A MULTI-LEVEL BASED REAL TIME TRAFFIC INFORMATION STORAGE MODEL FOR MOBILE LBS

LI Kai*, ZHONG Er-shun[a]

[a]Institute of Geographical Sciences and Natural Resources Research, Chinese Academy of Sciences, Beijing, 100101, China - (likai, zhonges)@supermap.com
[b]Graduate of School of the Chinese Academy of Sciences, Beijing, 100049, China

ThS-8: Location Based Service (LBS)

KEY WORDS: Mobile GIS, Navigation, Data Organization, Dynamic Modelling, Multi-Scale Representation

ABSTRACT:

The fundamental feature of the next generation LBS systems is supporting the Real-time Traffic Information (RTTI). However, in order to speed up the data accessing, The Physical Storage Format (PSF) of digital map on mobile device is distinguished from that of the T-GIS in database. This difference raises some issues when storing the RTTI in the mobile client of LBS systems. This study focuses on the model which combines traffic information with hierarchal network model which is popular in mobile navigation systems. We proposed a new Multi-Level model (DyHiRD) storing the RTTI based on the Hierarchical Linear Link Coding algorithm (HLLC). The characteristic of DyHiRD is that hold the consistency of RTTI in levels after RTTI updating. The experiment is taken Beijing's road network as example data, and then the client system update local data once received traffic data broadcast by the traffic center. The result gives a proof of that DyHiRD is high efficiency in updating performance and compressing data size. Therefore DyHiRD has capability to satisfy the requirement of dynamic navigation application which supports the real time traffic information of large city.

1. INTRODUCTION

The Linear Reference System and Dynamic Segment (LRS&DS) is a popular model for organizing traffic information of geographic network. But it's hard to apply LRS&DS to mobile GIS system. The main reason is the redundancy of Hierarchical Network Topology (HiTopo) model which is effective on improving memory size and speedup path analysis (NAVTECH, 1999; Kiwi-Forum, 2000; PSI, 2007). Although HiTopo is so efficient, it has a significant drawback that is inconvenient for data updating. In order to keep the updating operation's consistency, there are two kinds of relationship needed considering in storing RTTI based on HiTopo. Firstly a high-level travel link is constructed by many road segments which are defined as road element, so a containing relationship from each road segment to its corresponding travel links should be saved. Secondly a RTTI record may mapping to many road segments, so a connected relationship from the road segment to its previous and to its next also should be saved. One task of this study is that design a high performance storage model which is capable of recording above two relationships while takes less disk space

This study combines LRS&DS with HiTopo when designing the Dynamic Hierarchal Road Model (DyHiRD). DyHiRD is based on the Hierarchical Linear Link Coding algorithm (HLLC). The link code calculated by HLLC is proved to be implicating the containing and referenced relationship. It is more important that each link code take 8 bytes, and so DyHiRD is an ideal model for record RTTI in mobile system. In our experiment, we take Beijing roads as sample data. The device receives RTTI which broadcast by Beijing Traffic Manage Bureau through GPRS every 5 minutes, and then update local data. As the result showing, the peak value of time cost for updating is less than 1 second per 1,000 RTTI record. The practicability of DyHiRD completely meets the requirement of dynamic navigation application in large city.

After making a survey on the related research work about RTTI model in section 2, we conduct the framework and restriction of DyHiRD in section 3. Then, section 4 introduces the implements of DyHiRD. In the following section, we present experimental results to verify the updating performance of DyHiRD.

2. RELATED STUDY

The LRS & DS is a fundamental model to Intelligence Traffic System (ITS). The LRS & DS locates a traffic event by the measurement from a reference point to the event. Its advantage is less redundancy and higher stability. There are two key techniques to implement LRS & DS including how to identify link and how to locate reference point. The natural increasing link IDs are inconvenient for model's updating and sharing (Dueker K J, 1998). There are many methods to plan reference point which were given a description in (Scarponcini, 2002)Then (Koncz N , 2002; Guo and Kurt, 2004)proposed two new complex LRS & DS which take more factor into consideration. Based on above researches, some practical models have been suggested. NCHRP2 0-27(2) (Vonderohe, 1995)is a classic logical model, but its universality lead the complexity. Therefore, (Dueker K J , 1998; Scarponcini, 2002) respectively proposed a General model and Enterprising model to simplify NCHRP2 0-27(2). Then ISO also suggested a serial of relative models for LRS (ISO, 2003; ISO, 2004; ISO, 2007). In a word, all of above works supposed the road network as flat topology. While the HiTopo model is widely applied in mobile system, it's hard to apply those models directly. If it's supposed that each road segment records the relative road segment's and

