



**INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH
TECHNOLOGY**

**TIME-DEPENDENT A* ROUTE PLANNING SYSTEM--AN ONLINE APPROACH FOR
TAIWAN FREEWAY SYSTEM**

Tang-Hsien Chang*, Yuan-Hsiang Yeh, Bor-Chia Hsieh, Jen-Sung Tseng

* Department of Civil Engineering, National Taiwan University, Taiwan

ABSTRACT

Traditional route guidance programs often rely on spatial distance as the decision variable for their route calculations and they do not consider the actual traffic conditions. This paper used the real-time travel time of vehicles passing through road segments as the cost in route selection, to develop a time-dependent A* Algorithm for routing. Real-time data from a travel-time database was used as the data source for journey calculation. The travel-time database incorporated from Vehicle Detectors (VD) and Electronic Toll Collection (ETC) systems. Aside from the forward-search calculation method, backward-search capability was also added to cater for the needs of different users. The user would enter the starting point, destination and the expected departure or arrival time. The system would then combine the road segment travel data from the travel-time database for calculation, recommend the best route and provide the total time for the journey. Finally this research showed that the A* Algorithm could effectively and rapidly narrow the direction of search and produce a satisfactory recommended route.

KEYWORDS: Time-dependent shortest route, Dijkstra's Algorithm, A* Algorithm, Route guidance system

INTRODUCTION

The rapid development of transportation systems today has given us a multitude of options to reach our destinations. Choosing from the many alternatives available becomes an additional dilemma. There are many route guidance programs on the market. Most programs based their decisions on the shortest spatial distances whether they are intended for pre-trip or in-transit route planning. In reality the best routes recommended by these programs are often found to be congested. Much time is wasted unnecessarily waiting in congestion, resulting in grief and aggravation before the user finally reaches their destination.

When traveling long distances, people are more sensitive to the perception of time rather than distance. To the majority of users, the length of time spent on a journey is more important than the distance traveled. Therefore we believe it is necessary to have a route decision system that uses the temporal cost of traveling as its design basis. When compared with spatial cost which is fixed, temporal cost varies with time and is more difficult to predict. As the journey time increases, the travel time forecast for individual segments can cumulate to large deviations in total journey time. This considerably affects the accuracy of the route decisions. Therefore, it is necessary to have a reliable travel-time database that can provide accurate real-time data on travel segments in order to support a level of accuracy and consistency when making route decisions.

Journey planning affects many different people. There are the workers, students and commuters who rely on, or need to connect to other public transportations such as planes, trains and high-speed rail. Travelers are often under time pressure or must follow a rigid transportation time table. If the journey planning begins with the time of departure, the user would need to trial several different departure times to find the most appropriate time to leave. If one arrives early at one location, there is no efficient mean to utilize the additional waiting time for the next segment of the journey. Not to mention if one arrives late, it could result in financial losses at the place of work or due to missing subsequent connections to their destination. If the user wants to find a route based on an expected arrival time, it can be done by using the backward-search calculation method. This method provides the user with the latest time to depart. This allows drivers to effectively: use the time before the journey, reduce the traveling time and minimize the waiting time after arrival.

Dijkstra's Algorithm is the main theoretical method for finding the shortest path between two nodes [1][2]. The algorithm is applicable on network segments where costs are non-negatively weighted. Although Dijkstra's Algorithm is capable of finding the shortest route without having to traverse every node, the algorithm could also waste a lot of time searching nodes that are clearly not suitable for online APP on modern smart phones or on-board electronic devices. A* Algorithm can effectively limit the area of exploration by applying the search direction in finding a path towards the destination point. This paper built the most optimal route decision system using the A* Algorithm that was suitable to be used on Taiwan's national highways. In addition, it aims to provide drivers with appropriate service software to plan their journeys.

