A Simplified Method to Estimate Travel Cost based on Traffic-Adaptable Heuristics for Accelerating Path Search

Jin-Deog Kim, *Member, KIMICS*

Abstract—In the telematics system, a reasonable path search time should be guaranteed from a great number of user's queries, even though the optimal path with minimized travel time might be continuously changed by the traffic flows. Thus, the path search method should consider traffic flows of the roads and the search time as well. However, the existing path search methods are not able to cope efficiently with the change of the traffic flows and to search rapidly paths simultaneously.

This paper proposes a new path search method for fast computation. It also reflects the traffic flows efficiently. Especially, in order to simplify the computation of variable heuristic values, it employs a simplification method for estimating values of traffic-adaptable heuristics. The experiments are carried out with the A* algorithm and the proposed method in terms of the execution time, the number of node accesses and the accuracy. The results obtained from the experiments show that the method achieves very fast execution time and the reasonable accuracy as well.

Index Terms—Telematics, Path Search, Heuristics, Traffic Flows

I. INTRODUCTION

Owing to the rapid growth of the network communications and the diversification of way to use information in recent years, the traditional PC with wired communication plays an important role as an information creator and the mobile devices with wireless communication will be essential parts as an information consumer. The CNS(Car Navigation System) is getting a one of the killer application in the mobile devices such as mobile phone, PDA and telematics terminal.

There have been many works on finding the paths with shortest time recently. To find a path with shortest distance is common approach in telematics systems conventionally. Even if the path searched from a telematics device has shortest distance, the path is always not optimal path in terms of travel time, particularly when the road is congested and an accident occurs. Only few attempts have so far been studied on finding the paths with shortest travel time.

The path search method in the telematics system should consider traffic flows of the roads as well as the shortest travel time. The computation time should be also fast. It was difficult for the existing path search methods to reflect the traffic flows. Moreover, the computation time should be fast. The existing path search methods are not able to cope efficiently with the change of the traffic flows. Especially, it is also expected that the client based path finding would be a sharp rise in demand due to the lower maintenance cost.

However, the search method to use traffic information needs more computation time than the existing shortest path search method. Consequently, the path search method for telematics needs high accuracy and fast computation as well. Because queries from a great number of clients are concentrated upon a server in the server-oriented system, the fast search with high accuracy has a direct influence on the efficient management of the telematics server. In addition, to reduce its computation time is still a prerequisite in the client-oriented system in consideration of low system performance of telematics terminals.

This paper proposes a new path search method adaptable to traffic information. It is a variation of the A* algorithm[1,6,8] and uses variable heuristics according to the change of the traffic flows. Moreover, the search method employs a simplification method for estimating values of the adjustable heuristics for the sake of fast computation. The simplification method only adds graded decimal values instead of multiplication operation of floating point numbers.

The experiments conducted on the real-time traffic data clearly show that the proposed method outperforms the Dijkstra[3] and the A* algorithm. The simplification method also contributes greatly to the fast computation for calculating values to be estimated based on variable heuristic values. It is expected to be an important technology for intelligent car navigation of the telematics system to be widely used.

The rest of this paper is organized as follows. Section 2 investigates the related works on the path search algorithm. Section 3 examines a new path search method with the adjustable heuristics and the simplification method in detail. In section 4, the results of experiments on real traffic data are presented and analyzed. Finally, section 5 gives concluding remarks.

Manuscript received June 29, 2007.

Jin-Deog Kim is with the Department of Computer Engineering, Dongeui University, Busan, 614-714, Korea (Tel: +82-51-890-1745, Fax: +82-51-890-2629, Email: jdk@deu.ac.kr)

II. RELATED WORKS

The path search usually means the routes to reach destination from departure on the roads. Even though the searched path is shortest, however, the path is always not optimal path in terms of driving time, particularly when the road is congested and accidents occur on the searched path. Therefore, the method to search path in the telematics should consider traffic information for shortest travel time.

