An Effective Algorithm for Spatial Query using Spatial Adaptive Grid Index Filter

Hao Lu[1,2*], Ershun Zhong[1,3], Xieliu Shu[3], Tianbao Wang[1,2], Shaohua Wang[1,2]
[1]State Key Laboratory of Resources and Environmental Information System
Institute of Geographic Sciences and Natural Resources Research, CAS
Beijing, China
[2]Graduate University of the Chinese Academy of Sciences, Beijing, China
[3]SuperMap Software Co., Ltd., Beijing, China
*Corresponding author, e-mail: luhao@supermap.com

Abstract—**Spatial query is the core function of GIS analysis, the query data is mostly geometric point, line, polygon which has the location information and formatted query condition is a variety of spatial relationships. We focus on the spatial query efficiency improvement with the query data in case of large and complex, and propos an effective spatial adaptive grid index filter algorithm for spatial query. It is different from the commonly used uniform grid index algorithm as it constructs the adaptive grid index of data using the spatial distribution information then it can radically improve the efficiency of spatial query. This algorithm contains four steps, including boundary grid index construction, internal grid index construction, primary spatial query and advanced spatial query. It is realized by C++ language and verified by experiments, the spatial query efficiency than the traditional uniform grid index algorithm has significantly increased.**

Keywords-multilevel; adaptive; spatial index; spatial query

I. INTRODUCTION

The spatial query in GIS can be divided into range query, thematic query, temporal query, network query, spatial pattern query and spatial simulation model [1-3]. In these spatial query models, the most widely used and basic ways is the spatial query for object geometry coordinate, because the other models are basically contain this mode, or can be transformed into this pattern, so we focus on the research of spatial query for object geometry coordinate, and the spatial query in later articles is in the default for this model. The current classification of spatial query, including spatial selection and spatial join, where spatial selection is the most basic and widely used spatial query model, it is based the given query window to search, get all of objects in query window. The spatial join query predicates are based on the relationship of space, to get the two data sets a collection of objects that satisfy the query conditions, and our research is mainly suitable for the spatial join query model.

The core processes of spatial query can be summarized as two steps, the initial filtration and the exact query, the initial filter process is through the general methods, such as the MBR filtering to get the candidate set which is possible to meet the conditions from the data, then calculated the geometric precision to get the final results. Therefore the efficiency of spatial query optimization focused on reducing as much as possible the number of candidate set. In addition, simplified description of the query object and by the approximate estimation model to find the optimum cost of the query execution plan are also commonly used in spatial query optimization.

Spatial index is by the description of spatial data stored in the media to establish correspondence relationship of the logical records and physical records, and the aim is to improve the efficiency of space data acquisition [4]. Because the actual geospatial data is involved in various shapes, so it is very difficult to use a unified index strategy to achieve the purpose of enhancing the efficiency of data retrieval for all of the data, such as the uniform grid, R tree and its variants, quad tree, KDB tree, CELL tree indexing, they are all difficult to achieve a universally applicable to all types of data processing, so the main discussion is focused on improvements in one or some of the specific types of data refinement processing.

With the basic method of spatial index the whole space is divided into different search area and find spatial entities with a certain order in these areas [5], the search area divided by different methods, spatial index can be divided into three types, based on point regional index method, based on polygon regional index method and based on three-dimensional body regional division index method [5]. Meanwhile, the current spatial index study also focus on the combination of spatial index and storage structures, high-latitude spatial index research and use, index efficiency and retrieval efficiency balancing, and indexing of moving objects [6].

II. ALGORITHM OVERALL

Because the common spatial query data is related to three types including point, line, polygon, and spatial join query is usually binary relations between elements for matching and filter, if we consider the query object and objects be queried types are equivalent when they are exchanged, the combinations of spatial join query mode includes: point-point, point-line, point-polygon, line-line, line-polygon and polygon-polygon a total of six. In order to ensure the universal of algorithm design and application, we choose one of the most complex mode polygon-polygon to give a specific algorithm description and experimental verification, the other five models can be simplified by the polygon-polygon mode.