DIJKSTRA'S ALGORITHM

At present time most problems that involve finding the shortest distance between two nodes employ Dijkstra's Algorithm as the main theoretical method [1][2]. The computing time complexity is $O(n^2)$. It requires the cost of each network segment to be non-negative. The calculation process for Dijkstra's Algorithm is as shown in Figure 1. Although Dijkstra's Algorithm can obtain the most optimal solution, when it is applied on a network that is huge and complex, the algorithm often wastes a lot of its time searching in unnecessary directions. For example, if the destination point is located south-east of the starting point, Dijkstra's Algorithm would begin searching from the starting point in all directions for all possible paths, including the north-west and other directions. However the advantage of Dijkstra's Algorithm is that it does not need to traverse all nodes in order to find the shortest path. If the shortest path to the destination point has been found, other paths that also reach the destination point must have a travel time longer than the shortest path. Any sub-path of this shortest path must also be the shortest path.

A* ALGORITHM

A* Algorithm [3][4] is the mainstream technology for solving the shortest path problem in gaming software due to its heuristic design potential. It also eliminates many of the paths that are obviously unsuitable by using a set of special heuristic estimate formulas. The aim of the algorithm is to narrow the direction for searching for nodes by reducing the number of unnecessary nodes to be explored, thereby reducing the overall search time. The search depth of A* Algorithm does not output the most optimal solution produced from exploring the entire map by default. Instead the algorithm selects the appropriate cost function design to yield the second-best solution which is more efficient. Theoretically A* Algorithm is an extension and an improvement of Dijkstra's Algorithm.

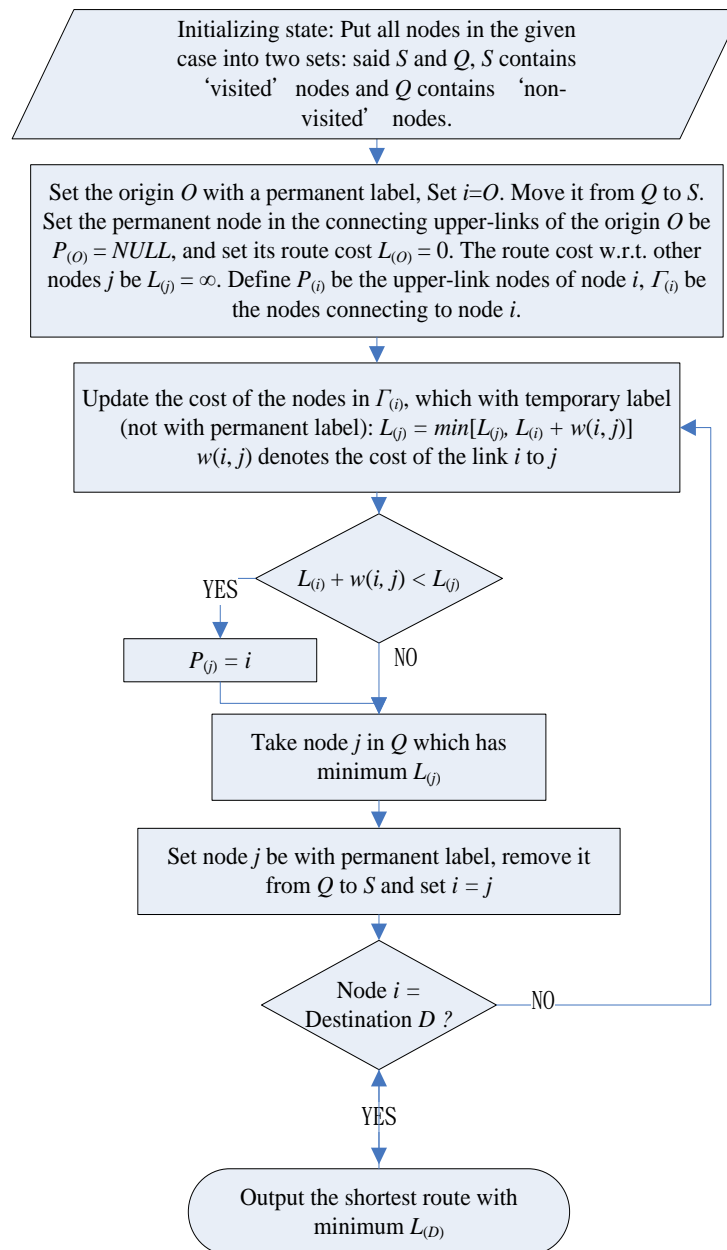


Figure 1. Dijkstra's Algorithm

A* Algorithm has a processing with evaluation as Formula (1):