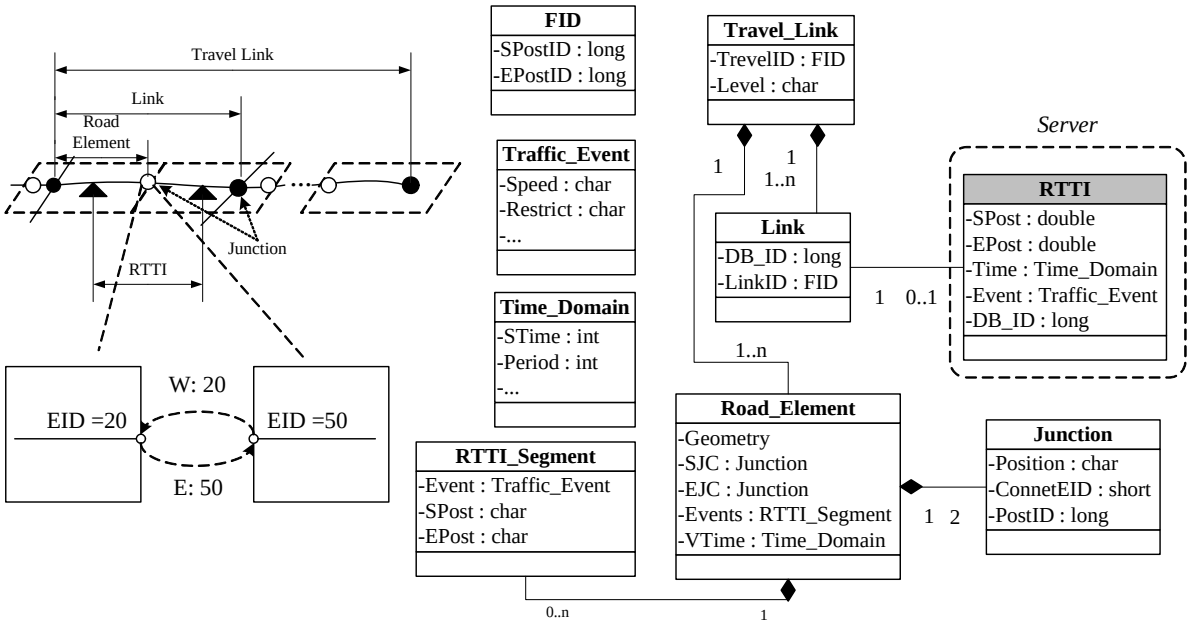

Figure 1 Logic framework of DyHiRD

Feature name	Definition
Road_Element	the part of road between two junction.
Junction	the cross of road elements or the cross of grid boundary and road
Travel_Link	a set of connected road elements
Link	the part of road between two road crossing
Event_Segment	the traffic event on road, using start position and end position to represent its location
FID	The identifier of object in DyHiRD

Table 1 Model element define of DyHiRD

travel link's ID to guarantee consistent updating, it would cost a large amount of disk space. In the following section, this paper will descript the solution of this topic.

3. DYHIRD LOGICAL MODEL

3.1 Framework of DyHiRD

Figure 1 is the logical model of DyHiRD and Tab1e 1 explains the definition of elements in Figure1. DyHiRD is extended from the node-link topology. Besides considering RTTI storage, DyHiRD also support the application of the path analysis and map display. DyHiRD has some characters descript below:

Firstly, DyHiRD is oriented to the hierarchical storage structure. The level order of element is record in the property of Travel_Link::Level. In practice, the algorithm of path analysis used Travel_Link as short-cut-link which connects two nodes at a long distance in order to compress the searched space. A Road_Element contained by Travel_Link records a part of geometry information of Travel_Link. Travel_Link in different level contains geometry data of road with different abstraction, which speeds up the map display in small scale.

Secondly, DyHiRD records the RTTI mapping relationship from central database to client PSF. There are two kinds of link identifier: DB_ID is an identifier of link in the central database, while LinkID whose class is FID is the corresponding identifier of link travel in the client PSF. Therefore once the RTTI is received, the client system mappings DB_ID into LinkID to look up the travel link for updating. Then it locates the related road elements by RTTI's SPost and EPost, and then traffic event is refreshed.