Many researches on path search have been studied so far. The Dijkstra algorithm[3] and the A* algorithm[1] are widely used for path finding. Although the Dijkstra algorithm yields optimal path in the accuracy, it is a time-intensive algorithm. The A* algorithm[1] shows a good performance as a results of employing the heuristic functions in the execution time. The performance is dependent on these heuristics. It is very difficult and time-intensive to decide a proper heuristic, particularly in the road networks whose flows are continuously changed.

Yang et el[14] proposed an approach to search path in the hierarchical road networks. It classifies the roads into three types(Major Road, Highways, Freeways).

Jagadeesh et el[16] proposed a hierarchical routing algorithm with acceptable loss of accuracy. A network pruning technique has been incorporated into the algorithm to reduce the search space. However, it doesn't explain how to make a grid and how to adapt traffic flow into heuristics.

The HiTi graph model[13] also proposed an approach to structure a topographical road map in a hierarchical fashion. It also proposed a new shortest path algorithm named SPAH. However, these don't apply the traffic flow in searching path.

There have been many commercial services[9,10] on the car navigation system recently. Most of them, however, provide the shortest-distance path only. The literatures [11,12] proposed another path search algorithms. The above researches are different from our study in that they have never dealt with variable heuristics and the simplification for estimating cost to goals.

III. PATH SEARCH METHOD

In order to search path in games, the heuristic weights of the A* are assigned to each pixel respectively[4]. Suppose that telematics system applies the A* to road network. Road network data are usually irregular and the number of node is very tremendous. Because assigning different heuristic weights to each node is very difficult and laborious, telematics systems have to use a fixed heuristic weight. As mentioned above, it is impossible to increase the performances of both the execution time and the accuracy simultaneously under a fixed heuristic[5]. Moreover, it is also impossible to decide a proper heuristic value immediately according to traffic flow changed continuously.

The newly proposed search method looks like A* except variable heuristics and the simplification. While the A* generally uses a fixed heuristic value, the proposed method decides the heuristic according to the velocity of its roads. We call the methods APS(Adaptable Path Search) and APSS(Adaptable Path Search with Simplification)

A. Representation of Road Networks

Figure 1 shows the road networks with orientation and rotation information.

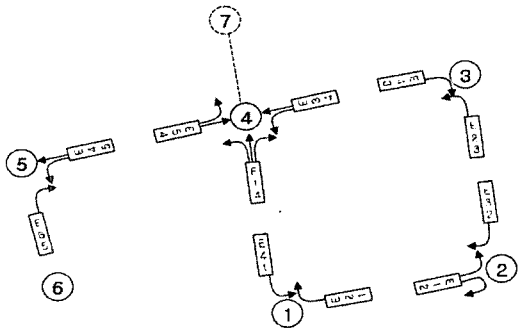


Fig. 1 Road Networks with Orientation and Rotation

In this paper, the road networks are represented by link-oriented approach. Each link has several connected link lists which express orientation and rotation information. For example, link E12 in the figure 2 is a link from node 1 to node 2. When the car goes through node 2, it can go the link E23(left turn) or E21(U-turn). The proposed link-oriented representation can reflect partial update immediately in the telematics devices.

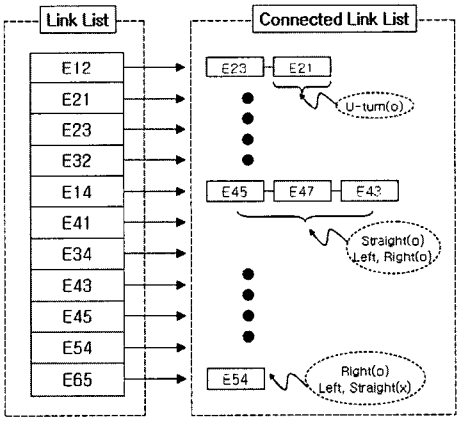


Fig. 2 Representation of Link-Oriented Road Networks

B. Adjustable Heuristic based on Fixed Grid

For the sake of efficient path search, we would like to focus attention on a new method to adjust heuristic to traffic flow immediately without human's interference.