$$f(n) = g(n) + h(n) \quad (1)$$

$g(n)$: Distance from starting point to the current node n

$h(n)$: Predicted distance from current node n to the destination point

$f(n)$: Evaluation score of the current node n

Where $h(n)$ can have the following scenarios:

1. $h(n) = 0$, which is equivalent to Dijkstra's Algorithm. The most optimal solution will be the target of the search.
2. $h(n) < D_{cd}$, A* Algorithm will search for the most optimal solution. Where D_{cd} denotes the distance from current node to the destination. The smaller the $h(n)$ is, the deeper the depth of search will be.
3. $h(n) = D_{cd}$, A* Algorithm will search for the second-best solution and can quickly produce the result.
4. $h(n) > D_{cd}$, no guarantee the shortest path can be found.

This paper aimed to build a time-dependent route calculation system that meets the users' needs. In doing so, this study referenced the A* Algorithm formula above and designed several time-dependent traffic parameters. The conventional use of spatial distance as the cost of travel segments was removed from the algorithm calculation. Instead, travel time through segments was implemented as the cost function. This allowed the calculation to reflect the changes in travel durations at different times of day.

Following the logic of Dijkstra's Algorithm, A* Algorithm would begin at the starting point. It would continuously add to the path nodes that were assigned temporary labels and had the lowest total travel cost. This process would continue until the destination has been reached. As mentioned above, searches in Dijkstra's Algorithm have no directionality, and thus the algorithm is less efficient in terms of speed. **Error! Reference source not found.** When using A* Algorithm, aside from considering the travel cost on each road segment, the straight-line distance between each downstream node and the destination point is also used as a reference. This is done by taking the straight line distance between the starting node and the destination point as the diameter, and the midpoint of the line as the center of a circle. Priority is given to searching nodes that are within this circle. The further away a node is from the center of the circle, the lower it would be in search priority. When searching the downstream nodes, the ability to ensure search priority towards the direction of the destination point would allow finding the shortest reasonable route within a shorter time frame, while at the same time not completely neglecting nodes in other directions. Related parameters are as follows:

1. $g'(n)$: Replacing the original travel distance $g(n)$ by taking into consideration of the cumulative travel time of the downstream nodes.
2. $h'(n)$: The transportation network utilized the Euclidean distance in A* Algorithm [3][4] as a type of evaluation, the heuristic formula for calculating the distance is as follows:

$$h'(n) = h(n) / \bar{V} \quad (2)$$

where, \bar{V} = the upper speed limit; $h(n) = G \times \sqrt{[(node.x - goal.x)^2 + (node.y - goal.y)^2]}$

G = cost of downstream node (distance);

$node.x$ = x coordinate value of the downstream node; $node.y$ = y coordinate value of the downstream node; $goal.x$ = x coordinate value of the destination point; $goal.y$ = y coordinate value of the destination point.

Since the derived heuristic distance $h(n)$ is still tied to spatial cost term, formula (2) is transformed to heuristic temporal cost term by dividing $h(n)$ by the upper speed limit of road segments.

CONSTRUCTION OF THE SYSTEM

The basic structure of the system is divided into three parts: user interface, route calculation module and travel-time database. The user can begin by entering the "starting point", "destination point" and "expected departure/arrival time". The user interface takes the information entered by the user and converts it into standard units (such as starting and destination points are converted into standard mileage, departure/arrival time are converted to standard times. The route calculation module will begin the time-dependent calculation for the shortest route after receiving the query. Since the travel duration on road segments varies over time, the system will continuously query the travel-time database for the road segments' latest travel time for the cost of the route during calculation. On every downstream node added, the system will record the route and the cumulative journey time until the destination has been reached. The system then returns the recorded route and total journey time as recommendation to the user. The system flow is as shown in Figure 2.

