Extensions of GAP-tree and its implementation based on a non-topological data model

Changxiu Cheng, Fangqu Niu, Jun Cai, Yanlu Zhu

LREIS, Institute of Geographical Sciences and Natural Resources Research, CAS, Beijing 100101, PR China
College of Information and Electrical Engineering of China Agriculture University, Beijing 100083, PR China
School of Geosciences and Environment Engineering, Central South University, Changsha 410083, PR China

Online Publication Date: 01 January 2008


To link to this article: DOI: 10.1080/13658810701602120
URL: http://dx.doi.org/10.1080/13658810701602120

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.informaworld.com/terms-and-conditions-of-access.pdf

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied and make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.
Research Article

Extensions of GAP-tree and its implementation based on a non-topological data model

CHANGXIU CHENG*, FANGQU NIU†‡, JUN CAI†§ and YANLU ZHU†
†LREIS, Institute of Geographical Sciences and Natural Resources Research, CAS,
Beijing 100101, PR China
‡College of Information and Electrical Engineering of China Agriculture University,
Beijing 100083, PR China
§School of Geosciences and Environment Engineering, Central South University,
Changsha 410083, PR China

(Received 3 November 2006; in final form 25 July 2007)

This paper discusses extensions of GAP-trees from three aspects and its implementation based on non-topological structure in order to enhance access to large vector data sets. First of all, we apply cartographic generalization rules to build a generalization procedure of the GAP-tree, which makes coarse representations more consistent with human cognition. Second, we replace the three-dimensional (pseudo-) Reactive-tree index with a 2D R-tree index and a B-tree index to improve the system efficiency. Finally, we compress a binary GAP-tree into multi-way GAP-trees in order to reduce data redundancy. The shallower multi-way GAP-trees not only eliminate redundant data but also accelerate the system’s response time. The extensions have been successfully implemented in PostgreSQL. A test of Beijing’s land-use data at the 1:10 000 scale demonstrates that the extended GAP-trees are efficient, compact, and easy to implement.

Keywords: Rules-based; Multi-way; GAP-trees; Large vector data set; Non-topological

1. Introduction

One of the goals of modern information society is to provide flexible access to information for everyone, anywhere, and anytime (Spaccapietra et al. 1999). Geographic Information Systems (GIS), as a modern approach to accessing geographic data, should be able to adaptively provide different data for different users, especially large vector data sets. Van Oosterom and Schenkelaars (1995) suggest that more details and features should be displayed when the user zooms in. This paper focuses on how to flexibly access data according to the scale of the display window.

Wavelet compression and pyramid models have made a breakthrough of adaptive fetching raster data. But when it comes to the vector data, there is no satisfactory solution. In recent years, four techniques have been proposed to flexibly fetch vector data:

*Corresponding author. Email: chengcx@lreis.ac.cn
1. Pre-storage Multi-scale Snapshots: Store a sequence of coarse snapshots from a detailed map in advance, and then transmit the different snapshot according to the different display scales such as the multi-granularity approach (Cheng et al. 2003). This is a simple and common approach, but it suffers from the scale limitations imposed by the source data and presents a difficulty of maintaining consistency within multiple versions of a map (Han et al. 2004).

2. Multi-scale Spatial Index: Build an index to describe the generalization procedure and the generalization results, and then coarse representations can be queried by the scale index and raw data; for example, Reactive-tree (van Oosterom 1991), GAP (generalized area partitioning)-tree (van Oosterom and Schenkelaars 1995), Z-Value (Li and Openshaw 1992, Zhou et al. 2002, Prasher and Zhou 2003), Three Level Data Model (Wei et al. 2000), and Multiple R-tree (Kwon and Yoon 2002). These techniques are compatible with the current spatial database technology, which does not require any changes on vector data model.

3. Multi-scale Vector Data Structure: Store scale as an important dimension in the vector data structure, and then a coarse representation can be obtained by reorganizing the vector data; for example, Strip-Tree (Ballard 1981), BLG (van Oosterom 1991), PR-File (Becker et al. 1991), and Map Cub Mode (Timpf 1998). This solution is more thorough and innovative, but it is difficult to communicate directly with the current GIS softwares because of the user-defined vector data model.