Thirdly, DyHiRD supports Grid File model which is widely used in mobile GIS. The related direction from grid which contains the road element to the grid which contains the road element's previous (or its next) is recorded in its start junction (or its end junction). By using the direction and the record index of the connected road element in the neighbouring grid, all road elements belong to the same travel link is connected into a list. Therefore all road segments affected by the same RTTI are able to be updated consistently.

Finally, DyHiRD is hierarchically consistent. For explaining this property, we give the definition of containing relationship. i is meant level index of network, T is meant travel link, $\subseteq$ is meant containing relationship.

$$\forall T_1 \subseteq T_2,\ satisfy: \quad SPostID_{T1} \geq SPostID_{T2},\ EPostID_{T1} \leq EPostID_{T2} \qquad (1)$$

Network is hierarchically consistent that

$$\forall T_i, \exists T_{i-1}, Then\ \ T_{i-1} \subseteq T_i, (i>1) \qquad (2)$$

Each travel link of DyHiRD has a unique identifier, TravelID, whose class is FID. FID is composed of SPostID and EPostID. SPostID is present the linear measurement of start junction while EPostID is present the linear measurement of end junction. Because the measurement of reference point in network is assigned from high-level to low-level, every travel link in higher level must contain the sub-travel link in lower level. Then DyHiRD satisfy define 2.

4. IMPLEMENT

4.1 Hierarchal Liner Link Code

The process of coding is described as below. After links in the detail layer are partitioned into a set of road elements (RD) by grids. Then the post of junction is assigned separately according to the level of link. The TravelID is composed of the post of start and end junctions of travel link. In the algorithm descript in Table 2, TraceForward and TraceBackward are similar. They set r as root, then do a deep first searching along (or against) the direction of traffic. The terminal condition of search is that the next node is not a procedo-node. Measure is meant the method to calculate the cost of link. The computing complex of HLLC is O (k•n) K is the number of levels.

4.2 Partition storage of DyHiRD

Partition storage is an effective strategy for memory control in mobile client system. There is a restriction for DyHiRD that the grids of higher level partition must completely overlap the grids of lower level partition, crossing is not allowed (see Figure 2). If not, the partition will insert a new boundary junction then destroy the LRS defined by HLLC (see Table 2). We proposed a partition algorithm which based on Q-Tree in Table 3. The DB is meant the original data. Because the algorithm maintenance strict overlapping relationship of multilevel grids, SavePage save network by querying road elements within the Bounds while avoid geometrical partition.

```
FUNCTION Coding（RD）
（1）LM = 0;                                    // initialize the variable of code
（2）FOR i = MAX_LEVEL TO 1     // coding from high level to low level
（3）        FOR EACH r IN RD                   // get a road element from RD
（4）            Travel={ r };                  // initialize a Travel link
（3）            IF  Level (r) < i OR Level (r)== i AND HasCode (r) THEN
                                CONTINUE;      // if r has been coded or r is lower than current level, then continue
（4）            TraceForward (r，i，Travel);   //Trace for the travel link after r along traffic flow
（5）            TraceBackward (r，i，Travel);  //Trace for the travel link previous r against traffic flow
（6）            IF  HasCode (r)  THEN          //If r have been code in upper level
（7）                    Travel.SPostID = Travel[0].SJC.PostID;     //assign SPostID with start junction post
                        Travel.EPostID = Travel[n].EJC.PostID;     // assign EPostID with end junction post
（8）            ELSE
（9）                    Travel.TravelID.SPostID = LM               //assign SPostID with current measurement
（10）                   FOR EACH s IN Travel                       //assign PostID of road element iteratively
（11）                           s.SJC.PostID = LM;
（12）                           LM += Measure(LM);                         //add up measurement
（13）                           s.EJC.PostID = LM;
                                NEXT s
（14）                   Travel.TravelID.EPostID = LM               // assign EPostID with current measurement
                        END IF
            NEXT r
        NEXT i
```

Table 2 Hierarchal coding algorithm

4.3 Linking road element

For maintenance the referenced relationship between road elements of the same travel link, the road elements of the same travel link are constructed a list. Junction::Position and Junction::ConnectEID is used for linking the next segment in different grid. Junction::Position is meant the direction of adjacent grid related current grid. Junction::ConnectEID is meant the record index of next road element in related page. Above attributions of each road element are saved when grid partition. Then the attributions' value of SJC (or EJC) is updated according the attributions' value of previous (or next) road element.