There are two parts to the route calculation module: the forward-search calculation procedure and the backward-search calculation procedure. The input data for the forward-search method are: starting point, destination point and expected departure time. The output data are: recommended travel route, arrival time and total journey time. The input data for the backward-search method are: starting point, destination point and expected arrival time. The output data are: recommended travel route, departure time, and total journey time. The following section provides the details on the forward-search and backward-search route calculation procedures.

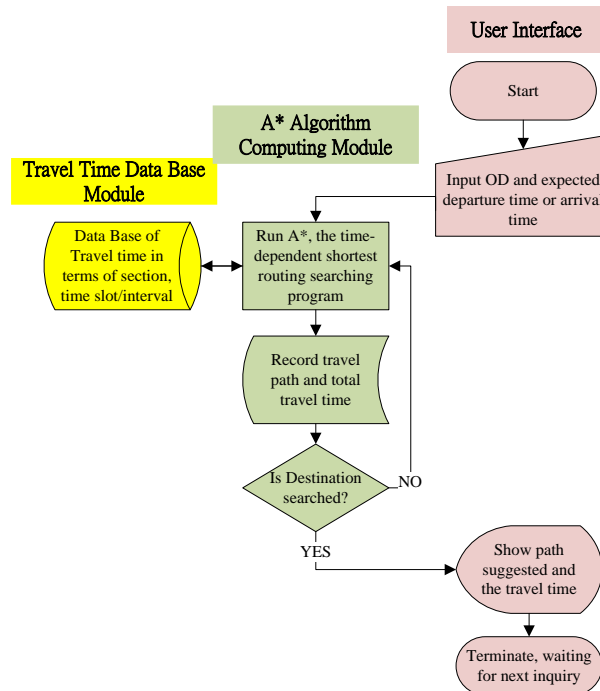


Figure 2. Flow chart of time-dependent route searching system

1. Forward-search route calculation procedure

The forward-search route calculation takes the following data as input: starting point, destination point, and expected departure time. When the route calculation begins, the starting point is given the permanent label. The process would look for all nodes connecting to the permanent label, query the segment travel time for each connecting node, and then assign the node the temporary label. Among these nodes with temporary labels, the process would pick the one node with the best A* score and update it with the permanent label. The above process is repeated until the destination point has been given the permanent label which at that point the calculation would conclude. The steps of the forward-search route calculation are as follows:

Step 1 : Update the following parameters of the starting point: $timeWhenPass = Departure\ Time$; $travelTime = 0$; $parent = null$; $hasVist = true$. Add the starting point into the visited-nodes set S .

Step 2 : Focusing on node i which is the newest member of set S , search for node j , a connecting node to node i with $hasVist = false$. Using node i as the segment's starting point, node j as the destination point and node i 's $timeWhenPass$ as the query time, obtain the latest travel duration for this segment ij from the travel-time database. If the sum of the segment's travel time and i 's $timeWhenPass$ is smaller than j 's $timeWhenPass$, then treat node i as the upstream node of node j and update the following j 's parameters: (1) $timeWhenPass = i$'s $timeWhenPass + ij$'s segment travel time; (2) $travelTime = ij$'s segment travel time; (3) $parent = i$.

Step 3 : Add node j to the shortlisted-nodes set Q .

Step 4 : From set Q , select node k which is the node with the best A* score. Remove node k from set Q and add it to set S .

Step 5 : Verify to see if the destination point has already been added to set S . If destination point has already been added to set S , that means the algorithm has found the destination. The calculation process would conclude at this point. If destination point has not been added to set S yet, return to *Step 2* and resume calculation.

2. Backward-search route calculation procedure

The backward-search route calculation procedure is similar to the forward-search procedure. The backward-search procedure takes the starting point, destination point and expected arrival time as input data. Prior to the calculation, the method simply swaps the starting and destination points entered by the user and proceeds with the calculation process of the forward-search procedure. However, there is one difference from the forward-search procedure when querying the travel costs for the road segments.