In order to adjust heuristic to continuously changed traffic flow, this paper employs the fixed grid[2,7]. The fixed grid decomposes entire map space into a number of unit cells, and each cell contains several roads.

The values for heuristics of each grid cell are determined as the maximum velocity of the roads contained in each cell as shown in figure 3.

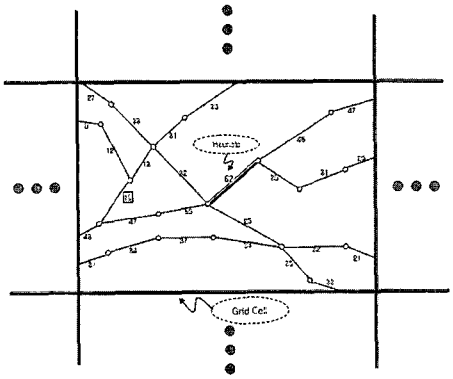


Fig. 3 Heuristic Value of Grid Cell

The velocity of the road is in inverse proportion to driving time. Thus, the heuristic weight is also in inverse proportion to the value. For example, while the high value(weak heuristic) could be assigned at the light traffic for high accuracy, the low value(strong heuristic) could be assigned at the heavy traffic for fast computation. Therefore, the adjustable heuristic is expected to bring about high accuracy and fast response time as well. The heuristic weight can be defined as follows.

heuristic weight = length of road link / velocity of road link

The shape of the search space of the A* is nearly a single ellipse with a departure and a destination as the peak points of ellipse. At the strong heuristic, the shape is narrow. At the weak heuristic, the shape is nearly circular.

The APS and APSS carry out partial search on the basis of each grid unlike the A*. The partial search yields the reduction of the search space and the execution time as well. Moreover, the search space is adaptable to current traffic flow.

C. Algorithm of the APS

The APS algorithm visits all the nodes connected with a current node, and then selects a node to be estimated that it has a minimum travel cost[15]. If the node is goal, the search is completed. The following pseudo code describes the APS algorithm.

APS Algorithm

```
{  Input : Start & Goal Node
   Data : Heuristic Value of Each Grid
   Output : Optimal_Path(List of Node)
   ------------------------------
   CurrentNode = Start;
   ClosedNode.Add(CurrentNode);
   While( CurrentNode != GoalNode)
   { Get Nodes connected CurrentNode;
                // exclude the nodes of ClosedNode
     OpenNode.Add(Nodes)
     ForEach( Ni in OpenNode )
     { Calc. Cost of each Node
        Cost : F(Ni) = G(Ni) + H(Ni),
        G(Ni) : Cost_From_Start(Ni)
        H(Ni) : Cost_To_Goal(Ni)
     }
     Select Nj which has min F(n);
     CurrentNode = Nj;
     ClosedNode.Add(CurrentNode);
   }
   Optimal_Path : ClosedNode
}
```

The cost of each node could be defined as follows : f(n) = g(n) + h(n). The g(n) is the already calculated travel cost from departure to current node. The h(n) the APS algorithm is the estimated travel cost to goal. The adjustable heuristics are used for estimating the h(n). The OpenNode of algorithm is a set of candidate nodes to be visit, and the ClosedNode is a set of already selected nodes. The following pseudo code describes the function 'Cost_To_Goal'.

Cost_To_Goal_Function

```
{  Input : Ni, GoalNode
   Data : Heuristic Value of Each Grid
   Output : Estimated Cost to Goal Node
   ---------------------------------
   Draw a straight line from Ni to Goal
   Get Grids overlapped with the line
   EstimatedCost = 0;
   ForEach( Gi in Grids )
   {  EstimatedCost += LineLength(Gi) / Heu(Gi) }
   return EstimatedCost
}
```

The outstanding difference between the A* and the APS is the method to get the cost h(n). For efficient estimating the cost, this paper applies the above adjustable heuristics into APS algorithm as shown in the function 'Cost_To_Goal'.

