

A route navigation system with a new revised shortest path routing algorithm and its performance evaluation

W. Wen & S. W. Hsu

*Department of Information Management,
LungHwa University of Science and Technology, Taiwan, R.O.C.*

Abstract

This paper presents a route navigation system with a new revised shortest path routing algorithm for solving road traffic problems. The system can avoid selecting no left (right) turns, one-way roads, and congested roads when it determines the shortest paths from source to destination. Also, the new revised routing algorithm is compared numerically with existing algorithms such as the Dijkstra algorithm and the A* algorithm. We choose a road network containing 4000 nodes which have 200 no left turn situations and find in particular that the road traffic problem of a 4000-node traffic network can be solved within only 0.651 seconds on average. The prototype system was built and some extra functions were added so that its benefits not only offer the shortest path but also provide information and security services for drivers.

Keywords: shortest path routing algorithm, global position system, route navigation system.

1 Introduction

Traffic congestion has increased steadily in most cities of high-tech countries. To solve congestion problems is not always feasible either by physically constructing new road facilities or by traditional transportation management systems. Therefore, it is important to develop more efficient methods to search an appropriate path on a road network. This is especially useful because no left (right) turns, one-way roads, and congestion roads are frequently encountered in everyday driving situations in big cities. Most road-vehicle cooperative driving systems are able to avoid the one-way roads. However, the systems do not solve



left (right) turn or congestion problems. In other words, the systems may find the shortest route, which includes no left (right) turns, or congestion situations. It means that the shortest path is not always a feasible solution for drivers.

In the past years, many researches focusing on searching the shortest paths for road traffic networks have been published and discussed. Fu and Rilett [5] investigated what they called the dynamic and stochastic shortest path problem by modeling link travel times as a continuous-time stochastic process. Horn [7] continued along the research trails of Chabini [2] used a less detailed articulation of travel dynamics, reflecting as he putted it, the recognition that information about network conditions in most parts of the world are most likely to be sparse and that merely estimates of average speed on individual network links are available in most cases. Ertan [4] built an indoor navigation system that needs ceiling-mounted infrared transmitters and receivers populated in a building. Chabini [1] listed dynamic shortest path problems depending on fastest versus minimum cost (or shortest) path problems. Chen and Yang [3] transformed a traffic-light network into an on-off time-switch network. For finding the shortest path, their study minimized total travel time and weighted number of stops in the on-off time-switch network. Tong and Chiou [16] made a survey regarding user route choices or switching behavior according to an Advanced Driver Information System (ADIS). In their paper, they found that users did express different characteristics and behaved differently. Additionally, the ADIS with an electronic map providing features of complex navigation, traffic condition, and route guidance is most preferred for Taiwanese drivers.

Besides, Liu [15] implemented the Dijkstra algorithm and the A* algorithm, to evaluate their performance. By using five categories of design systems, Disk-and-memory, Disjoint-networks, Grid-testing with shortcuts, Grid-testing without shortcuts, and Zero knowledge search with full network respectively, he found the total execution time of the A* algorithm on the five systems is always superior to that of the Dijkstra algorithm. Meanwhile, the design of the Disjoint-network is the best design based on the total execution time. Ikeda et al. [10] surveyed algorithms for the two-terminal shortest path problem and adopted four types algorithms-the Dijkstra algorithm, the A* algorithm, the bidirectional A* algorithm, and the bidirectional Dijkstra algorithm. The result of experiments in their paper shown the bidirectional A* algorithm is more effective than other algorithms [11-14].

The remainder of this paper is organized as follows. Section 2 introduces the architecture of the route navigation system. Section 2.1 describes definitions and notations for a new revised shortest path routing algorithm. In Section 2.2, the new revised shortest path routing algorithm for road traffic networks is illustrated. Section 3 illustrates system implementation and experiment results. Finally, some important conclusions and future work are discussed in Section 4.

