A Dynamic Path Computation Database Model in Mobile LBS System
모바일 LBS 시스템에서 동적 경로 계산 데이터베이스 모델

주 용 진[*]
Yong Jin Joo

요 약 최근, 모바일 시스템에서 DBMS를 활용한 위치기반서비스에 대한 관심이 높아지고 있으며, 향상된 차량항법 (in-vehicle navigation) 시스템에 있어 효과적인 저장, 트랜잭션 관리, 모델링과 공간 질의를 통해 현행 파일 기반 시스템이 가지는 한계를 극복할 것으로 기대되고 있다. 특히, 도로 네트워크 데이터는 경로 탐색 시스템에 있어 가장 중요한 영역에 해당하며 효율적인 관리와 유지를 필요로 한다. 이에 본 연구는 모바일 LBS 시스템에서 위상적인 네트워크 데이터를 위한 그래프 기반 지오 데이터베이스 모델 개발과 휴리스틱 접근에 기반을 둔 동적 경로 계산 알고리즘을 제시하는 것을 목적으로 한다. 이를 위해, 계층적 네트워크를 지원하는 데이터 모델을 설계하고, 모바일 LBS 시스템에서 수행 능력을 평가하기위한 경로 계산 시스템을 구현하였다. 마지막으로, 제시된 계층 그래프 모델 기반 경로 계산 알고리듬은 네트워크를 구성하는 노드 개수를 줄여 탐색 속도와 효율적 메모리 사용에 기여할 수 있을 확인 할 수 있다.

키워드 : GIS 프로그래밍, 모바일 웹 서비스, 위치기반서비스. 동적경로계획

Abstract Recently, interest in location-based service (LBS) which utilizes a DBMS in mobile system environment has been increasing, and it is expected to overcome the existing file-based system's limitation in advanced in-vehicle system by utilizing DBMS's advantages such as efficient storage, transaction management, modelling and spatial queries etc. In particular, the road network data corresponds to the most essential domain in a route planning system, which needs efficient management and maintenance. Accordingly, this study aims to develop an efficient graph-based geodata model for topological network data and to support dynamic path computation algorithm based on heuristic approach in mobile LBS system. To achieve this goal, we design a data model for supporting the hierarchy of network, and implement a path planning system to evaluate its performance in mobile LBS system. Last but not least, we find out that the designed path computation algorithm with hierarchical graph model reduced the number of nodes used for finding and improved the efficiency of memory.

Keywords : GIS Programming, Mobile Web Service, LBS, Dynamic Route Planning,

1. Introduction

The impact of tele and mobile information technology will increase the need for efficient access to geodata to be used in route planning system [3, 8]. The digital map used in existing route planning system is a static data provided as a CD-ROM or DVD, which data is periodically updated through

[†] This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF- 2009-413-D00001).

[*] Research Prof, Yong Jin Joo. Institute of Urban Science, University of Seoul, yjjoo75@uos.ac.kr(Corresponding Author)

replacement of the disk. The digital map is provided as each service provider's proprietary format (PSF : Physical Storage Format), and it is converted from raw data to the PSF at every updating period [1]. This data designed in PSF are service maps that can be accessed quickly and compressed efficiently. Since this file data management structure has great complexity in the data structure and conversion process, however, it is not suitable to process the map data varying in real time [11, 9, 10]. In particular, since the road network data in the route planning system has changes of data such as traffic congestion, average speed etc. every a few minutes, the use of spatial database is required to reflect it effectively[2]. Nevertheless, there has been few empirical research regarding spatial database model for map consistency of network topology which is capable of the supply of location-based data queries in mobile system. Therefore, this study aims to provide a comprehensive and flexible framework for a graph-based geodata model, not only considering features of the topological network data in an embedded DBMS, which is a lightweight DBMS for effective management of quite small databases contained in tiny mobile devices, but also supporting dynamic path computation algorithm based on heuristic approach in mobile LBS system.