For example, if the maximum speed of a grid is very low, the cost h(n) would be overestimated. It yields strong heuristic weight and the fast computation. On the contrary, if the maximum speed is very high, the cost would be underestimated, and the accuracy increases.

Therefore, the APS with adjustable heuristics can search a near optimal path because of the fast computation at the traffic congestion and the accurate search at the light traffic.

D. APSS with Simplification for Estimated Cost

The above function, Cost_To_Goal, needs to measure the length of line in each grid cell. It requires so several complex floating point computations that it results in slow computation.

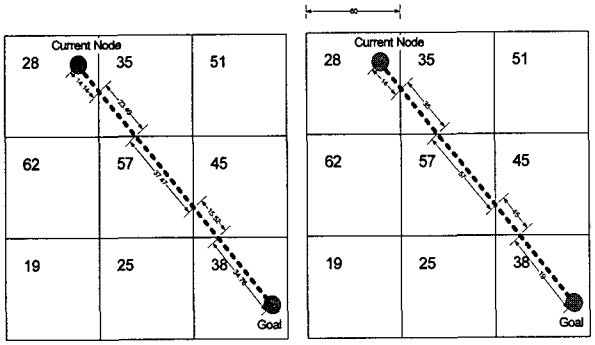
In order to reduce this computation time, a new simplification method for estimating cost to goal is proposed in this paper. The following pseudo code describes the simplification method.

Simplified_Cost_To_Goal_Function

```
{  Input : Ni, GoalNode
   Data : Heuristic Value of Each Grid
   Output : Estimated Cost to Goal Node
   ---------------------------------
   Get a grid cell(Gcur) to contain Ni
   Get a grid cell(Gdes) to contain Goal
   Get center points(Ccur, Cdes) of Gcur and Gdes
   Calc. slope(Sstart) between Cur and Cdes
   Do {
      Calc. slope(Scur) between center of Ni and Cdes
      If ( Sstart <= Scur)
         Gcur = adjacent Grid of X axis added by 1
```

```
    Else   Gcur = adjacent Grid of Y axis added by 1
  } While( Gcur != Cdes)
  EstimatedCost = Length to Goal / Average Heu.(Gcur)
  Return EstimatedCost
}
```

Figure 4 shows the difference between the APS and the APSS. The APSS only adds graded decimal values instead of multiplication operation of floating point numbers with due regard to the gradient between a departure and a destination in order to simplify the computation of estimation values. The estimated cost of APS is (14.14/28 + 23.49/35 + 37.47/57 + 15.52/45 + 34.76/38) = 3.093. The estimated cost of APSS is 125.38 / ((28+35+57+45+38)/5) = 3.088. Therefore, while the estimated cost is nearly identical, the computation step is greatly simplified. The effect of this simplification will be discussed in section 4.



(a) APS (b) APSS

Fig. 4 Estimating Cost to Goal

E. Performance Tuning

Even though the APSS is expected to show fast computation, the accuracy will slightly decrease. For the high accuracy, the change of the size of grid is worth consideration. In case of small-sized grid, the accuracy might increase because the APSS reflects more heuristics. Figure 5 shows these phenomena. The heuristics 57 and 38 of long lines in the figure are represented by 3 times respectively. The heuristics of the line is reflected as much as the length of line. Thus, the results of the path search are more accurate, even though the computation time slightly increases.

