A QUADTREE SPATIAL INDEX METHOD WITH INCLUSION RELATIONS FOR THE INCREMENTAL UPDATING OF VECTOR LANDCOVER DATABASE

X.G. Zhou[1,*], H.S. Wang[1,2]

[1]School of Geosciences and Info-physics, Central South University, Changsha 410083, China -zxgcsu@foxmail.com
[2]School of Software, South China Normal University, Foshan 528225, whs2020@126.com

Commission IV, ICWG IV/III

KEY WORDS: Spatial index; Complex polygon; Inclusion relation; Quadtree; Incremental updating

ABSTRACT:

In vector landcover database, there are a lot of complex polygons with many holes, even nesting holes. In the incremental updating (i.e., using the change-only information to update the land cover database), a new changed parcel usually has 2-dimensional intersections (e.g., overlap, cover, equal and inside, etc.) with several existing regions, automatic updating operations need to identify the affected objects for the new changes at first. If the existing parcels include complex polygons (i.e., the polygon with holes), it is still needed to determine if there are 2-dimensional intersections between the new changed polygon and each holes of the involved complex polygons. The relation between the complex polygon and its holes has not been presented in the current spatial data indexing methods, only the **MBB** (Minimum Bounding Box) of the exterior ring of the complex polygon has been stored, the non-involved holes can not be filtered at the first step of spatial access methods. As the refinement geometric operation is costly, therefore the updating process for the complex polygons is very complicated and low efficient using the current spatial data indexing methods. In order to solve this problem, an improved quadtree spatial index method is presented in this paper. In this method, the polygons is divided to two categories according to the relations with the quadrant axes, i.e., disjoint to the axes and intersect with the axes. The intersect polygons are still divided to 5 cases according to the intersection position among the polygons and the different level quadrant axes. The intersection polygons are stored in the different level root nodes in our index tree, and five buckets denoted as *XpB, XnB, YpB, YnB, XYB* are used to store the polygons intersecting the different level quadrant axes respectively. The polygons disjoint to all quadrant axes are stored in the leaf nodes in this method. The authors developed the spatial index structure with inclusion relations and the algorithms of the corresponding index operations (e.g., insert, delete and query) for the complex polygons. The effectiveness of the improved index is verified by an experiment of land cover data incremental updating. Experimental results show that the proposed index method is significantly more efficient than the traditional quadtree index in terms of spatial query efficiency, and the time efficiency of the incremental updating is increased about 3 times using the proposed index method than that using the traditional quadtree index.

1. INTRODUCTION

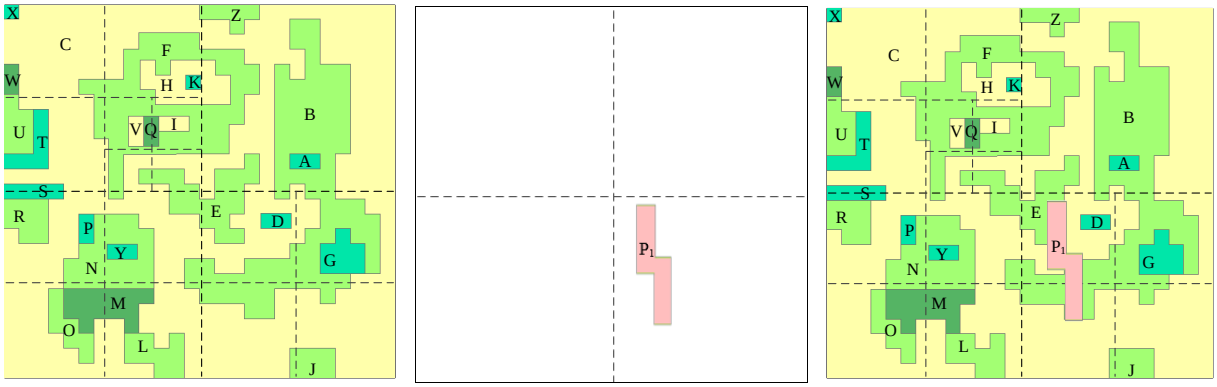
In vector landcover database, there are a lot of complex polygons with many holes, even nesting holes. In the incremental updating (i.e., using the change-only information to update the land cover database), a new changed parcel usually has 2-dimensional intersections (e.g., overlap, cover, equal and inside, etc.) with several existing regions, automatic updating operations need to identify the affected objects for the new changes at first. If the existing parcels include complex polygons (i.e., the polygon with holes even nesting holes), it is still needed to determine if there are 2-dimensional intersections between the new changed polygon and each holes of the involved complex polygons.