The proposed adaptive grid index spatial query algorithm (AGI-SQ) includes four procedures, namely the boundary grid index construction, the internal grid index construction, the primary spatial query and advanced spatial query. The first two processes can be considered as adaptive grid index build process. Because the research mode is polygon-polygon and we should be strict distinction between the boundaries and internal of the query object in the spatial query process, so we need to build the boundary grid index and internal grid index. When this algorithm is applied to the line related mode the internal grid index construction could be cut down. After the two processes can be considered as the use of spatial index and accurate calculation of the spatial relationship, considering the limited size of the index and the index storage and loading efficiency in spatial query process, we only store the necessary information the object ID of the polygon in the index. So we could only determine the spatial position relationship with the width of the grid index level in the primary spatial query procedure. When using the index information to determine the spatial relationship efficient filter out for most of the query objects, a small number of objects which cannot be determined may need the accurate calculation with the advanced spatial query procedure and obtain the complete spatial query results. Therefore, the overall process of the algorithm is in Fig. 1.

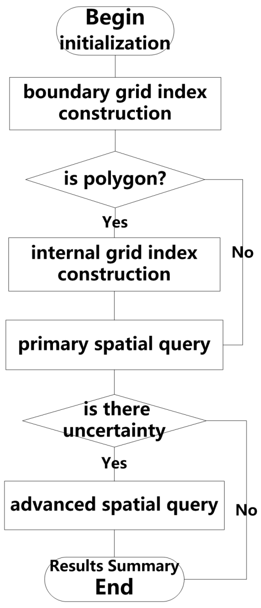

Figure 1. Algorithm flow chart

III. Data Structure

In order to facilitate the process later of algorithm description for the four core procedure, we give the declaration of some common parameters and variables.

The core structure of adaptive multi-level grid index is a pointer array, denoted *AdaptiveGrids*, every element in the pointer array corresponds to a basic grid index structure called *AdaptiveGridBase*, In the grid structure *AdaptiveGridBase* we store the object boundary information which across this grid denoted *BoundaryObjIndexs*, and the polygon object which contains this grid denoted *ContainedByRegions* , which *ContainedByRegions* is an array stores the object ID, *BoundaryObjIndexs* as an array of objects, each element object is *ObjectInfo*, which contains the object ID and object boundary points information *PointPos*. As shown the overall of the adaptive multi-level grid index is in Fig. 2.

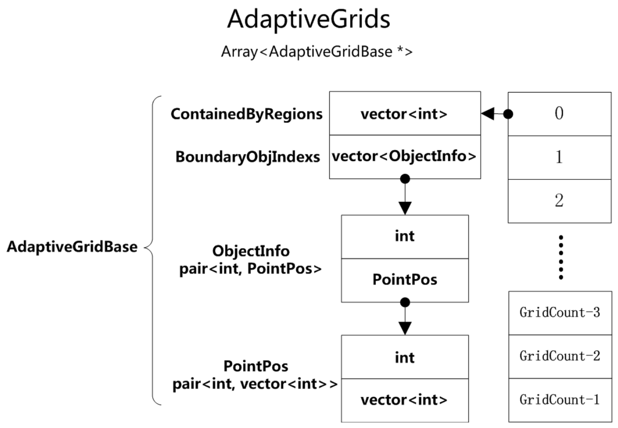

Figure 2. Adaptive grid idex core structure

Multi-level grid index levels count denoted *G*. Range of query data is the scope of the outermost layer of the index denoted *R*, it is a rectangular structure. Because in the process of boundary grid index construction and advanced spatial query, it is related to the vector data determine relationships by the intersection calculation, requiring the introduction of vector data tolerance denoted *T.* We agreed the scope of level 0 grid is *R*, while the upper level 1 grid for the grid (i.e. level 0 grid) by the cross, that the grid number of Level 1 is 2^2, and the length and width both as half of the grid level 0, so we agreed the total of *G* level grid number is *GridCount.* Each level grid has its own width and length, denoted H_i and W_i, $i \in [0, G-1]$.

The algorithm initialization is the process for computing basic control parameters of the adaptive grid index, including the index core structure *AdaptiveGrids* initialization, and initialization parameters, including: index range *R*, the index level count *G*, effective vector data tolerance *T*., the total number of index grid *GridCount*, each level index grid length and width H_i and W_i, $i \in [0, G-1]$.