4.4 RTTI updating

The RTTI updating process is divided into two steps, the first step is located the road elements and the second step is refresh the traffic events on the target road elements. In order to speed up updating process, we build an index for pointing to every travel link of each level with taking TravelID as primary key. The leaf of above index points to the head road element of link travel. GetNextRDElem is used for locating the next road element. If the next element is not in current grid, GetNextRDElem can get the neighboring grid using boundary junction's Position and the edge's coordinate of the grid. Then it looks up the adjacent element by ConnectEID which is also recorded in the junction.

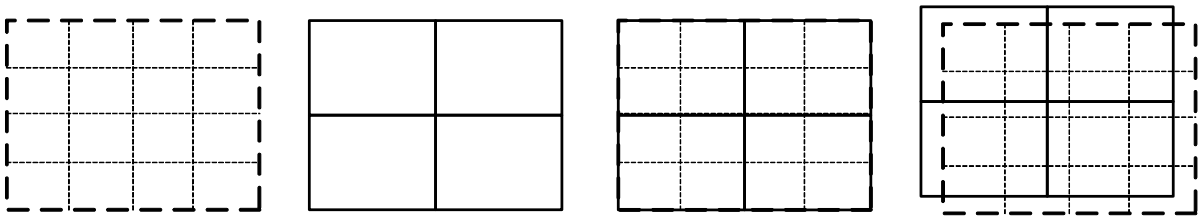

Figure 2 Restriction of DyHiRD's grid partition

```
FUNCTION Partition (Bounds, DB，Level)
（1）IF SavePage (DB, Bounds) < MAX_PAGESIZE THEN
            Return；                          //if the data size is limited in max page size , save page
and return
（2）SubRecLT = Quari(Bounds,0);    // left-top partition
            Partition (SubRecLT , DB, Level) ;
（3）SubRecRT = Quadr(Bounds,1)    // right-top partition
            Partition (SubRecRT , DB);
（4）SubRecLB = Quadr (Bounds,2)   // left -bottom partition
            Partition (SubRecLB , DB);
（5）SubRecRB = Quadr (Bounds,3)   // right-bottom partition
            Partition (SubRecRB , DB)
```

Table 3 Region partition algorithm

```
FUNCTION Update ( RTTI )
（1）FOR i = MAX_LEVEL TO 1                          // Update from higher level to low level
（2）        RDElem = GetRDElem ( RTTI.DB_ID,i );  // Read first road element
（3）        SPostID = RDElem. SJC.PostID + Measure (RTTID.SPost); //calculate the event start position
（4）        EPostID = RDElem. SJC.PostID + Measure (RTTID.EPost); //calculate the event end position
（5）        WHILE (RDElem IS NULL)                  // iterate each element to match targets.
（6）            IF RDElem. SJC.PostID <= EPostID AND RDElem. EJC.PostID >= SPostID THEN
（7）                    RTTI_Segment.SPost = MAX (RDElem. SJC.PostID, SPostID);
（8）                    RTTI_Segment.EPost = MIN (RDElem. EJC.PostID, EPostID);
（9）                    SetSegment(RDElem, RTTI_Segment, RTTI, i)   //refresh RTTI_Segment
（10）           ELSE IF RDElem. SJC.PostID > EPostID  THEN
                            BREAK;
                        END IF
（11）           RDElem = GetNextRDElem(RDElem);          //get next the road element
            END WHILE
    NEXT  i
```

Table 4 RTTI update algorithm

5. EXPERIMENT

We did the experiment according to above algorithms to validate DyHiRD's performance. The example data includes all the roads of Beijing city, total is 123, 968 links. Other detail information of test data is showed in Table 5. The network is made up 4 levels. It cost 15 minutes to build the model. The data size is about 10MB. The test environment is described as below: CPU is ARM 500MHZ, memory is 64MB and operation system is Linux 2.6.24.

We update the RTTI which is downloaded from the Traffic Center of Beijing every 5 minutes with GPRS. The peak number of RTTI is above 4000, Figure 3 shows Traffic information display of multi-level after updating completing. Besides using No-strategy, we also optimize updating algorithm by time-strategy (exclude the link out of date), event-strategy (exclude the link's status unchanged) and competitive strategy. From Figure 4, the actual number of updating in above strategy is greatly cut off.