In GIS, **Spatial index is used to make** refinement geometric operations just execute only on a limited number of objects. Common spatial index methods include Quadtree (Hjaltason & Samet, 2002; Wei & Tanaka, 2012), **R-trees (**Guttman, 1984), R+tree (Sellis, et al, 1987), R* tree (Beckmann, et al, 1990), Grid spatial index, **Hilbert R-tree,** kd-tree, etc (Philippe et al, 2002; Wikipedia, 2018). Typically these methods are grouped using the minimum bounding Box **(MBB**) to minimize the index file size and increase the filter efficiency. While the relationship between the complex polygon and its holes has not been presented in the current spatial data indexing methods, only the **MBB** of the exterior ring of the complex polygon has been stored, the non-involved holes can not be filtered at the first step of spatial access methods. For example, in Figure 1, C is a complex polygon with more than 1000 holes, e.g., B, D, E, F, P, etc. (Figure 1a). P_1 is a new changed polygon ((Figure 1b). Among the holes, P_1 just have 2-dimensional intersections with B and E. However, in updating the refinement geometric operation has to be done between P_1 and the all holes of C using the current index method. As the refinement geometric operation is costly, therefore the updating process for the complex polygons is very complicated and low efficient using the current spatial data indexing methods.

Noted that in the incremental updating of land cover database, a new change usually just intersects with limited several polygons (or several holes of the complex polygon). If the inclusion relation between the complex polygon and its holes can be presented in spatial index, the non-involved holes can be filtered at the first step, the incremental updating efficiency for land cover database will be highly improved. Furthermore,

* Corresponding author

linear quadtree indexx is dynamic and efficient in terms of memory space and response time, and has been popular used in commercial spatial DBMSs. Based on the above observations, an improved quadtree spatial index method is presented in this paper. In this method, the polygons is divided to two categories according to the relations with the quadrant axes, i.e., disjoint to the axes and intersect with the axes. The intersection polygons are stored in the different level root nodes in our index tree, and five buckets are used to store the polygons intersecting the different level quadrant axes respectively. The polygons disjoint to all quadrant axes are stored in the leaf nodes in this method.



(a) Data before updating (b) Increment parcel (c) Data after updating

Figure 1. Example Data of Land Cover Database Incremental Updating

The remainder of the paper is structured as follows. In Section 2, the construction of the improved quadtree spatial indexing is presented, including the structure of inclusion relations between the complex polygon and its holes, the optimization of the traditional spatial indexing quadtree and its construction. The corresponding index quadtree modification operations caused by the dynamic operations of spatial objects (e.g., insert, delete and query) is discussed in Section 3. Based on the improved quadtree spatial indexing for complex polygon, an experimental test of this study and a comparison between our improved quadtree index method and the MX-CIF quadtree index are presented in Section 4. Section 5 provides a summary and concludes the discussion.

2. THE CONSTRUCTION OF THE IMPROVED QUADTREE INDEXING

As mentioned above, in the incremental updating of land cover database, it is needed to store the inclusion relation between the complex polygon and its holes. In current GIS topological model, some topological relations (i.e., the connection relations between the nodes and arcs; the adjacent relationship between polygons) are stored explicitly. However, the inclusion relation between the complex polygon and its holes is not explicitly represented. Therefore at first a method used to store the inclusion relationship is presented.