IV. ALGORITHM DESCRIPTION

A. *Boundary Grid Idex Construction*

As the control parameters have been calculated in the initialization process, then we need to calculate through all the boundary of query objects to construct boundary grid index, we call it *SeekCrossedGridIDs* process. In the process, we calculate the boundary arc bottom level grid index according to the starting node and ending node coordinates, and the corresponding grid index structure should be recorded in the polygon object ID and the node number in the boundary arc. We define the starting node *pntStart* and the ending node *pntEnd*, than the bottom level grid index can be calculated as follows:

$$Index_{start} = \left\lfloor \frac{pntStart.y - R.bottom}{H_i} \right\rfloor * 2^{G-1} + \left\lfloor \frac{pntStart.x - R.left}{W_i} \right\rfloor (i = 0)$$

$$Index_{end} = \left\lfloor \frac{pntEnd.y - R.bottom}{H_i} \right\rfloor * 2^{G-1} + \left\lfloor \frac{pntEnd.x - R.left}{W_i} \right\rfloor (i = 0)$$

If the two points corresponding to the same index value, the information is directly recorded in this boundary grid index, if *pntStart* and *pntEnd* corresponding to different index values, we need use the breadth-first search to calculate the bottom level boundary grid index which the arc across according to the starting node index value and ending node index value. Construction effect of the boundary grid index is in Fig. 3.

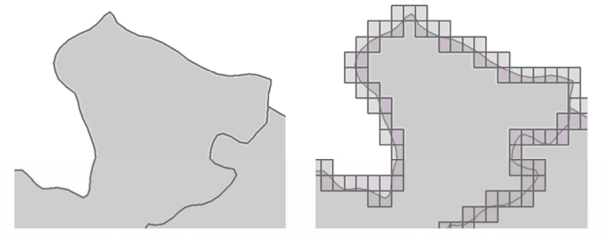

Figure 3. Boundary grid index construction

After finishing calculates the entire boundary arcs, a crucial step operation needed to be done, the grid index update, which we call *UpdateGridGrade* process. In this process, the multi-level grid should be updated respectively from bottom to up and from up to bottom, if find some level grid existences the data, the other upper level grids which corresponds by this grid should also be updated within the from bottom to up update process, if find some gird grade is less than the current grade, then the grade of its four sub-grids are set to the grid grade.

B. *Internal Grid Idex Construction*

After the boundary grid index construction, if dealing with polygon objects, we also need to construct the internal grid index. First, we should mark up the grid which be crossed by boundary arcs based on the results of the boundary grid index, then use the breadth-first search through each grid to find the grid which be contained by the query polygon object, and the search bound is the marked boundary index value. Note that because our algorithm is to build multi-level adaptive grid, so during the breadth-first search we should spans multiple levels of grid index, and the aim is to use as little as possible grid index to express the internal of polygon object, the internal grid index construction effect is in Fig. 4

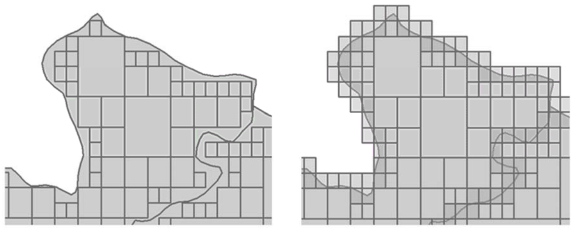

Figure 4. Internal grid index construction

C. *Primary Spatial Query*

After the above two spatial index build process, the polygon object has been constructed the boundary grid index and internal grid index, then we can do the spatial query according to the query predicates. The process for determining spatial relationship is divided into two parts, primary spatial query and advanced spatial query. With the primary spatial query we can quickly filter out most of the match (or not match) query object be recorded by the adaptive grid using the object ID index information. After filtering through this process some individual object cannot be determined the location relationship, and then for advanced spatial query to determine its precise spatial relationship.

We use the 9-intersection model to determine the spatial relationship of objects, and use the polygon contain query to illustrate the process. The 9-intersection model of contain relationship as shown below, which indicates that A is the query object, and B is the object be queried.