For this purpose, we firstly analyze the KIWI format and its configurations for a clear understanding of the existing advanced in-vehicle application in the following sections. Secondly, a scheme of the road network are designed for a route planning system in an embedded DBMS. Thirdly, we designed dynamic path computation algorithm based on heuristic approach with the help of suggested physical data model in order to save memory and increase speed of path computation through minimizing search space in large-scale road. In particular, we design shortest graph models combined plain(non-hierarchial structure) with hierarchical graph model, describing dynamic scenarios to apply. Then, a route planning system is implemented to support the model designed for performance evaluation. Next, performance of path finding is measured to verify the efficiency of the data model through case study. Finally, we draw some findings in the conclusion.

2. Design of a network database model

2.1 Analysis on requirements for data structure in CNS

The KIWI data model is a standard navigation service format developed to solve lacks of data interoperability in Japan. This section would like to extract requirements for designing a network road model by analyzing data structure of KIWI and its model. The KIWI data is composed of main map data frame, route planning data frame, route guidance data frame, index data etc., which they are associated with each other by necessity. The route planning data frame displays roads of map data on the screen, and the route guidance data frame carries out path finding to inform the results to users. In addition, the index data is used to access the map data quickly. The KIWI map data is managed in a parcel unit, and organized as a hierarchical structure which consist of arbitrary rectangles called regular parcels. Figure 1 shows the parcel hierarchical structure for each level of the KIWI data[6].

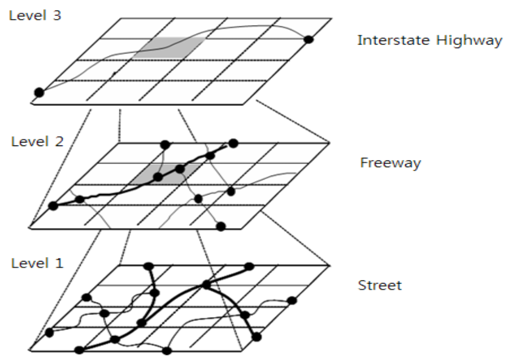


Fig. 1. Hierarchical structure of KIWI[8]

In the hierarchical structure, parcels are integrated in the higher level depending on the management unit. In other words, a parcel in the higher level corresponds multiple parcels in the lower level.

The hierarchical structure allows acquiring some range of map data corresponding to the required scale.

Consequently, considerations in the design of network road model derived from an analysis on the KIWI data model are as follows. First, three data of path indicating, finding, guiding should be organically associated with each other. The path finding is important to calculate an economical path, and should be carried out to satisfy the turn controls at intersections. In addition, geometrical information on the road network could be displayed on the screen, and users could be accurately guided to their destinations through the results of path finding. In order to display a map depending on its level (scale) and find paths quickly, a hierarchical data structure should be defined to design a schema. For this purpose, multiple levels are set up so that data of each corresponding level has different complexity. There are layers based on tiles with each other topic such as buildings, roads, and water systems etc., which a layer has several levels with different complexity of geographical data, and each level makes different layers with the same scale are managed as an integrated layer.

2.2 Logical data modeling

2.2.1 Topological model of hierarchy

Nodes and links on the road network should have a topological structure for path finding. The topological structure means relationships connected between neighboring objects, which is said relationships connected between intersections(nodes) and roads(links) in the road network. If a topological structure is made up with a structure in a large-scale road network, it is inefficient because unnecessary data is used for finding. Therefore, the topological structure of road network as a map unit should be taken into account. In addition, information on connections between maps need to be constructed to interconnect the topological structures in a map unit. This could make only the map going through from a starting point to a destination is used for finding.

If both roads with a high road class and roads with a low road class are to be used for finding in a large-scale road network, the finding speed is lowered and not only memory usages become inefficient but also quality of finding results is lowered. This is based on an empirical fact that driving via main roads with a high road class such as expressways, national highways could arrive at a destination faster than driving via local roads with a low road class for long-distance driving. Therefore, a hierarchical finding should be done by passing through links with the higher level when finding a long-distance path. For this purpose, it is needed to establish the topological structure of higher level data. Links of every level are used for road finding before entering into a road with a higher level from the starting point, and thereafter only roads with a higher level could be used for finding.

