3039, pp. 964-971.

Lu, F., Lu, D., Cui, W., 1999. Improved Dijkstra algorithm based on quad-heap priority queue and inverse adjacent list. *Journal of Image and Graphics*, 4A(12), pp. 1044-1050. (In Chinese)

Kanoulas, E., Du Y., Xia, T., Zhang, D., 2006. Finding fastest paths on a road network with speed patterns. In: The 22nd International Conference on Data Engineering, Atlanta, USA.

Kaufman, D.E., Smith, R.L., 1993. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. *IVHS Journal*, 1, pp. 1-11.

Kim, J., Jung, S., 2002. A dynamic window-based approximate shortest path re-computation method for digital road map databases in mobile environments. M.H. Shafazand and A.M. Tjoa (Eds.): EurAsia-ICT 2002, *Lecture Notes in Computer Science*, 2510, pp. 711-720.

Klunder, G.A., Post, H.N., 2006. The shortest path problem on large-scale real-road networks. *Networks*, 48(4), pp. 182-194.

Sung, K., Bell, M.G.H, Seong, M., Park, S., 2000. Shortest paths in a network with time-dependent flow speeds. *European Journal of Operational Research*, 121(1), pp. 32-39.

Yamane, K., Endo, Y., Fujiwara, J., Machii, K., 2004. Statistical traffic information for navigation system. In: The *11th World Congress on Intelligent Transportation System*, Nagoya, Aichi, Japan.

Zeng, W., Church, R.L., 2009. Finding shortest paths on real road networks: the case for A*. *International Journal of Geographical Information Science*, 23(4), pp. 531-543.