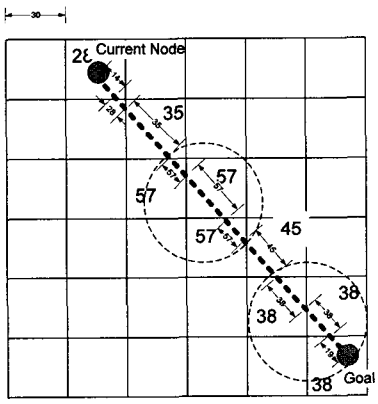


Fig. 5 Change of Grid-Size

IV. PERFORMANCE EVALUATION

The performances of the APS and APSS are compared with the Dijkstra and the A*. The test data for experiments are the road networks and real-time traffic data in Busan City. The path searches are carried out 700 times. The departure and the destination are selected randomly. The fixed heuristic values of the A* algorithm are 20, 40, 60, 80, 100 respectively. Figure 6 shows the path search program which is implemented on the Pentium IV PC and the C++ programming language.

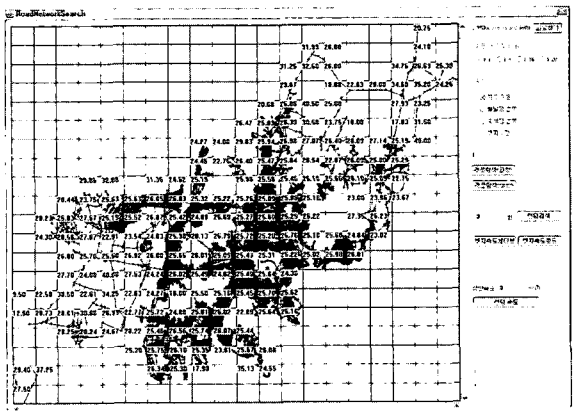


Fig. 6 Implementation of Path Search System

A. Accuracy

The terminology 'accuracy' in this paper can be defined as follows. The accuracy of Dijkstra algorithm is always 100% as mentioned above.

$$accuracy = \frac{\text{travel distance of optimal path}}{\text{travel distance of a given algorithm}}$$

Figure 7 depicts the accuracies of the A* and the APS. The APS generally outperforms A* algorithms except very high heuristic value(weak heuristic). In the A*, the execution time is in inverse proportion to the accuracy. Figure 7 and 8 will show this characteristic of the A*.

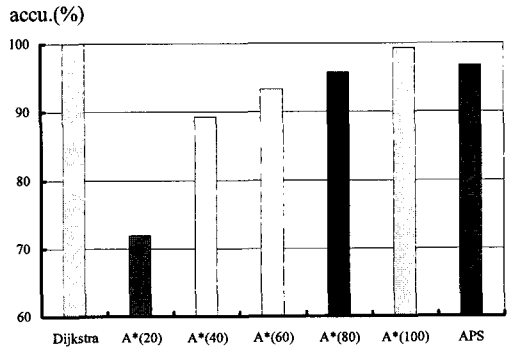


Fig. 7 Average Accuracy

The relationship between the accuracy and the travel distance is shown in table 1. The performance of short distance is better than that of long distance. If the distance is long, the accuracy is low. At the long distance and low heuristic value, the accuracy of the A* is very low(about 50%). On the contrary, the APS shows good

performances (about 95% and more) regardless of the travel distances.

Table 1 Accuracy by Travel Distance

method / distance	Dijk.	A* 20	A* 40	A* 60	A* 80	A* 100	APS
short(~10k)	100	90.2	93.1	95.2	98.5	100	99.8
mid.(10-40k)	100	70.5	88	92	95.3	99.3	95.8
long(40k~)	100	52.9	80.4	87.7	93.4	99	95.1

B. Execution Time

Figure 8 depicts the average execution time for searching paths. The times of the Dijkstra are always longer than the others. The execution time of the A* is various according to heuristic weight. Generally the average execution time of the APS is even shorter than that of the A* except very strong weight. While the execution time of the A* is very short at the strong weight(low value), the accuracy is very high as shown in the figure 7. The experimental results show that the execution time coincides with our previous expectations.