2.2.2 Storage model

This section defines a storage model to manage GIS data into a relational database and to process queries. The storage structure is based on the "SQL92" of OGC[12], which is a standard schema suggested for extending to a spatial database for relational databases. Figure 2 indicates the structure of the storage model. The table is divided into a Spatial Index (S) table, which stores information for space indexing, a Feature (F) table, which stores actual data, and an Attribute (A) table, which stores a variety of attribute information. The Feature (F) table physically stores geometrical information on

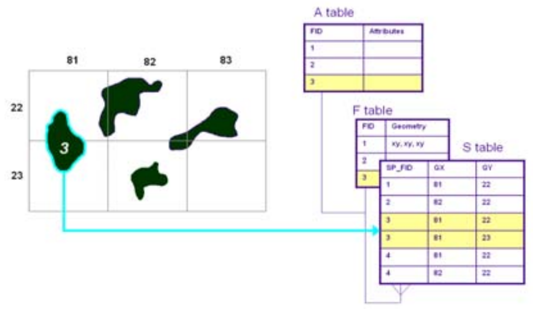


Fig. 2. Structure of the storage model

roads into binary columns. The Attribute (A) table stores attribute information such as distances, number of lanes, title of direction etc. for route guidance. Since there is an attribute in every stored geometrical object, the Feature (F) table has a one-to-one relationship with the Attribute (A) table. The Spatial Index (S) table plays a role as an index during space queries, which stores MBR (Minimum Bounding Rectangle) information, which is the minimum bounding rectangle of corresponding object, and IDs of grid indexes, which are space indexes. A one-to-many structure is used to store grid IDs into the index table of the relational access method.

2.3 Physical table scheme

Three elements of path finding, path indication, and path guidance should be involved in a data model for the road network. First, a topological structure of nodes and links, information on connected neighboring maps, and turn controls etc. should be stored in the path finding element. Second, geometrical information for indicating paths should be stored in the path indication element. Third, information for guidance such as direction names, road names etc. should be stored in the path guidance element. Therefore, the data table for road networks is divided into three elements above, and a physical model is designed by applying the storage model designed earlier as shown in Figure 3.

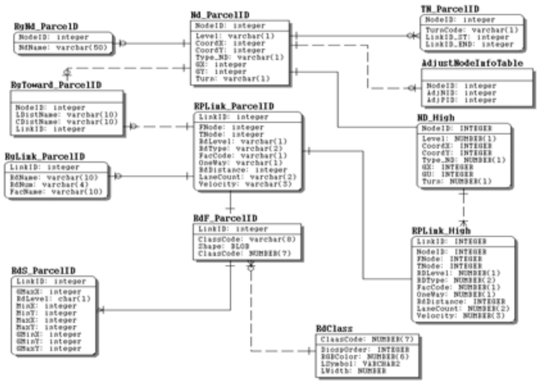


Fig. 3. Physical model of network data

2.3.1 Path finding table

The table for path finding is composed of four tables for nodes (ND), links (RPLink), information on connected neighboring maps (AdjustNodeInfoTable), and turn controls (TN). These tables are used for path finding so that all the storage type is Attribute (A). The node (ND) table stores node information, which is the basic unit for finding and guidance, and the link (RPLink) table stores link information connected to nodes. Since multiple links are connected to a node, the node (ND) table and the link (RPLink) table have a one-to-many relationship. Nodes and links form the topological relationships and each object has a unique ID with a map unit to make the path finding could be done as a map unit. The information on connected neighboring maps (AdjustNodeInfoTable) table stores connection information on neighboring nodes between maps to make the path finding could be done across the map boundaries. The turn controls (TN) table stores turn information on each node to control conditions for path finding. There may be also no turn control or several ones for certain nodes. In addition to above four tables, there are higher node (ND-High), higher link (RPLink-High) tables for finding the hierarchy. These two tables have a higher level of topological structure to make the hierarchical finding could be done.

2.3.2 Path indicating table

The table for path indication is composed of three tables such as Geometry (RdF), Index (RdS), and Indication Information (RdClass). Examining each storage type, the storage type of Geometry is Feature (F) because it stores shape coordinates of each link, Spatial Index (S) for the Index, and Attribute (A) for the class. The Geometry (RdF) table is matched to the link (RpLink) table with a one-to-one relationship and is searched using the index (RdS) table. The MBR and grid index of each link are stored in the index (RdS) table so that it is used to search a link object in certain areas. The indication order and color information are stored for each road class of each link in the indication information (RdClass).