2 The architecture of the route navigation system

Vehicle navigation systems have already been regarded mature commercial products for the car industry. However, there are still some drawbacks, which



need to be coped. Particularly, without considering current traffic situations, congestion problems cannot be solved by using an independent personal computer. Since, the collection of real-time traffic information plays an important role to successfully tackle congestion problems. Figure 1 presents a route navigation system, which mainly consists of a wireless GPS receiver, a high-speed traffic information server, a database, and an e-map. The GPS receiver takes charge of receiving GPS location data. The traffic information server gathers all essential traffic information including no left (right) turns, statuses of average car speed, car accidents, road repairs, number of parking space, number of hospital emergency beds, and number of hotel rooms etc. The e-map combined with the PGS receiver can be used to show driver's position, and the shortest path to guide drivers the shortest route to reach their destination. The method for finding the best route is a new revised shortest path routing algorithm. Finally, the user's dialogue interface is a friendly interface between a user and the system.

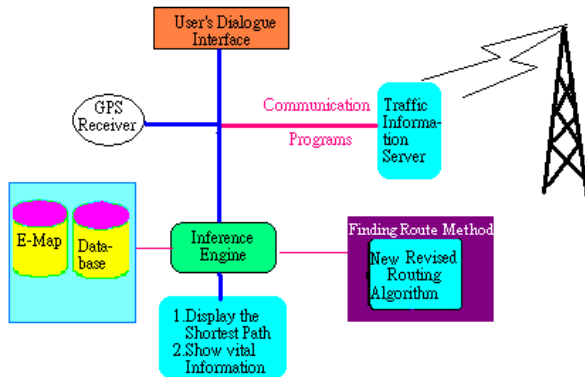


Figure 1: The architecture of a route navigation system.

With the progress of wireless network communication, online real-time road traffic information can be collected and stored in a high-speed server. Before computing the shortest path problems, every Route Navigation System first connects a traffic information server to get critical online real-time traffic messages such as car accidents, road repairs, and road congestion etc (see Figure 1). Then, all the traffic messages will be stored in the database and the system uses a new revised shortest path routing algorithm (see the next section) to find the shortest path. People maybe ask how to manage so many congestion roads in an urban area. Basically, it is quite simple. In this paper, we classify road statuses into 3 types of color: green, yellow, and red. Green color means the average car speed on a road is greater than 30 kilometers per hour. Yellow color means the average car speed on a road is less than or equal to 29 and greater than or equal to 15. Red color means the average car speed on a road is less than 14 kilometers per hour. Here, we only consider and focus on the average car speed

in red color and ignore the rest of color. A road with average car speed in red will be assigned a big M to present the distance of the road in the distance matrix. This method can greatly reduce the complexity of a road network. Therefore, the road will be avoid to be selected during computation because if chooses the road, it will cost tremendous. Using this concept, the system can easily find the shortest paths, which can automatically avoid congestion roads.

As for how to solve no left (right) turn problems, we will leave Sections 2.2 and 2.3 to explain more details. To tackle the problems, we develop a new revised shortest path routing algorithm for route networks. Whereas, before we touch how to solve the problems, we need to give some definitions and notations used in the algorithm and they will be described in the next section.

2.1 Definitions and notations

In order to clearly illustrate the new revised shortest path routing algorithm (the Wen-Hsu's algorithm) described in the next section, a simple example is given as the following. Assume that the architecture of a road network (see Figure 2) is composed of 7 nodes. Moreover, every pair of any two adjacent nodes is a two-way link. For example, the distance of the route 1-2 is 5 and the distance of the route 2-1 is also 5. Let node 1 be the source node and node 7 be the destination node. Also, suppose that the links between two nodes of every pair represent the distance denoted d_{ij} . $d_{ij} = M$ if there is no direct link and $d_{ij}=0$ if $i=j$. Based on the above assumptions, the distance matrix without considering no left turn problems can be presented as Figure 3.