4. Hybrid Techniques: Many other solutions cannot be easily classified as one of the three approaches discussed above. These other solutions usually involve ideas from more than one approach, such as the Multi-scale Hilbert R-tree (Chan and Chow 2002), the integration of Reactive Tree, GAP-tree and BLG-tree (van Oosterom and Schenkelaars 1995, van Oosterom 2005). We called them hybrid techniques in this paper.

The integration of Reactive-tree, GAP-tree, and BLG-tree may be a relatively practical solution among all of the techniques mentioned above. Van Oosterom and Schenkelaars (1995) presented the integration and gave the details of its implementation. In recent years, researchers have paid more attention to a topological GAP structure (tGAP structure). For example, Vermeij (2003) and Vermeij et al. (2003) first presented the tGAP structure, and the tGAP structure was subsequently used in progressive transfer (van Oosterom 2005) and vario-scale data server (van Oosterom et al. 2006).

However, non-topological data models, such as SDO_geometry and Wkb_Geometry, have been supported by many SDBMS (Spatial Database Management Systems) vendors, since non-topological data can be displayed much faster (Zhai et al. 2002). In order to make it easy to communicate with other GIS software, this paper will discuss how to develop and implement the GAP-trees based on the non-topological native geometry data type, especially for large data sets. Although some problems (such as data redundancy) may be solved in tGAP, it is still necessary to explore another solution from the perspective of non-topological structure.

The remainder of this paper is organized as follows. Section 2 introduces the Reactive-tree and GAP-tree, and discusses some further research topics. Section 3 shows the flow to create rule-based GAP-trees, the integration of GAP-tree with 2D, and the improvement from binary GAP-tree to multi-way Gap-trees. Section 3 also
includes a detailed algorithm of building the extended GAP-trees. Section 4 presents an implementation using PostgreSQL. This implementation is tested and evaluated with Beijing’s land-use data at 1:10 000, scales and the results are reported in section 5. The final section draws conclusions and discusses future work.

2. Researches about Reactive-tree and GAP-tree

The GAP-tree has to integrate with the Reactive-tree (van Oosterom and Schenkelaaars 1995) or pseudo-Reactive-tree (van Oosterom 2005), so this section begins with the Reactive-tree.

2.1 Reactive-tree

The Reactive-tree, which is used to control if the geometry object should be displayed or not, was proposed by van Oosterom (1991). The Reactive-tree is a variation of the R-tree. It assigns an importance value to each geometric object, and each object will be stored in a certain level according to its importance value, which is calculated by equation (1) (van Oosterom 1991). Less important objects receive lower values, while more important objects receive higher values. During a search, the tree will only retrieve those objects whose importance value is high enough. As figure 1 shows, the geometry objects, which are labelled 7, 10, 11, and 14, have higher important values and thus have a higher display priority.

\[ I(o) = f(\text{type, attributes, size}) \] (1)

The Reactive-tree, which is an early multi-scale spatial index technique, has made a significant impact on the development of multi-scale spatial index. However, the Reactive-tree only supports the selection operator. The generalization based on the Reactive-tree will leave out some areas with lower importance values and produce a map with holes.

2.2 GAP-tree

In 1993, the GAP-tree was introduced to avoid holes on a map created by the Reactive-tree (van Oosterom 1993). The GAP-tree is an area-partitioning hierarchy, which is used to decide which area is removed and also which area will fill the gap of the removed feature. The GAP-tree initially only supports the amalgamation operator. Subsequently, the GAP-tree also supports aggregation and merging operators.