2.3.3 Path guidance table

The table for path guidance is composed of three tables for node guidance (RgNd), link guidance (RgLink), and direction guidance (RgToward). All the storage types are Attribute (A). The node guidance (RgNd) table stores intersection names for path guidance. Since a node has no intersection name or one, the node (Nd) table and the node guidance (RgNd) table have a one-to-one or one-to-zero relationship. The link guidance (RgLink) table stores road names, number, and attribute information on facilities of roads for path guidance. The direction guidance (RgToward) table stores direction name information and has relations with the node (Nd) and link (RpLink) tables. Components of above three data models for the road network are organically operated each other. The path found through path finding elements is related with the path indication elements to display on the map. In addition, intersection names, direction names, and road facilities etc. could be guided by moving along the found path and using the path guidance elements.

3. Dynamic path computation algorithm

In this chapter, we designed dynamic path computation algorithm based on heuristic approach in order to save memory and increase speed of path computation through minimizing search space in large-scale road. In other words, Heuristic approach is the aspect point of empirical fact of arriving destination more quickly if we drive higher rank road such as highway and main road etc. rather than lower rank road in case of a long drive. So we design shortest graph models combined plain graph model with hierarchical graph model, describing dynamic scenarios. In this study, we constructed 2-level graph hierarchy in order to apply hierarchical graph model, developing topological network model in higher level through suggested physical data model. In addition, we constructed node and link structure in low level, endowed unique id number to ensure path finding every single map.

3.1 Design of shortest graph models

The most classical shortest graph model is the Dijkstra algorithm[5], whose complexity is $O(|E| + |V| \log|V|)$, where $|V|$ is the number of vertices and $|E|$ is the number of arcs. However, its performance deteriorates in terms of time efficiency when applied to large road networks, motivating several techniques for improving its response time. The A* algorithm[4], which uses a heuristic function to estimate the cost of the shortest path, is an improved version of the Dijkstra algorithm. The A* algorithm achieves acceleration by first searching the most promising nodes, and it guarantees optimal solution, provided that the cost function underestimates the actual shortest path. However, the computational effort is still quite high as the network size becomes large, making it unsuitable for real-time routing [14]. Hierarchical Strategy in order to figure out these problems has turned out to be very effective in robotic route planning and navigation system [7].

Graph model applied in this paper is extension model based on concept of HIPLA [15] which has known as a model for computing near optimal paths in large node graph as a similar algorithm as we do. That is, HIPLA(Hierarchical path planning algorithm) computes shortest path by finding among boundary nodes of the sub-graph and concatenates each route computed from edge to edge. Figure 4 illustrates path computed by HIPLA. Source sub-graph path $Ps = s \rightarrow v_1$ is concatenation of path computed from s to boundary node d_1 with edge connecting d_1 and boundary node v_1 where $d_1 \rightarrow v_1$ indicates an edge between d_1 and v1 while $s \rightarrow \rightarrow d_1$ indicates path between s and d_1.

However, this method decides node which will be surely passed in advance, performing route finding among boundary nodes. That's why this limitation

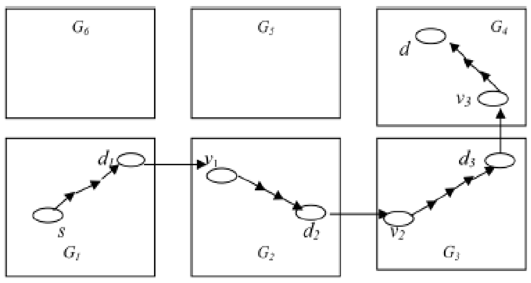


Fig. 4. Path finding by HIPLA

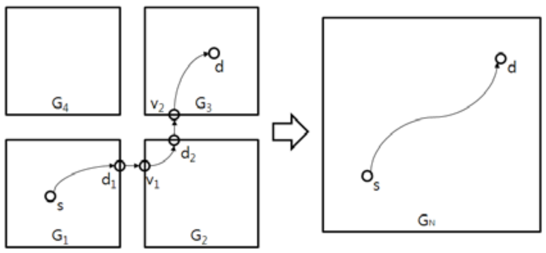


Fig. 5. Design of plain graph model

cannot always guarantee shortest path. To overcome the shortage, we improved the method which consist new graph through concatenate all sub-graph, which included in configuration of new graph lie from MBR (Minimum Boundary Rectangle) of origin and destination pair node, used in path finding as shown in Figure 5.