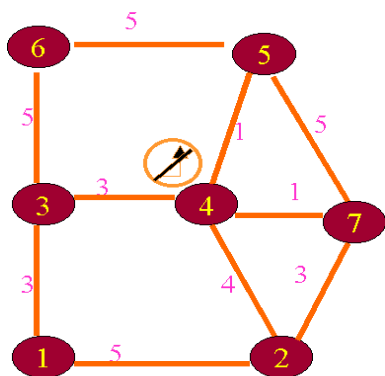


Figure 2: The architecture of a 7-node road network.

0	1	2	3	4	5	6	7
1	0	5	3	M	M	M	M
2	5	0	M	4	M	M	3
3	3	M	0	3	M	5	M
4	M	4	3	0	1	M	1
5	M	M	M	1	0	5	5
6	M	M	5	M	5	0	M
7	M	3	M	1	5	M	0

Figure 3: The distance matrix without considering no left turn problems.

Now assume that there is a no left turn sign at node 4. Although there is only a no left turn sign at node 4, as given in Figure 2, in fact, you are able to set no left turn situations as many as you like. Owing to the application of computer technologies, the algorithm can avoid the no left turn problem by using a pre-calculated and a build-in index to select all shortest paths. In the following, some

notations used in our algorithm are given as follows:

d_{ij} : the distance between node i and node j .

OPEN List: A link list includes the nodes, which connect to node n .

CLOSED List: A sorted link list.

$g(n)$: the distance of the path from the source node-to-node n .

$h(n)$: the distance of the path from node n to the destination node.

$f(n)=g(n)+h(n)$: the total distance from the source node to the destination node via node n .

2.2 The new revised shortest path routing algorithm

The A* algorithm by Hart and Nilsson [8] was developed by revising the Greedy algorithm. Basically, the Greedy algorithm uses a heuristic function to determine the best frontier node to expand next node, node n . For the route finding problem, a good heuristic would be straight-line distance to the goal (destination). Although the Greedy algorithm performs very fast in finding routes, sometimes its final result is not the shortest path. Therefore, the A* algorithm for finding the best route and an algorithm for finding edges contained in better routes can be presented by the following equation:

$f(n)=g(n)+h(n)$, where:

n : the current arrived node (i.e., the frontier node);

$g(n)$: the shortest distance from the starting node-to-node n ;

$h(n)$: the shortest distance from node n to destination;

$f(n)$: the minimum distance from the source node to destination via node n .

In addition, many research results [4,10,11,14] show the execution time of the A* algorithm was faster than that of the Dijkstra algorithm when a large number of nodes had been chosen. Hence, we adopt the A* algorithm and revise it to form our new algorithm. In the new algorithm, first sort out all no left turn points to generate a revised distance matrix. Then, use the A* algorithm to find all shortest paths. Although this paper gives an example for solving a no left turn problem, it can also work out no right turn or congestion problems. As for solving congestion problems, a high-speed server needs to provide some important information about accidents, road repairs, and flow speed of each road. Like solving one-way road problems, a big M is assigned to all congestion roads in the distance matrix. The revised routing algorithm (i.e., the Wen-Hsu's algorithm) is presented as follows:

Step 1: Construct a distance matrix. First let each distance, d_{ij} , of every pair be the entries of the matrix where $d_{ij}=0$ if $i=j$ and $d_{ij}=M$ if there is no direct link between i and j .

Step 2: Set all no left turn nodes as dummy nodes. Then directly connect each pair of the adjacent nodes of the dummy nodes. Calculate the distance of all pairs of routes, which are adjacent to the dummy nodes. Meanwhile, build an index for all the pairs of routes.

Step 3: Find the minimal values between the old value and the new value (see Figures 5 and 6).



