A MULTI-SCALE LINE STRUCTURE FOR PROGRESSIVE TRANSMISSION OF VECTOR DATA ACROSS THE WEB

REN Yingchao[a, *], ZHU Lin[b], YANG Chongjun[a]

[a]The State Key Laboratory of Remote Sensing Information Sciences, IRSA, CAS, Beijing, 100101
[b]College of Resources Environment and Tourism, Capital Normal University, Beijing, 100037, China - yingchaoren@gmail.com

Youth Forum

KEY WORDS: Multi-scale Line, Simplification, Multi-way Tree, Progressive Transmission

ABSTRACT:

This paper presents a multi-way tree based multi-scale line structure for progressive transmission of vector data. By topological restraints, the structure maintains the consistent topological relations among the lines of different scales. By integrating the data of the same scale into one certain level node, the structure provides more efficient hierarchical data management. And the order of the points in the original line is described by the relative order of points in the multi-scale line tree, so that we can restore the line in any scale, which also supports the modification of multi-scale line without rebuilding of the multi-scale structure. Meanwhile, the data in different scale are independent of each other, so that we can store the data in different scale into different tables in database, which will improve the first access efficiency.

1. INTRODUCTION

Geographic space is a complicated system with geographic elements with self-similarity in different scales interacting with each other. Geographic space in different scales represents different geographic phenomena and geographic laws. In the era of paper maps, people produced a series of maps with different map scales to reflect different geographic phenomena in different scales.

Geographic Information System (GIS) is an electronic tool to manage maps. The concept of map scale in traditional map has also changed. The functions of zoom in and zoom out make maps be able to be displayed in any map scales. However, traditional GIS can only manage maps with a single scale. So zoom in operation doesn't bring into the increasing of map information, while zoom out operation makes map information be too dense to be discerned so as to make it hard for users to get main information in the map. Secondly, operations of zoom in and zoom out are just to make maps with a single map scale to be displayed with different view scales which cannot reflect geographic phenomena in different scales in nature.

The data volume of geographical data is usually massive. Although the bandwidth of the network has got remarkable improved, the rapid growth of spatial data and the users' demands to the high resolution spatial data soon exhausts the limitedly improved bandwidth, which situation is more serious in wireless network. Nowadays, the limited bandwidth of the network has become an important bottleneck for the web issuing and access of the massive, high resolution spatial data across the web, which appears more serious to mobile devices based on the wireless network, such as cell phones, PDAs, on which a great deal of detailed data in the high resolution spatial data cannot be displayed, while still occupy bandwidth to download.

Progressive transmission of vector data provides a promising solution to that problem. The skeleton data with low resolution are transmitted to the client at first, then by users' request the detailed increment data are transmitted to the client and integrate with the existing data on the client to restore the high resolution data, so as to reduce the redundant data transmission, improve the utilization of the bandwidth while preserving the precision of the original data.

The key to the progressive transmission of vector data is to generate the multi-scale representation of spatial data. The ideal way is just to maintain a single base map with very large map scale. By user's demands, maps with small scales are generated from that base map on the fly. The abstract of map is a process of map generalization. Map generalization should preserve topological and geometric consistency among maps with different scales. So map generalization has to deal with complicated and time-consuming geometric computation, and meanwhile there exist a great deal of subjective factors which are hard to be regularized. Till now, the efficiency and the results of map generalization still cannot satisfy the demands of on-the-fly map generalization. Anther way is to maintain a multi-version map database. Similar to traditional paper maps, the multi-version technology integrates a series of maps with different map scales together. By users' requests for different map scales, the system accesses maps with corresponding scale and returns them to the users, so that users can get maps with corresponding map scales in different view scales. The multi-version technology satisfies the demands of users for maps with different scales. However, it also brings some new problems. Firstly, the system has to maintain maps with different scales, which heavies the burdens of system. Secondly, data update

Supported by the 863 Program of China (Grant No. 2006AA12Z208) and the CAS Innovation Program (Grant No. KZCX2-YW-304-02).

brings problems of maintenance of integrity and consistency of maps with different map scales. Thirdly, there inevitably exists redundancy among maps with different scales. Massive spatial data make redundancy great. And last, the multi-version technology can only provide maps with maps with several predefined scales. It cannot generate maps with any scale. A compromising way is to pre-process the map with large map scale by map generalization methods to generate a series of maps with different map scales. The results are managed in a hierarchical data structure and build vertical indexes among maps with different map scales. For the one hand, this method gets the high access efficiency as that in multi-version map database, for another hand, it also avoids the problems of maintenance of data integrity and reduces the redundancy.