![Figure 1. Reactive-tree (from van Oosterom, 1991).](image)
An example in figure 2 illustrates the GAP-tree and the extended GAP-trees. It shows a generalization example in five steps, ranging from detailed to coarse displays. Each polygon is given an id and a computed importance value (shown in a smaller font). At every step, the least important object (i.e. feature 6 in step 0) is removed, and its area is assigned to its neighbour (i.e. feature 2 in step 0) with the highest collapse value (calculated by formula 2). Also, a new id is assigned to the face, which is enlarged and becomes more important. The importance value of the enlarged object will be recomputed according to equation (1). This process continues until one large face is left. Therefore, the hierarchy of figure 2 can be stored as the GAP-tree shown in:

\[ \text{Collapse}(a, b) = f(L(a, b), \text{CompatibleTypes}(a, b), \text{WeightFactor}(b)) \]

Where \( L(a, b) \) is the common boundary between \( a \) and its neighbour \( b \), \( \text{CompatibleTypes}(a, b) \) determines how close the two feature types of \( a \) and \( b \) are, and \( \text{WeightFactor}(b) \) is weight factor of the feature type of \( b \).

### 2.3 Further research topics

#### 2.3.1 Generalization of the GAP-tree should involve a great deal of human experiments about the geographical data (Cecconi 2003).

The functions in equation (2) have taken the object’s area, length, and classification as parameters to decide how to generalize it. If we substituted an expert decision model for these functions, the generalization results would be more consistent with human cognition (figure 3).

#### 2.3.2 Efficiency of the (pseudo-) Reactive-tree is decreased under some conditions.

In figure 1, some objects could not be stored in a non-leaf node. This implies that the Reactive-tree cannot be optimally balanced, because not all leaf nodes are at the lowest level of the tree. Thus, searching in the Reactive-tree is not very efficient if the data do not have a hierarchical distribution (Chan and Chow...
2002). Consequently, the pseudo-Reactive-tree, which is built by merging a 2D box and a 1D importance range as a 3D box, is used to balance the Reactive-tree, but the 3D Reactive-tree is still not as efficient as the 2D R-tree due to the dimension disaster theory.

2.3.3 Binary GAP-tree is so detailed that it requires a large disk space to store the data. Van Oosterom’s group developed the tGAP structure to avoid the repeated storage of the same coordinates or same edges (van Oosterom 2005, van Oosterom et al. 2006). It is a good solution only for the topological model. When it comes to the non-topological model, we will focus on expressing the generalization procedure with as few representations as possible.

3. Extension of the GAP-tree

3.1 Gap-tree creation based on generalization rules

To make the coarse representation more consistent with human cognition, some cartographic generalization rules should be used to decide on the hierarchical structure of the GAP-tree. The rules will be different in different applications. In order to make the rules easy to modify and extend, the work flow of GAP-tree creation is separated into three relatively independent modules. As figure 4 shows, they are the pre-process module, rule-based module, and gap-trees model. First, the candidate features to be generalized are identified by the pre-process module, and then these features are transmitted to the rule-based module. The rule-based module provides a reasonable generalization suggestion and then transmits the generalization operator and the related features to the GAP-tree module. Finally, the GAP-tree module executes the generalization operator and records this process in a GAP-tree (figure 4).

These modules are relatively independent. Changes in the rule-based module will not cause changes in other modules. The detailed generalization rules in our systems are given in section 4.1.

Figure 3. GAP-tree associated with the scene in figure 2.

Figure 4. Flowchart to build Gap-trees based on generalization rules.
3.2 Integration of GAP-tree and 2D R-tree

The reason why the (pseudo-) Reactive-tree is inefficient for non-hierarchical distribution data is its attempt to integrate the spatial data and the display priority into one index. In fact, the display priority has been recorded in the GAP-tree. A B-tree index based on the display priority of the GAP-tree also can be used to query the data at a certain display level, so the display priority of Reactive-tree can be removed, and the (pseudo-) Reactive-tree would be turned into a 2D R-tree. The original index can be replaced with a 2D R-tree index and a B-tree index of display priority in GAP-tree (figures 5 and 6).

Based on the previous examples shown in figure 2, figure 5 shows that its GAP-tree takes the level information as scale index. The objects at a certain level can be fetched by the GAP-tree. For example, the objects at level 2 should be the union of the nodes at the level 2 (such as 7, 1), the leaf nodes above the level 2 (such as 3) the and the non-leaf nodes whose levels are above 2 and whose children’s levels are below 2 (such as 8).