Given a simple road segment with the starting point O and destination point D , the method of query to the travel-time database is to enter the starting point O , destination point D and the entry time into the segment at O , t_o . The database would return the travel duration at time t_o for traveling from O to D , $E_{OD}(t)$. When the backward-search method performs its calculation, it swaps the starting and destination points, and thus queries the time duration traveling from D to O passing through D at time t_D . However the direction of the user's journey is from O to D . Even though it may be over the same road segment, the forward and reverse travel times are not necessarily the same. Therefore the travel time duration for the segment for the direction from O to D is required as the segment's travel cost.

In order to get the departure time at point O for the given arrival time at point D , let's assume that over the brief instance of a database query the travel duration would not vary significantly. Using O as the starting point and t_D as the departure time, the database returns the segment's travel duration, $E_{OD}(t_D)$. If the segment's travel duration has not varied over this brief instance of the query, the actual departure time would be $t'_O = t_D - E_{OD}(t_D)$. A test is then needed to verify if the departure time meets the user's requirement. A query is sent to the travel-time database with the starting point O and departure time t'_O . The returned arrival time at point D would be $E_{OD}(t'_O)$. The calculation $\Delta t = E_{OD}(t'_O) - t_D$ would yield the difference between the actual arrival time and the expected arrival time. $\Delta t > 0$ represents the actual arrival time being later than the expected arrival time, whereas $\Delta t < 0$ represents the actual arrival time being earlier than the expected arrival time. If there is a large deviation between the actual arrival time and the expected arrival time, the departure time is adjusted by $t''_O = t'_O - \Delta t$ and it is tested again. This process is repeated until the deviation between actual and expected arrival times is small enough to be acceptable.

SYSTEM IMPLEMENTATION TEST AND RESULTS

This paper used Taiwan's National Freeway No. 1 and National Freeway No. 3 as the scope of the system implementation testing. Freeway No. 1 begins in Keelung City in the North and ends in Kaohsiung City (KAOHSN) in the South with a distance of 372.7 kilometers. Freeway No. 3 begins at Jijin Interchange in the North and reaches Dapeng Bay in the South with a distance of 431.5 kilometers. The routes are as shown in Figure 3. **Figure 3** The data sources for the travel-time database come from the Vehicle Detectors (VD) and Electronic Toll Collection (ETC) systems that are installed on the national highways. The VD system provides vehicle speed data and the ETC system provides the tolling information, particularly about travel time.

Based on the two calculation methods discussed above - the forward-search and the backward-search calculation procedures, the results from both methods were found to be consistent with each other. For the backward-search procedure, the number of searches was found to be higher and the calculation time longer than the forward-search procedure. Although there was only one best route, the search directions were not the same between the forward-search and the backward-search methods. The forward-search calculation began from the starting point whereas the backward-search calculation began from the destination point. The recommended routes were the same even though the number of nodes found and node details were not necessarily matching between the methods.



Figure 3. National Freeway No.1 and 3

To validate that the backward-search calculation could find the same route as recommended by the forward-search calculation, two test cases were designed, each with a starting and destination points combination: (1) a journey passing through one system interchange to another freeway; (2) a journey passing through two system interchanges. The expected arrival time was first entered through the backward-search route calculation method to obtain a departure time. This departure time was then entered into the forward-search route calculation to confirm the output of the backward-search method.

Test Case 1:

- Starting from: Dajia Toll Station at Freeway No. 3
- Arriving at: Dounan Toll Station at Freeway No. 1
- Expected arrival time: 20:00

Results are as shown in Table 1.

Test Case 2:

- Starting from: Longtan Toll Station at Freeway No. 3
- Arriving at: Yunlin Toll Station at Freeway No. 1
- Expected arrival time: 20:00

Results are as shown in Table 2.