There are two kinds of multi-scale data structures. One is object collection hierarchical structure and another one is object detail hierarchical structure. The object collection hierarchical structure is on the spatial object level, which is corresponding to selection and merging operators in map generation. The spatial object appears or doesn't appear in some hierarchy of the tree according the weight of the spatial object in the spatial object collection. This kind of data structure includes Reactive tree, GAP tree etc. spatial objects own inner structure and the details of the spatial objects varies according to the map scales. Object detail hierarchical structures represent the degree of the detail of line objects in different scales. Object detail hierarchical structures simplify the line in different scales, and stores the details structures into multi-scale structures like Strip tree[2], BLG tree, etc. However, both Strip-tree and BLG-tree are binary trees. The details with the same scales scatter in the different levels of the trees. So the paper design a new hierarchical structure, named multi-scale line tree (MSLT) [3], to mange multi-scale line generates by line simplification algorithms.

The details with same scale lie in the same tree hierarchy. Only the complete coarse skeleton line is stored in the first hierarchy of the MSLT. And only increment data are stored in the other levels of the tree. So the MSLT also supports progressive transmission of vector data across Web. When the user request line data with finer scales, the system only need to transmit increment data and integrate with the existing data in the client to restore the complete line in the finer scale So as to avoiding repeating transmission. It's very useful in network environment with limited bandwidth.

2. GENERALIZATION ALGORITHM OF THE MULTI-SCALE LINES

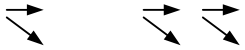
2.1 Generalization algorithm of the multi-scale lines

A geographical line is a complicated entity made up of lines with different resolution, in which the line with high resolution contains the information in the line with low resolution. Therefore, a line can be represented as a coarse skeleton and a series of details with different resolution.

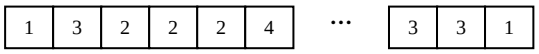
$$C = C_1 \oplus D_1 \oplus \cdots \oplus D_n$$

The generalization of multi-scale is a process to iteratedly simplify details with different resolution. There are many algorithms in map generalization to simplify lines, among which Visvalingam-Whyatt(VW) is an algorithm to simplify lines from bottom to top. At first, the algorithm eliminates the most detailed, namely the least important, vertices from the original line, and then iteratedly eliminate the most unimportand points from the current line. The VW algorithm is a progressive line simplification in according with the congition of human being in the zoom in and zoom out of the map. In this paper, we use VW algorithm to generate the multi-scale structure of the lines.

At, first, a threshold list $\{\varepsilon \mid \varepsilon i,\ \varepsilon_i < \varepsilon_{i+1}\}$ under different scales is predefined and they are applied into the line simplification in turn from beginning of the smallest one. At the beginning, ε_n is used to simplify the line C and get the result line Cn and the detailed increment data Dn. Then simplify Cn with threshold ε_{n-1}. Iteratedly simplify the line with threshold in the list until reaching ε_1, and we can get a skeleton line C_1 of C and a series of increment data D_i according to threshold ε_i.



According to the level of each vertex in the line in the threshold list, we assign a *level id* for each vertex. And we define the level id of the corasest skeleton is *1*, and that in the most detailed level is *n+1*. based on the level id of the vertices, we can construct the hierarchical structure of the line.



2.2 Topological consistency in line simplification

VW algorithm does not take topological relationships into account. So after simplification, lines may intersect themselves or other lines. For example, two lines separating from each other may be intersected.

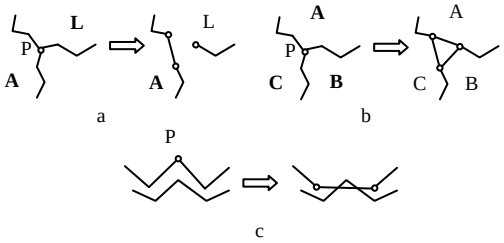


Figure 1Topological inconsistency

Line simplification is a process of vertex elimination, so topological inconsistency comes from wrong vertex elimination (Fig 1). There are two kinds of vertices in the line, the vertices which can be eliminated and the vertices which cannot be eliminated. The first kind of vertices can just affect the precision of the line, while the second kind of vertices may cause inconsistency in topological relationships.

Uneliminatable vertices can be summarized into following two categories:

1) endpoints of the arc,
2) inner vertices of the arc.

The endpoints of the arc are key vertices in topological relationship among spatial entities (Fig1 a, b). We can know such vertices before line simplification. When we build topological relationships on the spatial data without topological relationships, the endpoints of arcs are this kind of uneliminatable vertices.