Let *G=(V,E)* be a graph. Then in case of $v \in V$, edge said to be (v_i, v_j). Therefore, configuration of graph described above is defined as $G_N = (E_N, V_N) =$. $G_1 + G_2 + G_3 + G_4$. In the process of merging the sub-graphs, index of vertices and edges are also merged and composed in the new graph. That is, vertex are represented by $V_N = V_1 + V_2 + V_3 + V_4$. In the new graph, connected edge between boundaries

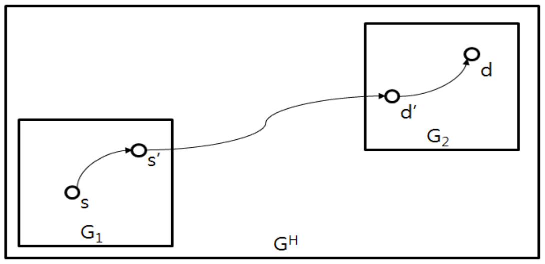


Fig. 6. Design of hierarchical graph model

```
adj_list_h = load adjacent list for high-level
function path_find(S, E)
    // path finding with the map area
    if dist(S,E) < minimum travel-distance criteria
        adj_list_se = load adjacency list from maps near S and E
        while Dijkstra(adj_list_se,S,E)=FALSE // if path finding fail
            delete adj_list_se              // extend the map area
            adj_list_se = load adjacency list extending the map area
        end while
        result_path = Dijkstra(adj_list_se, S, E)
    // path finding with hierarchical topology
    else
        adj_list_s = load adjacency list from maps near S
        adj_list_e = load adjacency list from maps near E
        S' = high-level vertex near S
        E' = high-level vertex near E
        // path finding near start point
        while Dijkstra(adj_list_s, S, S') = FALSE
            delete adj_list_s
            adj_list_s = load adjacency list extending the map area
        end while
        // path finding near finish point
        while Dijkstra(adj_list_e, E, E') = FALSE
            delete adj_list_e
            adj_list_e = load adjacency list extending the map area
        end while
        // high-level path finding
        Dijkstra(adj_list_h, S', E')
        result_path = Dijkstra(adj_list_s, S, S') +
                      Dijkstra(adj_list_e, E, E') +
                      Dijkstra(adj_list_s, S', E')
    end if
end path_find
```

Fig. 7. Pesudo code for path computation

of sub-graphs is composed. Let d_1 is a vertex of G_1 and v_1 is a vertex of G_2, if d_1 and v_1 are the same node as well as are connected with each other, edge(d_1, v_1) is newly added. That is, in the process of configuring G_N, all pairs of boundary edges $\subset E_N$ are satisfied. The method of configuring new graph described above is utilized in path finding at a short distance as shown in Figure 6.

In the hierarchical graph, let graph in higher level is called to be G^H, $G^H = (V^H, E^H)$ is defined. And each edge is represented by (v_i^H, v_j^H). In the hierarchical graph, in order to correspond from lower level to higher level, s' and d' should be determined, which is vertex of higher level in G_1 and G_2. $P_{s,d} = P_{s,s'} + P_{s',d'} + P_{d',d}$

3.2 Design of dynamic scenarios

This section designs the path finding function with the data model for road network designed earlier. The topological structure for nodes and links is stored as an adjacent list (graph data structure) for path finding. At this time, the adjacent list for higher-level nodes and links is constructed in advance as soon as the program is initiated. The reason to do so is to increase the efficiency of finding, which may become low since data of higher-level topological structure is so few. We applied Dijkstra Algorithm for path computation.

As shown in Figure 7 above, the beginning of the Pesudo code is to configure adjacent list for higher level. Path finding is divided into search for sub graph based on parcel (map) and for hierarchical graph used in higher level network. In order to apply appropriate method among two types of route planning based on distance, procedure checks out whether the distance between origin S and destination E is less than the minimum travel-distance criteria (we assume 10km based on the size of a parcel).