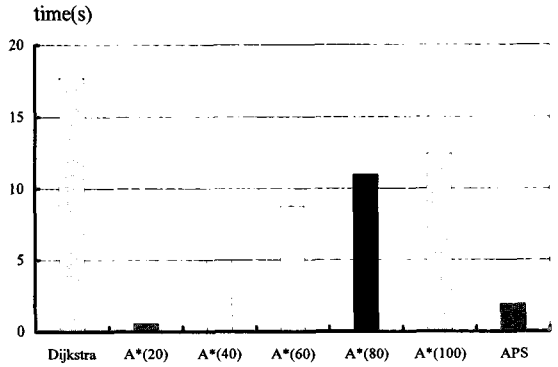


Fig. 8 Average Execution Time

The table 2 summarizes the execution of each method in case of various travel distances. At the long distance and high heuristic value, the execution time of the A* is very long(near to Dijkstra). The APS shows reasonable response time regardless of the travel distances.

Table 2. Execution Time by Travel Distance

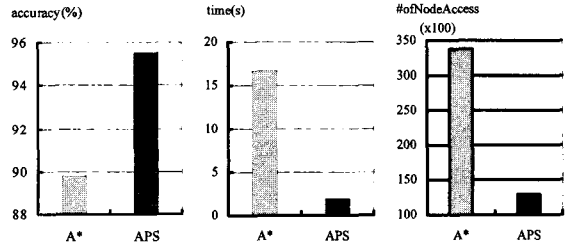
method / distance	Dijk.	A* 20	A* 40	A* 60	A* 80	A* 100	APS
short(~10k)	3.9	0.03	0.34	0.84	1.28	1.63	0.13
mid.(10-40k)	23.35	0.65	6.6	11.85	14.7	16.5	2.3
long(40k~)	47.56	7.4	34	42.4	44.8	45.7	12.67

C. Analysis of Each Method

Although the accuracy of the Dijkstra is always best(100%), the execution time and the number of node accesses is always worst.

The accuracy of the APS closes on that of the Dijkstra. The execution time and the number of node accesses of the APS are just 12% and 25% of the Dijkstra respectively. Particularly, because the APS is able to adjust the heuristic weight to traffic flow, it might bring about near optimal path.

The performances of the A* and the APS under the same conditions are analyzed in the figure 9. If the execution times are nearly the same, the APS definitely outperforms the A* in the accuracy(figure 9.a). If the accuracies are nearly the same, the APS also outperforms the A* in the execution time. The APS needs only 10% time of the A*(figure 9.b). If the accuracies are nearly the same, the APS also shows a good performance(figure 9.c).

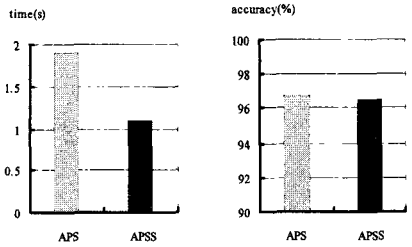


(a) accuracy (b) time (c) node access

Fig. 9 Performance Comparison(A* vs. Grid)

D. Effect of Simplification

Figure 10 shows the performances of APS and APSS. Even though the performance of APS is slightly better than that of APSS(about 0.3) in the accuracy aspect, the APSS outperforms APS(about 60%) in the operation time.



(a) time (b) accuracy

Fig. 10 Effect of the Simplification

V. CONCLUSIONS

This paper proposed APSS which is a new path search method for fast computation. It also reflects the traffic flows efficiently. The method uses adjustable heuristics for reflecting the traffic flows and simplifies the step for estimating cost to goal. The proposed APSS achieved good performances at the execution time and the accuracy as well. The APSS is also able to reflect real-time traffic flows due to the adjustable heuristic and the simplification method.