Complex polygon include hole and nested holes, in another word, the inclusion relation include direct inclusion and indirect inclusion. As figure 1(a) shows, the generalized region of F (i.e. the union of F and its holes) is one hole of C; the generalized region of H (i.e. the union of H and its hole- K) is one hole of F, and the union of V, Q and I is another hole of F. There is no direct inclusion relationship between C and H and K, in another word, C is the ParentPolygon of F; H and the union of V, Q and I, are the two Children Polygons of F. It is assumed that the current polygon is noted as **CP, the** Parent Polygon is denoted as PP, RIP is the ring of the current polygon in its parent, **and CPL is the** Children Polygon List of the current polygon. Then the direct inclusion relationship can be represented using the structure: {CP, PP, RIP, CPL}.

There is a problem in the traditional quadtree index. The search space is recursively decomposed into quadrants, the quadrants are named North West (NW), North East (NE), South West (SW), and South East (SE). The quadrants space is not overlap at the same level. While the object duplication in neighbour cells increases the index size seriously, for example, in figure 1(a), object E has to be stored in the quadrants of NW, NE, SW, SE; B in NE, SW, SE; F in NW, NE, SW, etc., at the first level. This will lay a burden on the index size and the dynamic operations.

Based on this observation, an improved quadtree spatial index method is presented in this paper. In this method, the polygons is divided to two categories according to the relations with the quadrant axes, i.e., disjoint to the axes and intersect with the axes. The intersect polygons are still divided to 5 cases according to the intersection position among the polygons and the different level quadrant axes, i.e., intersection only at the X positive axis, intersection only at the Y positive axis, intersection only at the X negative axis, intersection only at the Y negative axis, intersection at X and Y axes. The intersection polygons are stored in the different level root nodes in our index tree, and five buckets denoted as *XpB, XnB, YpB, YnB, XYB* are used to store the polygons intersecting the different level quadrant axes respectively. The polygons disjoint to all quadrant axes are stored in the leaf nodes in this method. The data structure of the nodes is as follows:

{NID, NMBB, Subtrees, PPtr, Depth, *XpB, XnB, YpB, YnB, XYB***}**

In the above structure, NID denotes the identity of the node; NMBB denotes the MBB of the all polygons stored in this node, i.e., the union of the MBB of the all polygons; Subtrees denotes the Subtrees of this node; **PPtr** denotes the Parent pointer in the tree; Depth denotes the depth of the node level; XpB, XnB, YpB, YnB, XYB are the buckets used to store the polygons intersecting the quadrant axes. The relation between the complex polygon and its holes is stored in the polygon structure: {**PID, PP, ER, LIPs}.** In this structu**re, PID** denotes the identity of the polygon**, PP** denotes the Parent polygon**, ER** denotes the exterior ring**, and LIPs** denotes the list of inside polygons. The improved quadtree spatial indexing method for

complex polygon is shown in Figure 2 using the data of Figure 1 (a) as example data. The left is the quadtree of Figure 1 (a), the right inclusion relationship table of Figure 1 (a).

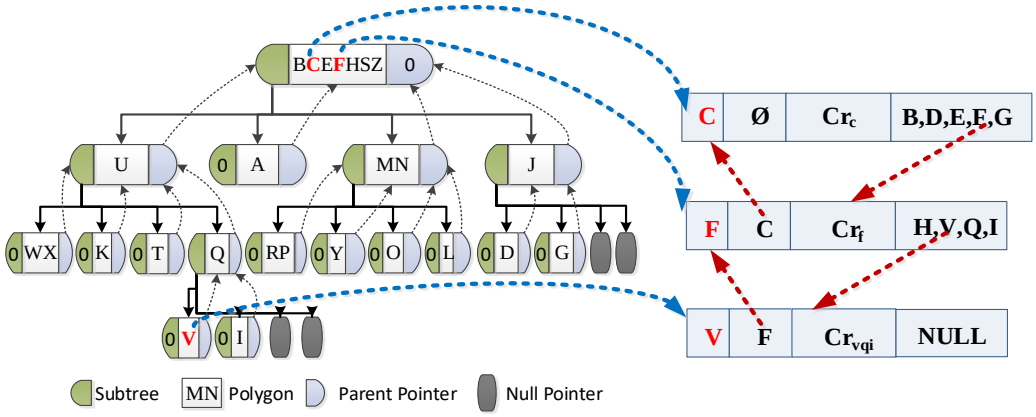


Figure 2. Improved Quadtree Index of Fig.1 (a) and the Nested Inclusion Relations among Polygon C, F, V