- Step 4: Generate the revised distance matrix as Figure 8 (Replace the values in Figure 1 with the minimal values).
- Step 5: Based on the revised distance matrix, put the start node 1 on a list called OPEN. Set $g(1) \leftarrow 0$ and $f(1) \leftarrow h(1)$
- Step 6: If OPEN List is empty, exit with failure; otherwise continue.
- Step 7: Remove from OPEN List that node n whose value of f function is the smallest value and put it on a list called CLOSED.
- Step 8: If n is a goal node, exit with the solution path obtained by tracing back through the pointers; otherwise continue.
- Step 9: Expand node n , generating all of its successors. (If there are no successors, go to Step 6.) For each successor n_i , compute $g_i \leftarrow g(n) + c(n, n_i)$. $c(n, n_i)$ is the distance from node n to its i_{th} successors
- Step 10: If a successor n_i is not already on either OPEN List or CLOSED List, set $g(n_i) \leftarrow g_i$ and $f(n_i) \leftarrow g_i + h(n_i)$. Put n_i on OPEN List and direct a pointer from it back to n .
- Step 11: If a successor n_i is already on OPEN or CLOSED, and if $g(n_i) > g_i$, then update it by setting $g(n_i) \leftarrow g_i$ and $f(n_i) \leftarrow g_i + h(n_i)$. Put n_i on OPEN List if it was on CLOSED List and redirect to n the pointer from n_i .
- Step 12: Go to Step 6.

For example, based on Figure 2, when node 4, which is a no left turn intersection, is taken into account, it is regarded as a dummy node. Every pair of nodes, which goes through node 4, is directly connected under ignoring the dummy node, as shown in Figure 4. Although node 4 is ignored, the new distance of the route 3-4-5 is M (i.e., the distance of 3-4, which is 3, plusses the distance of 4-5, which is a big number, M , is still assigned an M), because of the no left turn at the intersection, node 4. Similarly, the distance of the route 5-4-3 can be expressed $1+3=4$.

So, like the third step of the algorithm, the new distance of each route, which skips node 4, needs to be recalculated as Figure 5. According to Figure 5, we first consider the old distance of routes in Figure 3, which does not consider a dummy node and find their values (i.e., distance) recorded in Figure 6. After comparing the new value in Figure 5 and the old value in Figure 6, we choose the minimal value of every same route (i.e., they have the same source node and the destination node) are recorded as in Figure 7. On the other hand, the minimal value of the same routes in Figure 7 is $\min(\text{old value}, \text{new value})$. Then the distance of their corresponding routes in Figure 3 are replaced by the minimal values in Figure 7. Finally, the revised distance matrix needs to be modified as Figure 8. In Step 3, once the modified distance matrix (see Figure 8) has been found, the algorithm carries on the procedures from Step 5 to Step 12. It needs to repeatedly execute Step 5 to Step 12 until all shortest paths have been sorted out. Just remember when the shortest path 1-3-2-7 is found, the Route Navigation System (RNS) must recover the dummy node back as the route 1-3-4-2-7 by referencing the index described as Figure 7. Because of high speed of computation, the performance of the RNS is quite satisfactory.

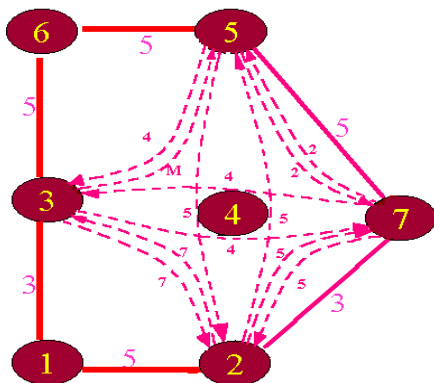


Figure 4: The revised architecture of routes the road network.

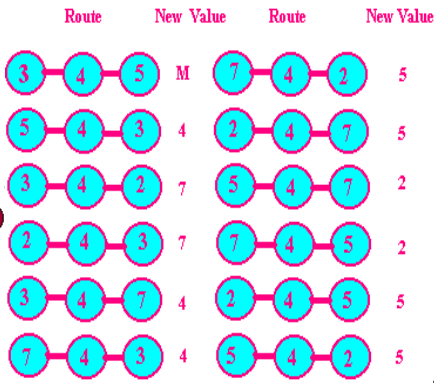


Figure 5: The revised distance of the with a dummy node.