- In case of route planning at short distance as shown in Figure 8(a), all parcels across rectangular area including origin and destination paris are constructed as an adjacent list and path finding performs by using Dijkstra algorithm. Because there is no road which is able to connect origin and destination pairs, if it is not possible to compute path finding, re-search is performed with areas expanded through adding neighbor 4 parcels.
- When it comes to hierarchical path computation, there are a total of 3 adjacent lists

such as near origin, near destination, and high level. Then, find node S′, E′ which are the closest origin and destination pairs and find individual optimum path using Dijkstra‘s algorithm. Lastly, concatenate the final path obtained above in all as shown in Figure 8(b).

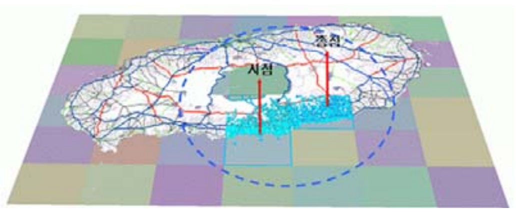
Unlike the forward path computation with a map unit (parcel or grid), one of hierarchical graph model carries out the bidirectional path finding from a lower-level to a specified higher-level. At this time, higher-level roads are used for the long-distance finding so that it makes more practical and efficient path finding could be done.

4. Case Study

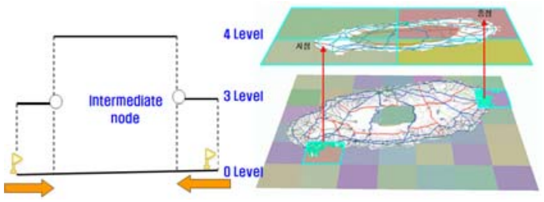
4.1 Implementation of prototype system

We developed prototype system to evaluate performance of the designed network model scheme and graph algorithm for path computation. The utility of path finding in the embedded DBMS environment would be liked to be demonstrated through an implementation of the path finding system. The implementation of the embedded spatial database in this study was designed by extending the SQLite DBMS. Data for the Jeju island was used, which was background, finding POI, and road network data comprising of six levels. The used test terminal was the HP iPAQ hx2400 (CPU 520 Mhz, 64 Mb RAM), which used 2G flash memory.

Implemented prototype system for vehicle navigation consists of three basic modules: main map display, path finding, and route guidance. Depending on distance between origin and destination paris, route planning module either the plain graph model



(a) Path computation at close distance with plain graph model



(b) Path computation at a long distance with hierarchical graph model

Fig. 8. Heuristic route model combined plain with hierarchical graph

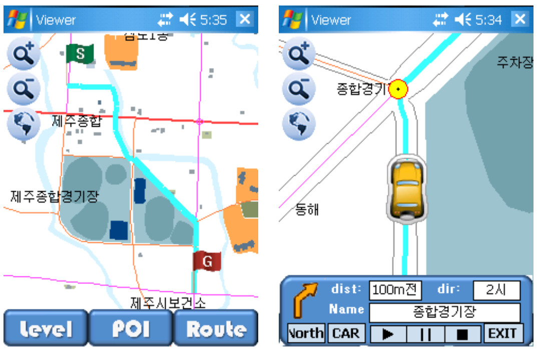


Fig. 9. The UI of Simulator : display of path computation and route guidance

or the hierarchical graph model. In addition, to display the result on the screen, search area is set up based on the current position and afterward retrieves route and its background objects in accordance with determined display order as shown Figure 9. Map display module, which consists of 6 levels, visualizes background data on the screen such as river, administrative boundary, railroad, facilities and annotation of place names etc.

4.2 Analysis on experimental evaluation

The study area for performance evaluation is Jeju island where consists of 42 parcels. The number of node, link for path finding is respectively 38,064 and 50,283. In order to verify the efficiency of suggested scheme and query processing, we evaluated algorithm in terms of computation time and accuracy. lastly, we compared the result of performance for short distance. Table1 and Table 2 show computation time on path between hierarchical and plain graph algorithm in small-scale network.