Table 1(a). Test case 1-Dijkstra's Algorithm

Forwarding Approach			Backwarding Approach		
Freeway	Nodes	Arrival Time	Freeway	Nodes	Arrival Time
3	Dajia Toll Station	18:58	3	Dajia Toll Station	18:58
3	Dajia	19:03	3	Dajia	19:04
3	Zhonggang Sys	19:07	3	Zhonggang Sys	19:08
3	Shalu	19:12	3	Shalu	19:13
3	Longjing	19:16	3	Longjing	19:17
3	Hemei	19:21	3	Hemei	19:22
3	Changhua Sys	19:24	3	Changhua Sys	19:25
1	Changhua Sys	19:24	1	Changhua Sys	19:25
1	Changhua	19:28	1	Changhua	19:29
1	Puyan System	19:34	1	Puyan System	19:35
1	Yuanlin	19:36	1	Yuanlin	19:37
1	Yuanlin Toll Station	19:40	1	Yuanlin Toll Station	19:41
1	Beidou	19:42	1	Beidou	19:43
1	Xiluo Service Area	19:47	1	Xiluo Service Area	19:48
1	Xiluo	19:48	1	Xiluo	19:49
1	Huwei	19:52	1	Huwei	19:53
1	Dounan	19:55	1	Dounan	19:56
1	Yunlin System	19:57	1	Yunlin System	19:58
1	Dounan Toll Station	19:59	1	Dounan Toll Station	20:00
Accumulated Travel time:		61minutes	Accumulated Travel time:		61minutes
System performing time:		258 ms	System performing time:		693 ms
The number of nodes searched		42 nodes	The number of nodes searched		54 nodes

Table 1(b). Test case 1-A* Algorithm

Forwarding Approach			Backwarding Approach		
Freeway	Nodes	Arrival Time	Freeway	Nodes	Arrival Time
3	Dajia Toll Station	18:58	3	Dajia Toll Station	18:58
3	Dajia	19:03	3	Dajia	19:04
3	Zhonggang System	19:07	3	Zhonggang System	19:08
3	Shalu	19:12	3	Shalu	19:13
3	Longjing	19:16	3	Longjing	19:17
3	Hemei	19:21	3	Hemei	19:22
3	Changhua System	19:24	3	Changhua System	19:25
1	Changhua System	19:24	1	Changhua System	19:25
1	Changhua	19:28	1	Changhua	19:29
1	Puyan System	19:34	1	Puyan System	19:35
1	Yuanlin	19:36	1	Yuanlin	19:37
1	Yuanlin Toll Station	19:40	1	Yuanlin Toll Station	19:41
1	Beidou	19:42	1	Beidou	19:43
1	Xiluo Service Area	19:47	1	Xiluo Service Area	19:48
1	Xiluo	19:48	1	Xiluo	19:49
1	Huwei	19:52	1	Huwei	19:53
1	Dounan	19:55	1	Dounan	19:56
1	Yunlin System	19:57	1	Yunlin System	19:58
1	Dounan Toll Station	19:59	1	Dounan Toll Station	20:00
Accumulated Travel time:		61 minutes	Accumulated Travel time:		61 minutes
System performing time:		134 ms	System performing time:		344 ms
The number of nodes searched		20 nodes	The number of nodes searched		25 nodes