Certainly, it is unnecessary to search throughout the whole tree on querying data at certain level. In Relational Database Management Systems (RDBMSs), some relational operators can be used to efficiently implement the search logic mentioned above. For example, SQL1 in section 4.2 shows how to use the relational operator to fetch objects at level 1. SQL 2 in section 4.2 shows how to use the GAP-tree and 2D R-tree to fetch the objects at level 1 and in the certain area whose box is BOX3D(484227.3935609076 253069.640625, 535426.1064390923 296054.65625).

3.3 From binary GAP-tree to multi-way GAP-trees

The Binary GAP-tree records in detail the generalization procedures of every two features from many nodes to one root, as figure 3 shows. It not only makes the GAP-tree deeper but also slows the response time. In fact, it is unnecessary to combine all features to one feature (i.e. object 11 in figure 5) and to itemize the generalization process of every two objects. Therefore, we can select some key representations (i.e. representations at step 0, 2, 4) to describe the generalization process. Then, the GAP-tree in figure 5 can be compressed to the GAP-trees in figure 6. Of course, the data in figure 2 are so simple that the GAP-trees are still binary. In large data sets, the GAP-tree should be multi-way tree because there must be some objects generalized from many small objects.

Figure 5. GAP-tree with display priority.
The compressed multi-way GAP-tree only stores some key representations to simulate the generalization process. It will cause less redundant data and make the system more efficient.

3.4 Algorithm for creating multi-way GAP-trees

1. Define a certain discriminable threshold in the current screen; for example, 2pix*3pix as the threshold for area, 2pix as the threshold for gap, if there is one feature whose MBR is below this threshold or any two features whose minimal distance is below this gap threshold, they will be candidates for generalization.
2. Set the display scales for key representations in GAP-trees. As we know, the system’s response time gradually declines during a continuous zoom operation. For a large data set, the response time is often unacceptable when users attempt to display at the overview or near the overview level. Therefore, there should be more coarse representations on (near) the overview than the views with larger display scales. In this paper, the geometric progression method is adopted to set the display scale. Its procedures are listed below:
   
   i. Calculate the display scale of overview as DSmin;
   ii. Calculate the display scale on which the minimal feature might be generalized as DSmax;
   iii. According to a factor whose default value is 1.4, the first display scale value is the product of the DSmin and this factor, and then the next display scale value will be the product of the above-computed display scale and the factor. This process continues until the computed display scale is larger than the DSmax. Figure 7 shows the distribution of display scale samples. If the user is willing to accept more data redundancy to expedite the response, they could choose a smaller factor value.
3. Initiate all nodes in GAP-trees. At first, all objects are scattered leafs, and their levels can be initialized as \( N \), where \( N \) is the total number of representations (figure 8(a)).
4. Find the candidate features at level \( i \), where \( i \) is from 0 to \( N \), and then transmit them to the rule-based module. In this step, the rule-based module will provide a generalization suggestion. Not all of the transmitted features are
generalized. The decision of which ones will be generalized is made by rule-based module.

5. Generalize the geo-data and revise the GAP-trees. After the map generalization, the levels of new features will be initialized as $N$; and the levels of original features will be modified to $i$. The evolution relationships will be recorded as the links between the parent nodes and their child nodes. Figure 8(a), (b), and (c) show the step-by-step procedure of building a multi-way GAP-trees.

6. $i = i + 1$; if $i$ is below $N$, then go to step 3.

After the creation of multi-way GAP-trees, the next consideration is the implementation.

4. Extended implementation

4.1 Rules of generalization

Figure 9 gives a flow chart of how to select appropriate operators and reasonable features. Some terms relevant to figure 9 are introduced below.

4.1.1 Exception features. Exception features are the objects that cannot be generalized, no matter how small they are. For example, the Xisha islands cannot be eliminated from the China map, although they are very small. These exception features should be defined in advance by giving some appropriate values to its attributes. For example, Feature (i).Name='Xisha islands'.'