In small-scale network, plain graph algorithm is more appropriate rather than hierarchical algorithm

Table 1. Path computation at close distance with plain graph model : 1,024m

Path Computation Process		time (ms)
SL-DL	Configuration of adjacent list	525
	Path finding	8
total		533

Table 2. Path computation at close distance with hierarchical graph model : 1,229m

Path Computation Process		time (ms)
SL-DL′	Configuration of adjacent list	452
	Path finding	11
SL′-DL	Configuration of adjacent list	401
	Path finding	29
SH′ -DH′	Path finding	176
total		1,069

in terms of computation time and accuracy. As shown in Figure 10, hierarchical algorithm forced the route to circle around in order to pass through the national highways which consist of the higher level node.

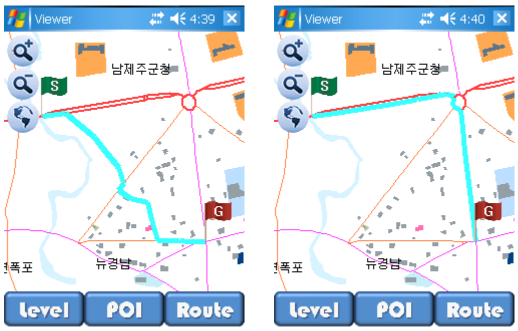


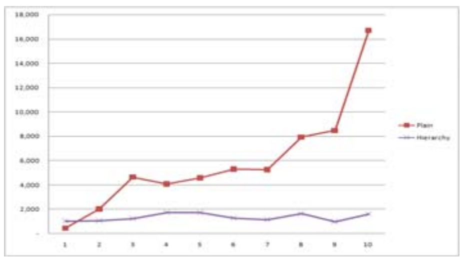
Fig. 10. Display result of Table 1(left) and Table 2(right)

Secondly, we compared performance comparison of the path computation algorithm in long-distance areas. Table 3 shows variation of computation time on path between hierarchical and plain graph algorithm in large-scale network as we gradually increase travel distance between two node.

The time found by the plain graph algorithm is considerable longer than the optimal hierarchical route. That's because the finding with hierarchical road network constructs two maps including the origin node and the destination node and the higher-level nodes, links into the adjacent list graph, the time to construct the adjacent list graph becomes shorter. In addition, it passes through the high-

Table 3. The comparison on evaluation result

Comparison of Computation Time



ID	Plain Algorithm		Hierarchical Algorithm	
	Distance (m)	Time (ms)	Distance (m)	Time (ms)
1	5,591	450	5,834	1,025
2	10,298	2,015	10,298	1,070
3	15,599	4,633	15,599	1,226
4	40,404	4,075	39,879	1,730
5	45,534	4,581	45,645	1,712
6	50,155	5,297	50,035	1,286
7	54,300	5,260	53,775	1,142
8	60,466	7,938	59,941	1,633
9	64,632	8,479	64,107	988
10	70,998	16,701	77,224	1,603

er-rank road with faster average speed in a long drive so that it provides the optimized path. Therefore, the path finding with the hierarchical road network has a good quality of paths and a very fast calculation speed in a long distance.

Consequently, the designed path finding with hierarchical graph model reduces the number of nodes used for finding and improves the efficiency of memory and finding by applying the hierarchical road level. On the other hand, it is clear that plain graph algorithm is no suitable for cases if the origin and destination pairs are far away.

5. Conclusion

The aim of this study is to develop an efficient graph-based geodata model for topological network data and to support dynamic path computation algorithm based on heuristic approach in mobile LBS system. To make it come true, we firstly review the KIWI, which is the existing advanced in-vehicle application, to grasp its configuration and requirements for developing data scheme. Next, we designed topological model of hierarchy and storage model as a logical model as well as physical table scheme such as path finding table, path indicating table, and path guidance table etc. Then, we designed dynamic path computation algorithm based on heuristic approach based on empirical fact using suggested physical data model and designed shortest graph models which is able to determine plain graph model or hierarchical graph model in accordance with the minimum travel-distance criteria. Then, We developed prototype system to evaluate performance of the designed network model scheme and graph algorithm for path computation.