Table 2(a). Test case 2-Dijkstra's Algorithm

Forwarding Approach			Backwarding Approach		
Freeway	Nodes	Arrival Time	Freeway	Nodes	Arrival Time
3	Longtan Toll Station	18:17	3	Longtan Toll Station	18:17
3	Guanxi	18:24	3	Guanxi	18:25
3	Zhulin	18:36	3	Zhulin	18:37
3	Baoshan	18:45	3	Baoshan	18:45
3	Hsinchu System	18:47	3	Hsinchu System	18:48
1	Hsinchu System	18:47	1	Hsinchu System	18:48
1	Toufen	18:54	1	Toufen	18:54
1	Zaoqiao Toll Station	18:58	1	Zaoqiao Toll Station	18:58
1	Miaoli	19:07	1	Miaoli	19:07
1	Tongluo	19:12	1	Tongluo	19:12
1	Sanyi	19:18	1	Sanyi	19:18
1	Taian Service Area	19:24	1	Taian Service Area	19:24
1	Houli	19:25	1	Houli	19:25
1	Houli Toll Station	19:26	1	Houli Toll Station	19:26
1	Taichung System	19:28	1	Taichung System	19:28
1	Fengyuan	19:30	1	Fengyuan	19:30
1	Daya	19:34	1	Daya	19:34
1	Taichung	19:36	1	Taichung	19:36
1	Nantun	19:38	1	Nantun	19:38
1	Wangtian	19:42	1	Wangtian	19:42
1	Changhua System	19:44	1	Changhua System	19:44
1	Changhua	19:47	1	Changhua	19:47
1	Puyan System	19:53	1	Puyan System	19:53
1	Yuanlin	19:55	1	Yuanlin	19:55
1	Yuanlin Toll Station	19:59	1	Yuanlin Toll Station	20:00
Accumulated Travel time:		102 minutes	Accumulated Travel time:		102 minutes
System performing time:		397 ms	System performing time:		1422 ms
The number of nodes searched		66 nodes	The number of nodes searched		97 nodes

Table 2(b). Test case 2-A* Algorithm

Forwarding Approach			Backwarding Approach		
Freeway	Nodes	Arrival Time	Freeway	Nodes	Arrival Time
3	Longtan Toll Station	18:17	3	Longtan Toll Station	18:17
3	Guanxi	18:24	3	Guanxi	18:25
3	Zhulin	18:36	3	Zhulin	18:37
3	Baoshan	18:45	3	Baoshan	18:45
3	Hsinchu System	18:47	3	Hsinchu System	18:48
1	Hsinchu System	18:47	1	Hsinchu System	18:48
1	Toufen	18:54	1	Toufen	18:54
1	Zaoqiao Toll Station	18:58	1	Zaoqiao Toll Station	18:58
1	Miaoli	19:07	1	Miaoli	19:07
1	Tongluo	19:12	1	Tongluo	19:12
1	Sanyi	19:18	1	Sanyi	19:18
1	Taian Service Area	19:24	1	Taian Service Area	19:24
1	Houli	19:25	1	Houli	19:25
1	Houli Toll Station	19:26	1	Houli Toll Station	19:26
1	Taichung System	19:28	1	Taichung System	19:28
1	Fengyuan	19:30	1	Fengyuan	19:30
1	Daya	19:34	1	Daya	19:34
1	Taichung	19:36	1	Taichung	19:36
1	Nantun	19:38	1	Nantun	19:38
1	Wangtian	19:42	1	Wangtian	19:42
1	Changhua System	19:44	1	Changhua System	19:44
1	Changhua	19:47	1	Changhua	19:47
1	Puyan System	19:53	1	Puyan System	19:53
1	Yuanlin	19:55	1	Yuanlin	19:55
1	Yuanlin Toll Station	19:59	1	Yuanlin Toll Station	20:00
Accumulated Travel time:		102 minutes	Accumulated Travel time:		102 minutes
System performing time:		197 ms	System performing time:		786 ms
The number of nodes searched		30 nodes	The number of nodes searched		53 nodes

Examination of the outputs from forward-search and backward-search calculations demonstrated that the solutions from both methods were consistent with each other. This was regardless of whether Dijkstra's Algorithm or A* Algorithm was used. The backward-search method was found to have a higher number of searches and longer calculation time than the forward-search method. Although there was only one best route, the direction of search for the nodes were not the same between the two methods. The forward-search calculation began from the starting point whereas the backward-search calculation began from the destination point. The recommended routes were the same even though the number of nodes found and node details were not necessarily matching between the methods. Furthermore, a comparison was made between the backward-search calculation using A* Algorithm and the forward-search calculation using Dijkstra's Algorithm. Although the former calculation found a lower number of nodes than the latter, the former still took longer to run as compared with the latter due to the fact that when querying the travel time duration of a road segment, the forward-search method could directly use the time when entering the segment to find the travel duration for the segment. In contrast, the backward-search method required repeatedly testing the entry time to a segment for the travel duration in hope of finding the right entry time that would yield the desired exit time leaving the segment. Over the same road segment, the forward-search route calculation would require only one query to the database to obtain the travel time duration for the segment. The backward-search calculation would instead repeat the query several times to get the appropriate travel time. Therefore, in the situation when there is the same number of nodes found, the backward-search method would still require longer calculation time than the forward-search method.