To build the the improved quadtree, at first, the MBB of all polygons and holes are calculated (Wei & Tanaka, 2012) and the inclusion table are constructed by the way. The search space is recursively decomposed into quadrants until the number of MBBs in each node is less than the threshold number; The polygons whose MBB overlaps the quadrant axes are stored to the five buckets *XpB, XnB, YpB, YnB, XYB*, the objects whose MBB purely inside the quadrant are stored to the quadrant nodes at different level respectively. In this improved quadtree, each object is just stored in one node, i.e., the node corresponding to the minimize quadrant includes its MBB.

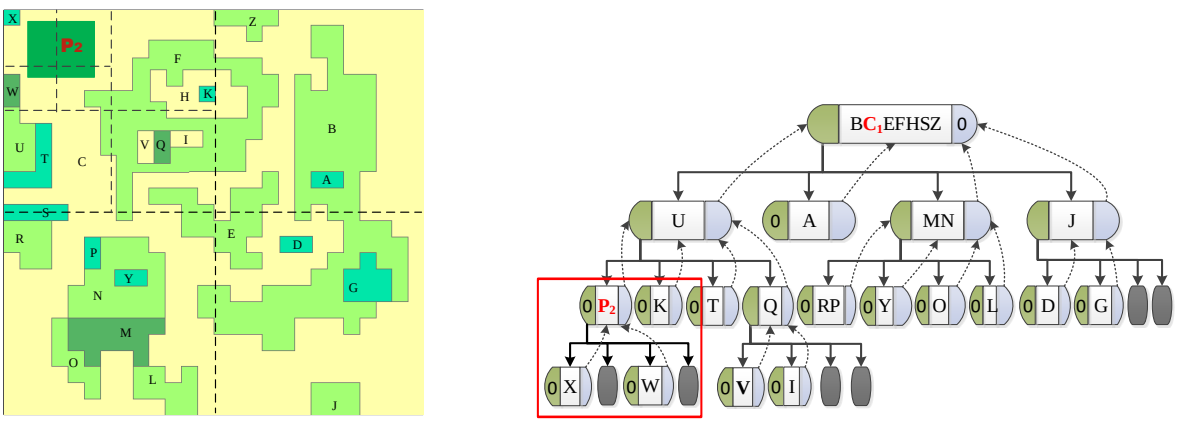
3. OPERATIONS FOR THE IMPROVED QUADTREE INDEXING

Query ususally include point and window query. As the improved quadtree is aimed to solve the problem with the complex polygons, we mainly discuss the window query in this paper. For the window query, the following steps are included:
1) From the tree root to leaf, charge if any polygons overlapped the argument MBB in the five buckets and the quadrant nodes at each level, and construct the list of candidate MBBs;
2) Scan the list of candidates, for the complex polygons, select the children polygon with RIP overlapped the argument MBB to the result set; for the other polygons,;
3) For the other candidate polygons, if any point of the argument window is inside the exterior ring, and not in the candidate children’s polygon, add it to the result set.

In the updating process, dynamic insertion and deletion operations are used frequently. We will describe the dynamic operations in following.

To insert a polygon to the improved quadtree, at first, the bucket or leaf will be found, then two cases may happen: 1) the page is not full, a new entry is inserted; 2) the page is full, then the quadrant is split. The inclusion relation table has to be maintained during insertion process. Figure 3 is used as an example to illustrate the insertion operation.

1) P_2 should be insert into the NW quadrant node at third level, as the number of objects equals to 3, larger than the threshold number 2, NW quadrant node is split;
2) P_2 is stored in the root bucket at the forth level, the old objects in the NW quadrant node at third level are stored to the corresponding quadrant node at forth level;
3) Searching the father polygon of P_2 upward from the NW quadrant node at third level, get the father polygon of P_2, i.e. C;
4) Using C_1 to replace C, and update the inclusion relation table.