4.1.2 Small features and small gap. Definitions about the small feature and small gap are divided into normal criterion and specific criterion. The normal criterion is required, and the specific criterion is added by the user. If a feature belongs to one class defined in specific criterion, the rules related to this class will work; otherwise, the normal criterion will work. Taking land-use data as an example, figure 10 shows some definitions about small objects and the small gap. The parameters of ‘ThresholdArea’ function and ‘ThresholdLength’ function could be modified according to different requirements specified by the users.
4.1.3 Rules of elimination, merge, amalgamation, and aggregation. The rules of elimination, merge, amalgamation and aggregation are divided into graph rules and attribute rules. The graph rules describe the prerequisites to implement the various generalization operators. The attribute rules record how to generalize the features’ attribute after graph rules. Users can add more attribute rules according to their requirements. It is not necessary to evaluate if the feature(i) is small object or if the minimal distance between feature(i) and feature(j) is small gap again, because the decisions have been made by the logic in figure 9.

4.2 Storage structure

Three tables are defined to store the GAP-trees in PostgreSQL:

1. Structure of XXX_Meta Table (table 1), where ‘level_id’ is the id number of level and ‘Display_scale’ is the display scale corresponding to the level.
2. Structure of XXX feature class Table (table 2), where ‘Ogc_fid’ is the identity of spatial object, ‘Wkb_geometry’ stores the geometry information of object,
Small Objects

ThresholdArea (x,y) -- A function to get the area corresponding to x Pix * y Pix

Normal Criterion:
If(feature(i).MBR < ThresholdArea (4,6)) then the feature(i) is small objects.

Specifically Criterion:
If(((feature(i).landuse_type is 'residential quarters' or 'industrial') and (feature(i).MBR<ThresholdArea (2,3))
or((feature(i).landuse_type is 'forest', 'pasture' or 'unused lands') and (feature(i).MBR<ThresholdArea (6,9)))
or((feature(i).landuse_type is 'water area') and (feature(i).MBR<ThresholdArea (1,1)))
then feature(i) is small objects.

Small Gap

ThresholdLength (x) -- A function to get the length corresponding to x Pix
MinimalDistance(a,b) -- A function to get the minimal distance between geometry a and b

Normal Criterion:
Gap=(MinimalDistance(feature(i),feature(j))
If(Gap < ThresholdLength (3)) then the gap is small gap.

Specifically Criterion:
If((feature(i).landuse_type is 'highways') and (Gap < ThresholdLength (5))
or((feature(i).landuse_type is 'rural roads') and (Gap < ThresholdLength (2)))
then the gap is small gap.

Figure 10. Examples of small objects and a small gap.

'isleaf' indicates if the object is a leaf node in the GAP-trees, 'level' indicates the level of the GAP-trees at which the object is located, and 'A1' to 'An' are the attribute data fields.

3. Structure of XXX_GAP_trees Table (table 3), where 'Ogc_fid' is the identity of spatial object, 'Parent_id' is the identity of the parent object of object 'Ogc_fid', and 'Child_id' is the identity of the child object of object 'Ogc_fid'.

4.3 Indexes

In order to accelerate the database's response, the following indexes are built in this paper:

1. a 2D R-tree index of 'Wkb_geometry';
2. two B-tree indexes of 'level' and 'child_level'.

<table>
<thead>
<tr>
<th>level_id</th>
<th>Display_scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.263</td>
</tr>
<tr>
<td>1</td>
<td>0.011</td>
</tr>
<tr>
<td>2</td>
<td>0.005</td>
</tr>
</tbody>
</table>
4.4 Query the different representations according to the display environment

During the pan and zoom operations, it is easy to obtain the current display scale of map view. Then, the data should be selected from the level whose display scale is the nearest to the current display scale. For example, if the current display is 0.010, the data should be fetched from level 1. The SQL1 is used to query all records corresponding to level 1; the second SQL is used to query all records located in BOX3D area in level 1.