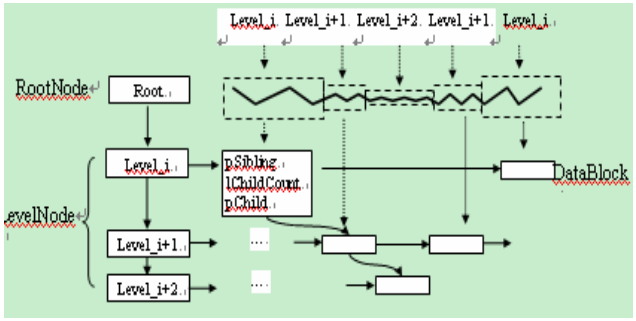


Figure 2 . Multi-scale line structure

The inner vertices may cause the simplified lines to intersect themselves or other spatial entities (Fig 1 c). Such vertices can only be dynamically judged in the process of line simplification. The regulation is that when there is a vertex p falling into the triangle made up of vertex p_i and his previous vertex $p_{i-1}$and next vertex p_{i+1}, the elimination of p_i is doomed to intersect with himself or other lines, so vertex p_i is a uneliminatable vertex.

By the constraints of two kinds of uneliminatable vertices, we can preserve the consistency of the topological relationships after line simplification.

3. MULTI-SCALE LINE MODEL

3.1 Multi-scale line model

After iterated line simplification, we classify the vertices in the line into different scale levels. Now we can build a multi-scale line structure to preserve the simplification result, so that when users access the multi-scale line, we just need to traverse the multi-scale line structure rather than simplify the line on the fly.

The multi-scale line model is a tree structure, made up of three kinds of nodes, the root node, level nodes and data block nodes. Fig2 shows the hierarchical structure.

I) ***Root Node***. Root node is the entrance of the multi-scale line mode. It encapsulates the metadata of the line, such as id of the line, the Minimum Bounding Box, the levels of the hierarchy, and the access methods of the model. When users request data from the model, the Root Node locate to the related scale level according the scale information from users and call related methods to generate the line or increment data according to the scale.

II) ***Level Node***. Level Nodes consist of main frame of the multi-scale line model. All vertices with the same scale are aggregated into the same level node. Compared with Strip tree and BLG tree, the level in the model of the paper appears more clear and easily to be extended.

The level node is defined as following in C++:

```
struct ScaleNode

{
    float       fScale
    long        lLevelCount;;
    DataBlock *pBlockList;
};
```

fScale is the scale of the current level node. When users request data, the model determines which level should be visited according to fScale.

lLevelCount describes how many levels the model has.
pBlockList points a data node list.

In the multi-scale line mode, only the first level preserves the complete coarse line, while rest levels just preserve the increment data to the neighbouring previous level, which reduce the data redundant data and is easy to be maintained.
III) ***DataBlock Node***. Neighbouring vertices in the line with the same scale make up of a DataBlock Node. As Fig 2 shows, the first 4 vertices and the last 3 vertices make up of two data node respectively.

The DataBlock Node is defined in C++ as following:

```
struct DataBlock
{
    FPOINT    *pPoints;
    long        lPointCount;
    DataBlock *pSibling;
    DataBlock  *pChild;
    long        lChildCount;
};
```

pPoints recorods the coordinates of the vertices in the line.
lPointCount describes how many vertices in the Data Block Node.

pSibling points next neighbouring Data Block Node in the same level.

pChild points the DataBlock Node of next level just after current DataBlock Node. And lChildCount describe how DataBlockNodes of next level from current DataBlockNode to his sibling DataBlock Node, which are used to build vertical index among levels of the multi-scale line model.
The resolution of the vertices between two neighbouring DataBlockNodes are higher than those in these two DataBlockNodes, therefore they are simplified into a line segment under current scale.

3.2 Vertical index

We have built the tree based hierarchical structure of the multi-scale line. And we need to know how the vertices between neighbouring levels are interrelated with each other, so as to rebuild the original line structure level by level. The vertical

index provides us a way to interrelate the vertices between neighbouring levels.

As Fig 3 shows, a line C on the *Level_i+1* is simplified into a line segment S in Level_i. v_s and v_e are two endpoints of S. Therefore, if inserting the eliminated vertices on the detail level between v_s and v_e, we can restore the line *C* in *Level_i+1*. The pChild of BlockS points to the DataBlockNode on the *Level_i+1* with v_s as his previous vertex, and lChildCount records how many DataBlocks there are between v_s and v_e on the *Level_i+1*. And now, we build the vertical index of the multi-scale line mode. By traversing vertical index, we can get increment data of the line on any scales, and restore the line on any specified scales.