Figure 3 Traffic information display of multi-level

Level	Size of Road	Size of Grid	Data Size（byte）
1	102649	343	9143930
2	7942	162	854200
3	8416	36	395126
4	4961	16	148860

Table 5 Sample data detail information

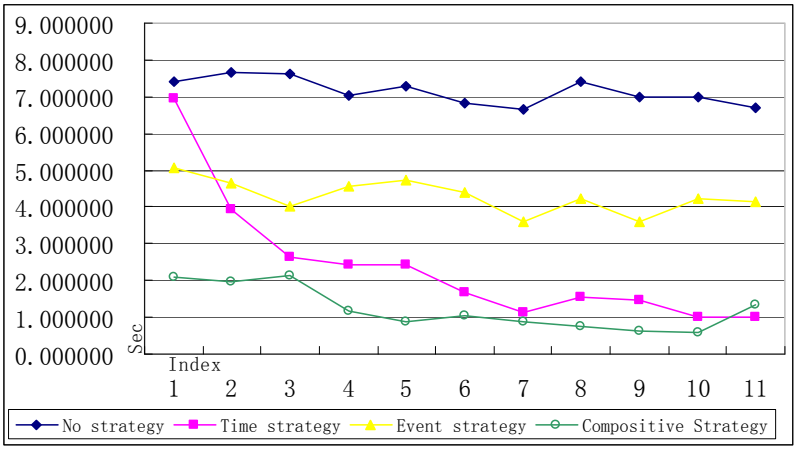

Figure 4 Time cost of each update

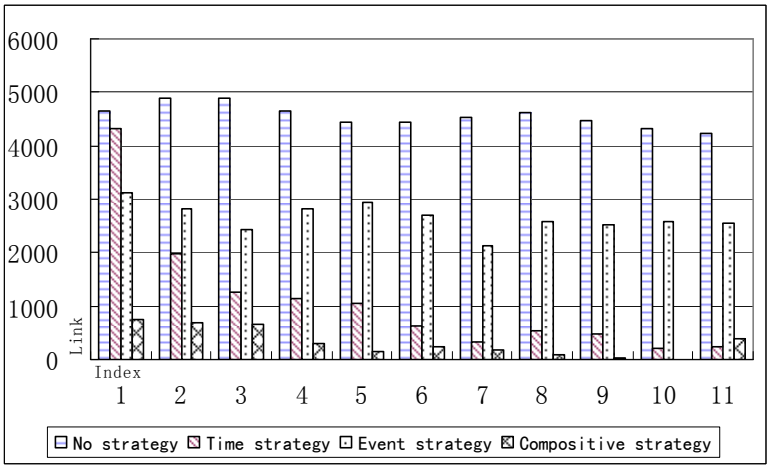

Figure 5 Link number of each update

6. CONCLUDE

This study suggests a new multi-level and partition based RTTI storage model DyHiRD. DyHiRD solves the problem of maintaining the consistency when updating RTTI in multi-level network model. As the experimental result showing, DyHiRD is a high-performance model which is able to satisfy the requirement of dynamic LBS in large city.

REFERENCES

Dueker K J , B. J. A. (1998). "GIS-T Enterprise Data Model with Suggested Implementation Choices" *Journal of the Urban and Regional Information Systems Association* 10 (1): 12-36.

Guo, B. and C. E. Kurt (2004). "Towards temporal dynamic segmentation." *GeoInformatica* 8 (3): 265-283.
ISO (2003). Traffic and Travel Information (TTI)-TTI messages via traffic message coding-Location referencing for ALERT-C (Part 3). 14819. ISO.

ISO (2004). Intelligent transport system-Geographic Data Files(GDF)-overall data specification. 14825. ISO.

ISO (2007). Geographic information - Location-based services-Multimodal routing and navigation. 19134. TC211. ISO.

Kiwi-Forum (2000). Kiwi Format Specification version 1.2.2.
Koncz N, A. T. M. (2002). "A Data Model for Multi-dimensional Transportation Location Referencing System." URISURISA 14 (2): 27-41.

NAVTECH (1999). NAVTECH PSF Specification for SDAL Format version 1.7.

PSI (2007). PSF Standardisation Initiative web site: http://www.psf-initiative.com/.

Scarponcini, P. (2002). "Generalized model for linear referencing in transportation." *GeoInformatica* 6 (1): 35-55.

Vonderohe, A. P. (1995). Results of a Workshop on A Generic Data Model for Linear Referencing Systems. Wisconsin Madison, University of Wisconsin Madison. Ph.D.