In a case study, we found out that in small-scale network, plain graph algorithm is more appropriate rather than hierarchical algorithm in terms of computation time and accuracy. On the contrary, in long-distance areas. The computation time on path found by the plain graph algorithm is considerable longer than the optimal hierarchical route. That is, the path finding with the hierarchical road network has a good quality of paths and a very fast calculation speed in a long distance. Finally, we can come to the conclusion that the designed path computation algorithm with hierarchical graph model reduced the number of nodes used for finding and improved the efficiency of memory and finding by applying the hierarchical road level.

This study is expected to play a crucial role in database model for various application services in mobile devices using location-based services. A study on a synchronization method to process the real-time data is required based on this study in the future.

References

[1] E. Basiaensen, 2003 "ActMAP: real-time map updates for advanced in-vehicle applica-tions," Proceedings, 10th World Congress on ITS,

Madrid, November.

[2] M. Breunig and W. Baer, 2004, "Database support for mobile route planning systems", Computers, Environment and Urban Systems, Vol 28,No 2004, pp. 595-610.

[3] T. Brinkhoff, 1999, "Requirements of traffic telematics to spatial databases" In Lecture Notes in Computer Science: Proceedings of the 6th International Symposium on Large Spatial Databases, Hong Kong, China, Vol 1651, pp. 365 - 369.

[4] I. Chabini and S. Lan, 2002, "Adaptations of the A*. algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks," IEEE Trans. Intell. Transp. Syst., Vol. 3, No. 1, pp. 60-74.

[5] E. W. Dijkstra. 1959, "A note on two problems in connexion with graphs," Vol.1, No.1, pp 269-271.

[6] H. FUJIMOTO, 2001, "World Wide Vehicle Navigation System Using KIWI Format ", デンソ. テクニカルレビュ. Vol. 6 No.1.

[7] J. A. Fernandez-Madrigal and J. Gonzalez, 2002, "Multihierarchical graph search," IEEE Trans. Pattern Anal. Mach. Intell., Vol.24, No.1, pp 103-113.

[8] T. Hamada, 2002, Kiwi Format and Telematics.

[9] Y. J. Joo and S. H. Park, 2006, "Design and Implementation of Map Databases for Telematics and Car Navigation Systems using an Embedded DBMS," The Journal of GIS Association of Korea, Vol 14, No. 4.

[10] Y. J. Joo and S. H. Kim, 2011, "A New Route Guidance Method Considering Pedestrian Level of Service using Multi-Criteria Decision Making Technique," Journal of Korea Spatial Information Society, Vol. 19, No.1, pp. 91-99.

[11] G. G. Moon , Y. J. Joo, S. H. Park, 2011, "Lossless Vector Data Compression Using the Hybrid Approach of BytePacking and Lempel-Ziv in Embedded DBMS," Journal of Korea Spatial Information Society, Vol. 19, No.1 ,pp.115-124.

[12] Open Geospatial Consotium, Inc., 2005, Simple Features - SQL 1.1

[13] H. J. Sim, J. J. Kim, I. S. Shin and K. J. Han, 2008, "Embedded spatio-temporal DBMS for mobile device", Spring Conference on GIS Association of Korea, pp. 59-66.

[14] Q. Song and X. F. Wang, 2011, "Efficient Routing on Large Road Networks Using Hierarchical Communities," IEEE Trans. intel. Transp. Syst., Vol. 12, No. 1.

[15] R. Rajagopalan , K. G. Mehrotra, C. K. Mohan and P. K. Varshney, 2008, "Hierarchical path computation approach for large graphs," IEEE Trans. Aerosp. Electron. Syst., Vol. 44, No. 2, pp 427-440.

[16] A. Zipf and J. Strob, 2002, Geoinformation mobil. Heidelberg, Germany: Herbert Wichmann, pp. 230.

Manuscript received : 2011.05.03
Revised : 2011.06.15
Accepted : 2011.06.29

Yong Jin Joo

2001 Dept. of GeoInformatic Engineering, Inha University (B.S.)
2003 Dept. of GeoInformatic Engineering, Inha University(M.S.)
2009 Dept. of GeoInformatic Engineering, Inha University (Ph.D.)
2009~Present Research Professor, Institute of Urban Science, University of Seoul
Research Expertise : Spatial DBMS, LBS & Embedded System. Spatial Reasoning & Ontology, Urban Growth Simulation Model