- SQL 1:

\[
\text{Select } * \text{ from Beijing_landuse}
\]

\[
\text{where(level = 1 or (level > 1 and isleaf = 1) or (level > 1 and child_level < 1))}
\]

- SQL 2:

\[
\text{select } * \text{ from Beijing_landuse}
\]

\[
\text{where (level = 1 or (level > 1 and isleaf = 1)}
\]

\[
\text{or (level > 1 and child_level < 1)) and wkb_geometry} \&\& \text{GeometryFromText ('BOX3D(484227.3935609076 253069.640625, 535426.1064390923 296054.65625)')}:: \text{box3d, } -1
\]

4.5 Implementation of simplification

Van Oosterom customized several spatial data types to implement the BLG-tree, for example, the ‘PolyLine2’ in PostgreSQL and the ‘blgtree’ in Oracle. These data structures are greatly different with familiar data types (i.e. Wkb_Geometry or SDO_Geometry). Thus, the translation model (i.e. Blg2Ply in PostgreSQL,

Table 2. Table ‘XXX’ with features’ geometry, attributes and its lever in the GAP-trees

<table>
<thead>
<tr>
<th>Ogc_fid</th>
<th>isleaf</th>
<th>Level</th>
<th>Child_level</th>
<th>Wkb_geometry</th>
<th>A1</th>
<th>.....</th>
<th>An</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y</td>
<td>0</td>
<td>Null</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
<td>0</td>
<td>Null</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Table ‘XXX_GAP_trees’ with the relationships between parents and children of the GAP-trees

<table>
<thead>
<tr>
<th>Ogc_fid</th>
<th>Parent_id</th>
<th>Child_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Generalized in Oracle) is indispensable for communicating with current GIS software.

In this paper, we used the $M$ values of the geometry points to record its simplification results using the Douglas–Peucker algorithm. The $M$ value of a point is corresponding to a level value, something like figure 8. When fetching the data at level $i$, the system will select these points whose $M$ values are equal or above $i$ by the ‘Simply_Level’ function developed by ourselves. This solution will be simple and easy to implement, and the geometry also can be directly recognized by other GIS software.

5. Test

5.1 Test data and building GAP-tree

The test data are the Beijing land-use data at 1:10 000 scale. They contain 186 329 polygons (about 240 MB) and 11 668, 862 vertices. The SDBMS used in the test is PostgreSQL and PostGIS running on a Dell 2600 computer. A client software, named the Multi-viewer, runs on a P4 computer with a 1-GHz CPU and 256 MB of RAM. The Multi-viewer software is programmed with Java, and its module of building the GAP-trees is coded with C.

The definitions of small objects, small gap, and generalization rules have been shown in figures 10 and 11, and the progressive factor is set at 1.4. According to step 2 of the algorithm in section 3.4, the sequence of display scales should be computed as 0.000836, 0.001170, 0.001638, 0.002293, 0.003212, 0.004495, 0.006293, 0.008812, 0.012335, 0.017270, and 0.024182. Based on these rules and the steps followed in section 3.4, the multi-way GAP-trees can be easily built.

5.2 Testing results

After building the GAP-tree, we took the centre point of map data as the map view’s centre and adjusted the display scale to the above-mentioned 11 display scales on using the extended GAP-trees & Simplify_Level function or not. The number of queried recorders and the average response time of system were recorded in table 4. Figure 12 shows screen captures using the extended GAP-trees & Simplify_Level function, while figure 13 shows screen captures without the extended GAP-trees & Simplify_Level function.

5.3 Results analysis

The extended GAP-tree & Simplify_Level function proposed in this paper dramatically reduces the response time while keeping a comparable visual quality. The response time is controlled within 10 s, which is an acceptable waiting time. Figure 14 shows the effect of time reduction, and the similar visual quality can be deduced from a comparison of figures 12 and 13.

It may not be sufficient to compare the efficiency between the extended GAP-trees and the original one due to the different test environments, different test input data and different simplify methods employed. However, the ratio of response time may be a useful indicator. The maximum ratio of response time between using the extended GAP-trees & Simplify_level function or not is about 45; while the maximum ratio of response time between using the GAP-tree & BLG-tree or not is only 39 (285.8/7.3) in DLMS DFAD and 29 (143.5/4.9) in WDB II (van Oosterom and Schenkelaars 1995).
6. Conclusions and prospects

6.1 Summary of the main results

1. The coarse representations can be made more consistent with human cognition to build the GAP-trees by using some generalization rules from the cartographer.