CONCLUSION

This system did not use the spatial distance as travel cost which is commonly adapted elsewhere. Instead, it applied the travel time as the cost for the reason that travelers are more sensitive to the duration of travel rather than to distance. In introducing the travel-time database, the concept of time-dependency was integrated into the route calculation. This allowed the calculations to reflect the ever-changing traffic condition over time and provide more accurate prediction of journey time duration and route guidance.

This paper compared Dijkstra's Algorithm and its extension A* Algorithm. It was found that A* Algorithm could narrow the direction of search for paths, effectively reducing the search time and increasing calculation efficiency. When the network becomes huge and complex, the advantages of A* Algorithm would become even more significant. Although the area of search by A* Algorithm might not be as extensive as that of Dijkstra's Algorithm, A* Algorithm was able to produce satisfactory routes within shorter time frames. A* Algorithm only needed to search through 30% ~ 65% of nodes when compared to Dijkstra's Algorithm to find the same recommended route when implementing the system on a sample road network.

For users who desire to arrive at the chosen destination at a specific time, the backward-search route calculation method, which takes in the starting point, destination and expected arrival time, can meet the users' needs. It saves the users' time from having to repeatedly query using different departure times.

Although the backward-search calculation method can find the same best route for travel as the forward-search method, the backward-search method requires repeated queries with the segment's entry times to obtain an appropriate travel duration for the segment. As a result, the backward-search method requires longer calculation time than the forward-search method even when the same number of nodes is found.





ACKNOWLEDGEMENTS

The research team thanks to Far Eastern Electronic Toll Collection Co, Ltd. as the reaserch sponsor and gave many support.

REFERENCES

- [1] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs." *Numerische Mathematik* 1, pp. 269-271, 1959.
- [2] B.V., Cherkassky, A.V. Goldberg, and T. Radzik, "Shortest paths algorithms: Theory and Experimental Evaluation." *Mathematical Programming* 73(2), pp.129-174, 1996.
- [3] G.A. Klunder and H.N. Post. "The Shortest Path Problem on Large-scale Real-road Networks." *NETWORKS* 48(4), pp.182-194, 2006.
- [4] N. Sturtevant and M. Buro, "Partial Pathfinding Using Map Abstraction and Refinement." *Proceedings of AAAI*, pp. 1392-1397, 2005.

AUTHOR BIBLIOGRAPHY

	<p>Tang-Hsien Chang, He received Ph.D degree from National Taiwan University, Taipei, Taiwan, 1986. He is currently a Professor of Civil Engineering Department at National Taiwan University. His major field is related to Intelligent Transportation Systems, Automatic Vehicle Control, Connected Vehicles, Traffic Measurement and Control, Dynamic Systems, Telematics, Near Field Informatics, Internet of Things. He was the prime chair of ITS Taiwan (1998-2000), executive chair of ITS Asia Pacific (2002-2003) and BOD member of ITS world congress for ten years long (1997-2008). He is now BOD member of TTIA (Taiwan Telematics Industry Association) and the ITS group chair of TIOTA (Taiwan Internet Of Things Alliance). Email:</p>
	<p>Yuan-Hsiang Yeh He is now a Ph.D candidate, National Taiwan University. He is also a professional traffic engineer since 2000.</p>
	<p>Bor-Chia Hsieh He got master degree from National Taiwan. He is now a research assistant in ITS Lab, Professor Chang's team.</p>
	<p>Jen-Sung Tseng He is now a Ph.D candidate, National Taiwan University. He is also the Senior Engineer of Planning Division, Taiwan Area National Expressway Engineering Bureau, Ministry of Transportation and Communications, Taiwan ROC.</p>