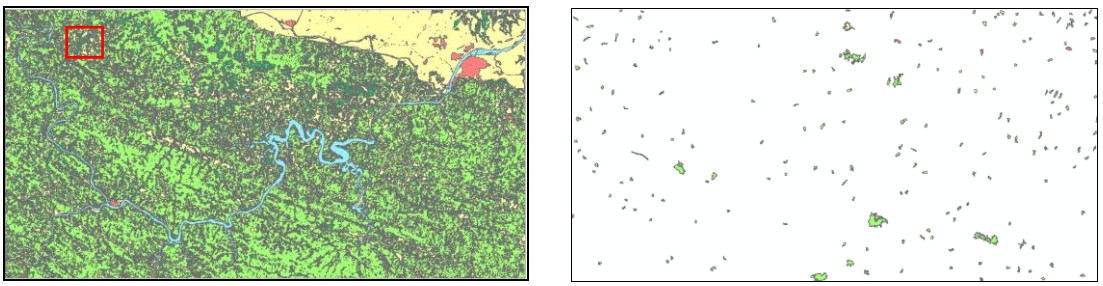
(a) Insertion P_2 to Sample Data shown in Figure 1(a) (b) the changed quadtree after inserting P_2

Figure 3. Index tree after inserting P_2 to the sample data shown in Figure 1(a)

Deletion is the reverse operation of insertion, it may cause the quadrant node union.

4. EXPERIMENTAL TEST AND COMPARISON

The improved quadtree index is implemented using VS2013. An incremental updating method is used to test the improved quadtree index for complex polygon objects. The methods presented in this paper was intensively tested using a land cover vector data of the Shanxi Province China. The area covers from north latitude 36.5412° to 38.3987°, east longitude 108.3241° to 110.8301°. The base state is classified from Landsat ETM+/TM 30m spatial resolution image in 2009. The test region is composed of 104230 parcels (Figure 4.a) originally. In order to get different test data, different the smallest area threshold value is used to merge small polygons to big ones. The base data polygons generally include many holes. The most complex polygon includes 5573 holes. The change-only data are the change parcels from 2009 to 2000 for the test region and include 181 new parcels (Figure 4.b) that were produced using a remote-sensing image change detection software. To test the performance of our improved method, a comparison between our method and the improved MX-CIF (Wei &, Tanaka, 2012) is made.



(a) Vector Land Cover Data in 2009 (b) the increment data from 2009 to 2000

Figure 4. Experimental test Vector Land Cover Data before Updating

The costs of constructing the improved quadtree index for different data is shown in table 1. In this experiment, the land cover objects are stored in the same file (i.e. Shapefile), ten is used as the threshold number of each node.

Number of polygons	Levels	Time	nodes
		second	
6585	6	0.6671	777
12166	6	1.6406	2217
26003	7	5.0704	3289
56087	8	16.9004	9689
104230	9	43.1691	16073

Table 1. Costs for the improved quadtree index construction

To test the query efficiency, 5 times experiments are made. 100 query points and 100 rectangles are produced in the test area randomly every time. The result is shown in table 2.

	Point Query		Window Query	
	Our method ms	MX-CIF ms	Our method ms	MX-CIF ms
1	3.261	7.300	56.309	102.695
2	2.942	6.552	55.674	81.362
3	3.441	6.413	51.864	79.996
4	3.032	7.859	64.634	92.702
5	2.882	7.021	51.239	84.249
average	3.112	7.029	55.944	88.201

Table 2. Query Time with Our Index Comparing to CIF Quadtree Index

As the purpose of this study is to improve the efficiency of incremental updating for land cover database with many complex parcels. The result is shown in table 3. In table 3, the maximum holes is the holes of the most complex polygon in the test data.

Polgs	Max holes	Delete polgs	Insert polgs	Update time (seconds)		Imprs
				MX-CIF	Our method	Times
6585	614	824	381	77.107	26.622	2.9
12166	1079	951	455	99.348	29.618	3.4
26003	2140	1120	595	144.397	34.626	4.2
56087	3817	1378	869	208.727	39.866	5.2
104230	5573	1609	1228	286.135	46.021	6.2

Table 3. Incremental Updating Time with Our Index Comparing to CIF Quadtree Index

In table 3, “Polgs” is the abbreviation of “polygons”. “Imprs” is the abbreviation of “improvements”, it means the improved times our method comparing to the CIF Quadtree, i.e. “Imprs = MX-CIF/Our method”. The cost time of updating process includes the time used to reconstruct the index structure and inclusion table.