Route	Old Value	Route	Old Value
3-5	M	7-2	3
5-3	M	2-7	3
3-2	M	5-7	5
2-3	M	7-5	5
3-7	M	2-5	M
7-3	M	5-2	M

Figure 6: The original distance of the routes without a dummy node.

Min(old value, new value)		Min(old value, new value)	
Route	Value	Route	Value
3-4-5	M	7-4-2	3
5-4-3	4	2-4-7	3
3-4-2	7	5-4-7	2
2-4-3	7	7-4-5	2
3-4-7	4	2-4-5	5
7-4-3	4	5-4-2	5

Figure 7: The minimal distances of the routes after comparing the new value and the old value.

	0	1	2	3	4	5	6	7
1	0	5	3	M	M	M	M	M
2	5	0	7	M	5	M	3	3
3	3	7	0	M	M	5	4	4
4	M	M	M	0	M	M	M	M
5	M	5	4	M	0	5	2	2
6	M	M	5	M	5	0	M	M
7	M	3	4	M	2	M	0	0

Figure 8: The revised distance matrix of the road network.



Figure 9: The system architecture of hardware -the connection of a PC, a wireless PCMCIA card, and a wireless GPS.

3 System implementation and experiment results

To calculate the efficiency of the new algorithm, we have implemented a Route Navigation System (RNS). The architecture of hardware is shown as Figure 9. The system was built by using the new revised routing algorithm. The RNS is developed on a PC-based portable computer. The speed of the Center Process Unit is Pentium 1.7GHz. The size of its main memory is 256M RAM. Through a communication PCMCIA card, the application system is able to connect a server via the Internet and get some vital guidance information, for instance, the number of parking spaces in a parking lot, the number of beds in a hotel, etc (see Figure 10). Meanwhile, a wireless Global Positioning System (GPS) receiver is installed on the system. Therefore, the RNS not only can provide the position of the car but also suggest the shortest path from source to destination. The system also can avoid congestion if some roads have heavy traffic because of accidents, rush hour, or road construction (see Figure 11). Basically, in the experiment of performance evaluation, we adopt 4 kinds of algorithms (i.e., Dijkstra's algorithm, Dijkstra+'s algorithm, the A* algorithm, and the revised routing algorithm (i.e., Wen-Hsu's algorithm).)

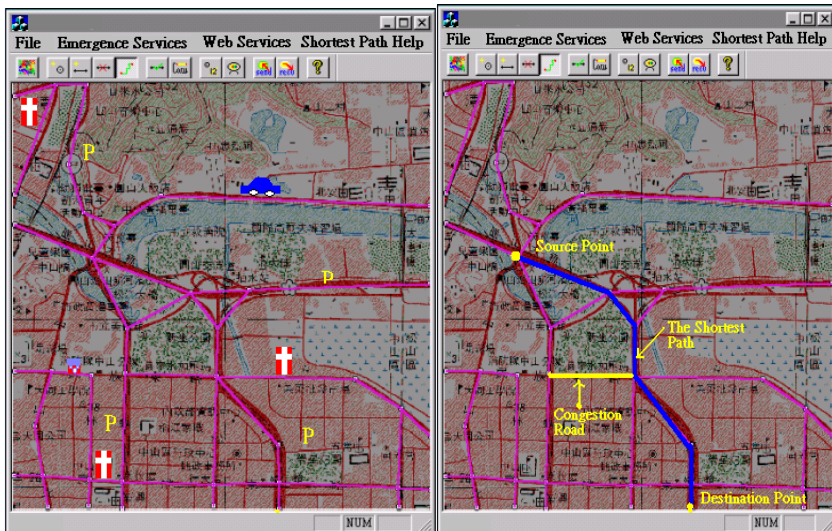


Figure 10: The show map for hospitals, police stations, and repairing centers. Figure 11: The shortest path from source to destination with free congestion.