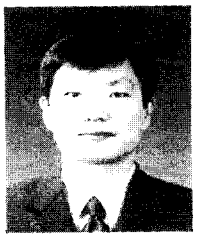
The results obtained from the experiments show that the execution time and the accuracy of the A* are easily influenced by the heuristic weight nevertheless it could

not be determined automatically by the system. Moreover, there is no way to upgrade both the execution time and the accuracy simultaneously. On the contrary, the APS achieves good performances in terms of the operation time(12% of the Dijkstra) and the accuracy(about 95% of the Dijkstra). The APSS improves the operation time(about 60% of the APS).

In summary, it seems reasonable to conclude that the proposed APSS is appropriate for the telematics server which should process a great number of queries from the several clients and the telematics clients with low processing capacity. Particularly, because the pre-searched path should be recalculated whenever the traffic flow changes, the APSS with fast execution and the adjustability is expected to be important technology for telematics system to be widely used recently.

REFERENCES

[1] http://en.wikipedia.org/wikiA%2A_search_algorithm.
[2] H. Lu, B.C. Ooi, "Spatial Indexing : Past and Future", IEEE Data Engineering Bulletin, Vol. 16, No. 3, pp 16-21, 1993.
[3] William Stallings, "Data & Computer Communications, sixth Edition", Prentice Hall, Inc, 2001.
[4] http://theory.stanford.edu/~amitp/Game. Programming.
[5] Stephan Winter, "Modeling Costs of Truns in Route Planning", Journal of GeoInformatica, Vol. 6, No. 4, pp. 345-360, 2002
[6] Se-Il Lee, "Units' Path Finding Method Proposal for A*Algorithm in the Tilemap, Journal of KCSI, Vol. 9, No. 3, pp. 71-77, 2004.
[7] Hyun-Sub Lee, Jun-Hwan An, Jin-Deog Kim, "Optimal Path Navigation Algorithm based on Traffic Information", Proc. of KMIC, Vol.8, No. 2, pp. 425-428, 2004.
[8] H. Kaindl, G. Kainz, "Bidirectional Heuristic Search Reconsidered", Journal of Artificial Intelligence Research, Vol.7, pp.283-317, 1997.
[9] http://qnavi.bizemeka.com.
[10] htp://drive.nate.com.
[11] Stefano Pallottino, Maria Grazia Scutella, "Shortest Path Algorithms in Transportation Models : Classical and Innovative Aspects", TR, Univ. of Pisa, 1998.
[12] Woo Young Kwon, Sanghoon Lee, Il Hong Suh, "A reinforcement learning approach involving a shortest path finding algorithm", Proc. of IROS2003, Vol. 1, pp. 436-441, 2003.
[13] Sungwon Jung, Sakti Pramanik, "An Efficient Path Computation Model for Hierarchically Structured Topographical Road Maps", TKDE, Vol. 14, No. 5, pp.1029-1046, 2002.
[14] T. A. Yang, S. Shekhar, B. Hamidzadeh and P. A. Hancock, "Path planning and evaluation in IVHS databases," VNIS, pp.283-290, 1991.
[15] J. D. Kim, D.H. Kim, D.S. Cho, C.H. Ban, K.H. Kim, K.W. Min, "An Efficient Path Search Method based on Adjustable Heuristic to Traffic Flow", APIS, Vol. 6, pp. 510-513, 2007.
[16] G. R. Jagadeesh, T. Srikanthan, K. H. Quek, "Heuristic Techniques for Accelerating Hierarchical Routing on Road Networks", IEEE Trans. Intelligent Transportation Systems, Vol. 3, No. 4, pp. 301-309, 2002.



Jin-Deog Kim
was born in Busan, Korea, 1968. He received his undergraduate education at the Busan National University, Korea. He earned his M.S. and Ph.D. in Computer Engineering from the Busan National University, Korea. He is currently with the Department of Computer Engineering, Dongeui University as an associated professor. He has published over 100 papers in the areas of database and geographical information system. His areas of research include a query processing in the spatial database, parallel processing of spatial operation, update strategy of moving object, path search method in the telematics systems and LBS application with RFID, etc.