From table 2 to 3, we can conclude that, comparing to MX-CIF quadtree index, the efficiency for query and updating are improved very much using our method. Especially, the updating efficiency has been improved to several times, furthermore, experimental results show that the greater the dada and complex, the more efficient our improved method is. When the polygon number of the base state is more than 100000, and the holes of the most complex polygon arrived 5573, the updating efficiency is improved more than 6 times.

5. SUMMARY AND DISCUSSION

An improved quadtree spatial index method is presented in this paper. In this method, the polygons intersect to the quadrant axes are stored in the five buckets of different level root nodes, and the polygons disjoint to all quadrant axes are stored in the leaf nodes. The inclusion relations between the complex polygons and their holes are stored in an inclusion table explicitly. The algorithms used to construct the spatial index

structure and inclusion relationship table are developed, the corresponding index operations (e.g., insert, delete and query) are presented. The effectiveness of the improved index is verified by an experiment of land cover data incremental updating. Experimental results show that the time efficiency of the incremental updating is significantly improved, and the improvement is increasing with the data size.Comparison to the existing methods, our method has the following characteristics:

1) The MBB of the holes of the complex polygons are stored explicitly, that can improved the filter efficiency for the updating process for the complex polygons.

2) The polygons intersect to the quadrant axes are stored in the five buckets of different level root nodes, which much decreased the duplicate storage of the objects cover neighbour quadrants, and improved the query efficiency.

3) The inclusion relations between the complex polygons and their holes are stored explicitly, which can improve the updating efficiency for complex polygons.

The objects usually form a complete coverage of the space in land cover (land use) database per se. While the objects can be stored in one layer (i.e. in one file or table), or stored in several different layers. However, the authors just implemented and test the models and algorithms for the case of the objects stored in one layer, i.e. the complex polygons and their holes are stored in the same layer. The other case, i.e. the complex polygons and their holes may stored in different layers will be studied in the further studies.

ACKNOWLEDGEMENTS

The work described in this paper was supported by the National Key Research and Development Program of China (NO.2016YFB0501403) and the National Natural Science Foundation of China (No. 41371366).

REFERENCES

Becker, C., Ostermann, J. Pahl, M., 2012. Mono-Temporal GIS Update Assistance System Based on Unsupervised Coherence Analysis and Evolutionary Optimisation. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. I-4, pp.233-238.

Murakami, S., Takemoto, T., Ito, Y., 2012. Data Updating Methods for Spatial Data Infrastructure That Maintain Infrastructure Quality and Enable Its Sustainable Operation. In: *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXIX-B4, pp. 29-33.

Guttman, A., 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts.

Sellis, T. K., Roussopoulos, N., Faloutsos, C., 1987. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. International Conference on Very Large Data Bases, Brighton.

Beckmann, N., Kriegel, H.-P., Schneider, R., et al., 1990. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles[C]. ACM SIGMOD international conference on management of data, Atlantic City, New Jersey.

Hjaltason, G. R., Samet. H., 2002. Speeding up Construction of Pmr Quadtree-Based Spatial Indexes. The VLDB Journal, 11(2): pp.109-137.

Kedem, G., 1982. The Quad-Cif Tree: A Data Structure for Hierarchical on-Line Algorithms. 19th Design Automation Conference, Las Vegas.

Wei, Y., Tanaka, S., 2012. Performance Improvement of Mx-Cif Quadtree by Reducing the Query Results. *International Journal of Computer Theory & Engineering*, 4(6), pp. 902-906.

Zimmermann, R., Ku, W. S., Chu, W. C., 2004. Efficient Query Routing in Distributed Spatial Databases. ACM International Workshop on Geographic Information Systems. ACM New York. pp. 176-183.

Philippe Rigaux, Michel Scholl, Agnes Voisard, 2002, Spatial Databases with Application to GIS, Academic Press, USA.