The main difference between the Dijkstra algorithm and Dijkstra+'s algorithm is in the latter we add Step 1 to Step 5 in the revised routing algorithm to count a revised distance matrix. Then use the Dijkstra algorithm to find all shortest paths from the source to the destination. In this experiment, we chose a road network containing 1000 nodes, 2000 nodes, 3000 nodes, and 4000 nodes respectively to

analyse their execution time. Meanwhile, the experiment was also done by classifying a short range (about 12 kilometers), a medium range (about 24 kilometers), and a long range (about 48 kilometers). For computing the average execution time, every situation of the algorithms was run 30 times. Moreover, during the computation of the algorithms, all data of the distance matrix are loaded into the main memory of a personal computer. All no left turn conditions (or no right turn conditions) are automatically pre-calculated and directly updated once a user chooses them. It is worth mentioning that the above extra work is only done once when the road network is changed in the future.

Figure 12 shows the average execution time of the A* algorithm for the 1000 nodes is 0.099 seconds, which is very close to that of the Wen-Hsu's algorithm, 0.098 seconds. Whereas the average execution time of the A* algorithm and the Wen-Hsu's algorithm (e.g., 0.667 and 0.651) are much better than that of Dijkstra's algorithm and Dijkstra+'s algorithm (e.g., 1.418 and 1.522). In general, the speed of the execution time of the A* algorithm and the Wen-Hsu's algorithm is superior to that of the Figure 12 Average execution time for Dijkstra's and the Dijkstra+'s algorithms except for the situation with 1000 nodes. There is no different between the A* algorithm and the Wen-Hsu's algorithm. In other words, a better efficiency can be generated when a large number of nodes are chosen. Consequently, this experiment proves the execution time is still pretty fast although it takes into account no left turn problems.

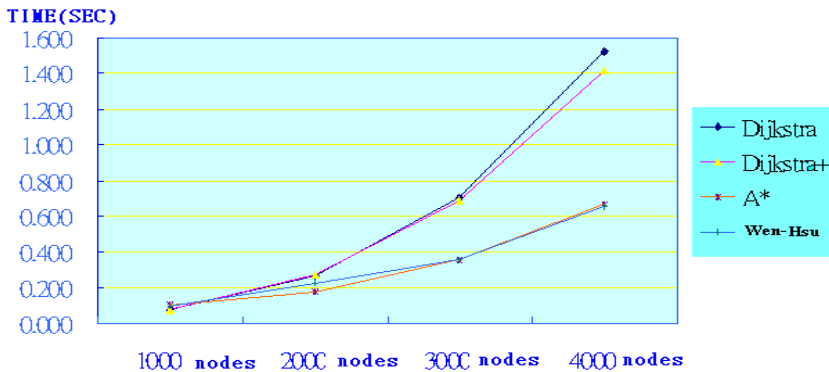


Figure 12: the average execution time of the four algorithms.

4 Conclusions and future work

This paper proposed a route navigation system with a new modified algorithm for solving the no left (right) turns, one-way roads, and congestion problems on road networks. For evaluating its performance, a prototype system, Route Navigation System (RNS), was built. The system physically proves the efficiency of the system architecture. In terms of the final results of the performance analysis, the system not only provides a good solution for solving road congestion problems but also responds much faster. We strongly believe

the system with the new revised routing algorithm will play an important role in traffic navigation systems for users' guidance on road networks.

Although this paper designs and analyzes the RNS, there are still several aspects where we can further improve its functions. In particular, we can extend its functions to assist the RNS to have an automatic reasoning mechanism using a knowledge-based inference engine. In addition, the method of artificial neural networks for searching the best routes of which we had used in the past can also be added for helping make decisions. So, knowing how to build an efficient searching and learning mechanism for the RNS will be also a major research issue in the future.