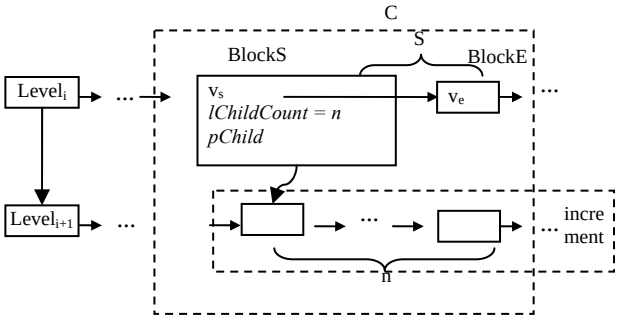


Figure 3 Vertical Index

4. OPERATION ON MULTI-SCALE LINE

4.1 Generate line in the specified scale

When users request the line with coarsest level, we just need to return them the line on the first level.

When users request line in the level i, we need to traverse the multi-scale line mode with depth first. At first, the model determines the scale level nLevel according to the users' request, and that is the highest level the model traverse. The algorithm first visit the first DataBlock dataBlock on the top level of the multi-scale line model. If there is the child node in dataBlock, the algorithom turns to the child DataBlock node referenced by *pChild* to get detail data of the line and iteratedly visit the child node until reach level i. Or from the DataBlock referenced by *pChild*, the algorithm visits his *lChildCount* sibling nodes. According to the visiting order, append the vertices to a vertex array to generate the line C under the scale level i.

4.2 Increment data

Increment data are the vertices eliminated when line is simplified from scalei to scalej. Increment data not only express the detail data, but also the inserting position of the detail data in the original line, so that we can restore original data from increment data.
The structure of the increment data is defined as following:
struct IncrementDataSet

```
{
    double      dScale[2]
    GFPoints    Points;
    GNumbers Offsets;
    GNumbers Anchors;
};
```

dScale is an array with two elements, scalei and scalej.
Points records the coordinates of the increment vertices.
After simplification, the line may be simplified into several sub lines. As Fig 3 shows, the increment data between Level i+1 and Level i are made up of two sub lines.

Offsets records the number of vertices in each sub line. As Fig3 shows, since there are no more detailed data between two sub lines on Level i+1, the two sub lines can be integrated into one increment line.

Anchors array records the location where the increment data should be inserted into when we rebuild line from scalej to scalei.

The generalization of increment data is a process of traversing of multi-scale line. At first, determine the level i and level j corresponding to scalej and scalei. And then, by algorithm in Section 4.1, traversing the multi-scale line mode, the only difference is just to record the vertices between Level i and Level j. But we need to count the number of vertices with the level less than Leveli, and those are anchors.

4.3 Rebuild original line

Rebuilding is a reverse process of line simplification. By gradually integrating increment data and simplified data, we can rebuild the original line step by step.

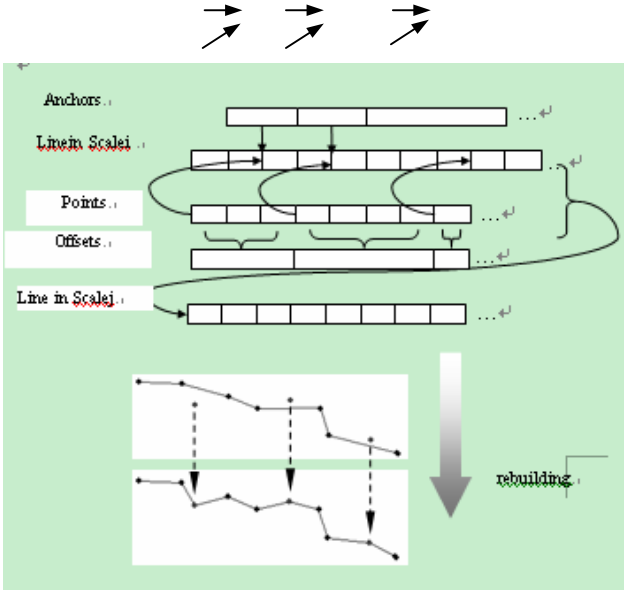


Figure 4 Rebuild original line

After generating increment data, we can use increment data and existing line data to restore line with high resolution. Some parts of the line are simplified into line segments, therefore as long as the eliminated vertices are inserted into original position, the line is restored. In increment structure, each piece of increment sub ling corresponds to sub line i under scale i. Anchor array records the location of each piece of simplified line in Leveli. We just insert increment sub line into subline i,

then we can restore the line under scale scale j. Fig 4 show the process of line rebuilding.