Table 4. Testing result on using GAP-trees & Simplify_Level function or not.

<table>
<thead>
<tr>
<th>Display scale</th>
<th>Using extended GAP-trees &amp; Simplify_Level function</th>
<th>Without extended GAP-trees &amp; Simplify_Level function</th>
<th>Ratio of response time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. of queried recorders</td>
<td>Average response time (s)</td>
<td>No. of queried recorders</td>
</tr>
<tr>
<td>0.000836</td>
<td>1405</td>
<td>4.613</td>
<td>186329</td>
</tr>
<tr>
<td>0.001170</td>
<td>2335</td>
<td>7.251</td>
<td>160131</td>
</tr>
<tr>
<td>0.001639</td>
<td>2872</td>
<td>6.932</td>
<td>108549</td>
</tr>
<tr>
<td>0.002294</td>
<td>3068</td>
<td>4.551</td>
<td>67519</td>
</tr>
<tr>
<td>0.003212</td>
<td>2958</td>
<td>2.893</td>
<td>38606</td>
</tr>
<tr>
<td>0.004496</td>
<td>2579</td>
<td>1.768</td>
<td>20726</td>
</tr>
<tr>
<td>0.006295</td>
<td>2097</td>
<td>1.089</td>
<td>10061</td>
</tr>
<tr>
<td>0.008813</td>
<td>1651</td>
<td>0.687</td>
<td>5066</td>
</tr>
<tr>
<td>0.012338</td>
<td>1147</td>
<td>0.440</td>
<td>2440</td>
</tr>
<tr>
<td>0.017273</td>
<td>702</td>
<td>0.248</td>
<td>1095</td>
</tr>
<tr>
<td>0.024182</td>
<td>417</td>
<td>0.602</td>
<td>417</td>
</tr>
</tbody>
</table>
2. It is easy to alter and extend the generalization rules for different applications, since the pre-process module, rule-based module, and gap-trees model are relatively independent.

3. The system can be made more efficient by replacing the (pseudo-) Reactive-tree with a 2D R-tree index and B-tree index.

Figure 12. Some screenshots on using GAP-trees & Simplify_Level function.

Figure 13. Some screenshots without GAP-trees & Simplify_Level function.
4. The multi-way GAP-trees use key representations to simulate the generalization process, thus improving the utilization factor of redundant data and system efficiency.

5. This paper makes full use of some prevalent technologies in SDBMS (i.e. 2D R-tree, Wkb_geometry, M value) to implement the extended GAP-trees. Therefore, it is easy to implement and transplant, and the implementation will be more compatible with the current GIS software.

6. The test shows that the multi-way GAP-trees are efficient, compact, and easy to implement in current RDBMS.

6.2 Possible further enhancements

Future research on further improving the functionality of the extended GAP-trees could include the following topics:

1. Develop more rules for rule-based module. Only some rules for land-use data can be integrated in our system. In fact, a different application needs different rules. More rules are needed for the rule-based module.

2. Develop additional scientific sampling methods to select reasonable representations. The selection of key representations should be deduced by the characteristics of vector data, the characteristics of map view, and the users’ requirements for the response time. The sampling method in section 3.4 is only a rough estimate. More scientific sampling methods are needed to make the extend GAP-trees more efficient.

3. Implement the extended GAP-trees based on other generalization operators. The extension only implements the simplification, elimination, merge, amalgamation and aggregation operators. In fact, the generalization is a holistic process that includes exaggeration, classification, symbolization, typification, and anamorphose, etc. In order to be more consistent with the cartographer cognition, the implementation of other generalization operators is required.
Acknowledgements
This research was supported by the Hi-Tech Research and Development Program of China under grant no. 2007AA120401, the National Natural Science Foundation of China under grant No.40401047 and the Front Project of IGSNRR under grant No.066U0307SZ. Their support is gratefully acknowledged. Thanks also to Prof. Shih-Lung Shaw for his constructive comments that have substantially improved this paper.

References