References

- [1] Chabini, I., Discrete Dynamic Shortest Path Problems in Transportation Applications, *Transportation Research Record*, 1998.
- [2] Chabini, I., A New Algorithm for Shortest Paths in Discrete Dynamic Networks, *8th IFAC/IFIP/IFORS Symposium on transportation systems, Tech Univ Crete, Greece*, June 1997, pp. 16-18.
- [3] Chen, Y.L. & Yang, H.H. Minimization of Travel Time and Weighted Number of Stops in a Traffic-light Network, *European Journal of Operational Research*, 144, 2003, pp. 565-580.
- [4] Erlan, F., Fast Shortest Path Algorithm for Large Road Networks, *Conf2001, Department of Engineering Science, University of Auckland, New Zealand*, 2001, pp. 1-10.
- [5] Fu, L. & Rilett, L.R., Expected Shortest Paths in Dynamic and Stochastic Traffic Networks, *Transportation Research, Part B: Methodological*, vol. 32, no. 7, 1996, pp. 499-516.
- [6] Helal, A., Moore, S.E. & Ramachandran, B.D., An Integrated Navigation System for Visually Impaired and Disable, Wearable Computers, Proceedings. *Fifth International Symposium on 2001*, pp. 149 -156.
- [7] Horn, M.E.T., Efficient Modeling of Travel in Networks with Time-varying Link Speeds, *CSIRO Mathematical and Information Sciences Technical Report CMIS 99/97*, <http://www.cmis.csiro.au/Mark.Horn/>, 1999.
- [8] Hart, P.E. & Nilson, N.J., A Formal Basis of the Heuristic Determination of Minimum Cost Paths, *IEEE Transactions of Systems Science and Cybernetics*, 4(2), 1968, pp. 100-107.
- [9] Hsu, H.W., Dynamic Route Planning with Considering Real-time Traffic Flow on the Vehicle Navigation System, *Ph.D. thesis*, 1999.
- [10] Ikeda, T., Hsu, M.Y. & Imai, H., A Fast A* Algorithm for Finding Better Routes by AI Search Techniques, *Vehicle Navigation and Information Systems Conference Proceedings, IEEE*, , 1994, pp. 291-296.
- [11] Ikeda, T. & Imai, H., Fast A* Algorithms for Multiple Sequence Alignment, *Proceedings of the Genome Informatics Workshop, Universal Academy Press*, 1994, pp. 90-99.



- [12] Ikeda, T., Imai, H., Nishimura, S., Shimoura, H., Hashimoto, T., Tenmoku, K. & Mitoh, K., Bidirectional A* Algorithm for the Shortest Path Problem in Route Navigation Systems, *IPSJ SIG Notes 94-AL-40-12*, IPSJ, 1994.
- [13] Ikeda, T. & Imai, H., Fast A* Algorithms for Multiple Sequence Alignment. *IPSJ SIG Notes 94-AL-42-7*, IPSJ, 1994.
- [14] Ikeda, T. & Imai, H., Fast A* Algorithm for Multiple Alignment Problem and Its Computational Efficiency in Practice, *ISM Reports on Statistical Computing, "Optimization: Modeling and Algorithms 7"*, Institute of Statistical Mathematics, Vol. 77, 1995, pp. 14-25.
- [15] Liu, B., Routing Finding by Using Knowledge about the Road Network, *IEEE Transactions on System, man, and Cybernetics-Part A: Systems and Humans.*, Vol. 27, No. 4, , July, 1997, pp. 425-430.
- [16] Tong, C.C. & Chiou, A., A Preliminary Experimental Investigation of Display Patterns for Driver-in-vehicle Information System in Taiwan, *Vehicle Navigation and Information Systems Conference Proceedings, IEEE*, 1994, pp. 389-394.