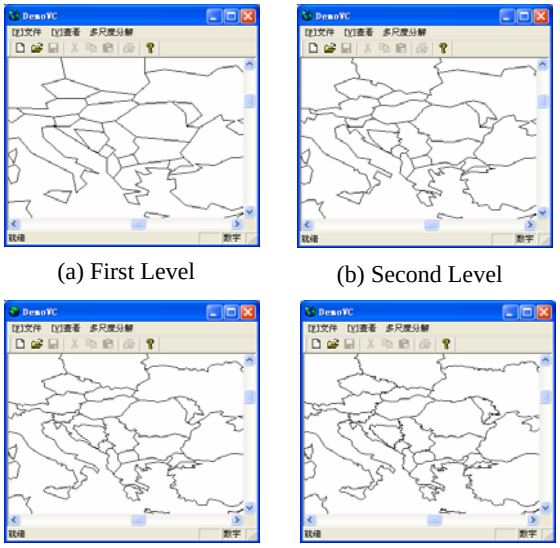
5. EXPRIMENTS

Compared with models in [3][4], the model proposed in this paper owns following characters:

1) The models in this paper support editing. We use related id instead of absolute id to build vertical index in the line to represent order of vertices in the line. When new vertex is add into the line, we just need to modify several nodes in the multi-scale line model in local area, without affecting the whole structure.

2) Efficient database access. For the models in [3][4], even we just request the line in the coarsest level, it still must get data with all scale level. Spatial data are usually stored in the fields of BLOB field. The efficiency to access the BLOB field is low. The data in different level in our model are independent of each other. Therefore, data in different level can be stored into different tables or tablespaces. When users just need data with coarsest scale, we just need read data of that level.

In this paper, we developed a prototype system for generalization and representation of multi-scale line. The system simplified the lines into three levels. The results are showed in Fig 5. We can find the lines in the third level are very close to the shape of original lines while the data volume is just 31.4% of original one. So we still can get good visual effects when users firstly request data and get satisfying web response speed, which is meaningful to mobile terminals such as cell phones and PDAs based on the wireless network.
From Fig 5 we can also find topological relationships are also well persevered.



(a) First Level (b) Second Level

(c) Third Level (d) Original Line

Figure 5 Multi-scale line of four levels

We compared the response time or our model and that in [3] for retrieving the data with coarsest scale from database and render on the screen. The experiment environment is PIII800，512M RAM，SQL Server2000. The experiment result is showed in Table 1. We use two line data packages in this experiment. The data volumes of the two data packages are 1.9 MB and 3.2 MB, and the data volumes of the first level after simplification are 0.57MB and 0.96MB respectively. We measure the response time of user request to the coarsest level of data for the first time. For the first data package, the response time of our model is 0.42s while that of MSLT is 1.03s. Since there is less database IO cost, it need less response time in our model.

Data Volume(MB)		Response time for first users' request(s)	
Original Data	First Level	MSLT	Our model
1.9	0.57	1.03	0.42
3.2	0.96	0.47	0.19

Table 1 response time of retrieving and rendering of data of coarsest level between MSLT and our model

From above table, we can draw such conclusion that our model is more efficient in retrieving and render coarsest data for the first user access.

6. CONCLUSION

Scale is anther important property of spatial data besides geometry and attribute. In this paper, based on the analysis of spatial characters of spatial lines, by Visvalingam-Whyatt algorithm, we simplify spatial line into different scales. In this paper, we present a multi-way tree based multi-scale line model to store and manage line information under different scales. By increment data, the simplified lines can integrate with increment data to restore original data. Compared with Strip tree, our model can clearly manage data under different scales. Compared with MSLT of Jones, our model supports edition of multi-scale line, and own high database access efficiency.

REFERENCE

1. Buttenfield, B.P. Transmitting Vector Geospatial Data across the Internet[C]. Proceedings GIScience, Lecture Notes in Computer Science, Vol. 2478. Springer-Verlag, Berlin. 2002:51-64.
2. Ballard D H. Strip Trees: A Hierarchical Representation for Curves. Communications of the ACM. 1981, 24(5):310-321.

3. Chirstopher B. Jones, Ian M. Abraham. Design considerations for a scale-independent cartographic database[C]. In Proceedings 2nd International Symposium on Spatial Data Handlinig, Seattle 1986:348-398.

4. Min Zhou, Michela Bertolotto. A Data Structure for Efficient Transmission of Generalised Vector Maps[C]. International Conference on Computational Science 2004: 948-955.

5. Visvalingam M, Whyatt J D. Line generalization by repeated elimination of the smallest area. The Cartographic Journal, 1993;30(1): 46-51.