TABLE I. CONTAIN RELATIONSHIP MODEL

		B (Be Queried Object)		
		Internal	***Boundary***	***External***
A (Query Object)	***Internal***	T	*	*
	Boundary	*	*	*
	External	F	*	*

In this model dim(I(A)∩I(B)) = T, it means the intersection of interval A object and interval B object is non-empty. dim(E(A)∩I(B)) = F，it means the intersection of boundary A object and interval B object is empty.

We define BGC_i is the number of boundary grid of the object with ID i, IGC_i is the number of internal grid of the object with ID i. We also define BI_i is the corresponds number of grid which object's boundary index grid and query object internal index grid, and we have similar II_i, IB_i, BB_i.

```
The 9-intersection model based on the description above, the contain relationship can be expressed as:
If $BI_i = BGC_i$ && $II_i = IGC_i$
    Then meet contain relationship
Else if $IB_i$ ! = 0 and $BI_i + II_i + BB_i = BGC_i + IGC_i$
    Then the need to do advanced spatial query
Else does not meet contain relationship
```

Because When the boundary and internal grid of be queried object are all in the internal grid of the query object, it meets the contain relationship. If the intersection of be queried object's internal grid and query object's boundary grid is empty, and has $BI_i + II_i + BB_i = BGC_i + IGC_i$, then it means be queried object is very small and near the boundary of query object. It also means both of them have the same boundary grid, so it need for advanced spatial query to refinement judge.

Because we can determine the boundary and internal between query object and be queried object in linear time through the multi-level adaptive grid index, so it is from the fundamental guarantee for the algorithm's efficiency in the query.

D. Advanced Spatial Query

When the end of the primary spatial query, if there are some objects which spatial relationships cannot be directly determined by the adaptive grid index would require the use of advanced spatial query to determine, and the main use of advanced spatial query is the arc ID information stored by the first step of boundary grid index construction, accurate to determine the intersection line.

We define the be queried objects as R_1, the query object as R_2, Define the boundary arcs of R_1 as L_1, and the boundary arcs of R_2 as L_2. The accurately determine algorithm described in pseudo code as follows:

Calculate the same boundary index grid intersection point of L_1 and L_2 as *Pnt.*

If *Pnt* is within the L_1, Then do not meet the contain relationship

Else if *Pnt* is the endpoint of L_1, but the mid-point of L_1 is not in the R_2, Then do not meet the contain relationship

Else meet the contain relationship.

If all boundary arcs of R_1 do not intersect with boundary arcs of R_2, Then if the endpoint of R_1 is in R_2, meet the contain relationship

Under normal circumstances, only a small number of objects need to determine the spatial relationship by advanced spatial query.

V. Experiment

To verify the effectiveness of this algorithm, we use C++ language to develop a prototype system and the compare test with the traditional uniform grid index spatial query algorithm (UGI-SQ). Experimental environment for a frequency of 2.6 GHz dual-core processor PC, RAM is 3GB. We take 12 sets of real data to test in order to ensure the reliability of experiment.

A. Time Complexity Analysis

Because the algorithm mainly includes the boundary grid index construction, the internal grid index construction, the primary spatial query, so the time complexity of this algorithm is also determined by the three parts. In the boundary grid index construction process the index construction of boundary arc will be calculated respectively, we suppose n is the number of polygon object arcs, then the number of calculating intersecting arcs is $k \times n$, the time complexity of this process is $O(n)$. In the process of internal grid index construction, including boundary index marking and internal index grid searching, so its time complexity is $O(n + l^2)$, where l is the bottom of the grid X direction (or Y direction) number of cells. If the number of be queried objects is p, and the number of query objects is q, the primary spatial query process is both internal and boundary grid determine, its time complexity is $O(pq)$. Therefore, the algorithm time complexity is $O(n + l^2 + pq)$.

B. Results and Analyses

TABLE II. CONTRAST TEST RESULTS

		UGI-SQ		AGI-SQ			
No.	***Queried Count***	***Query Time***	***Query Result***	***Index Build***	***Index Load***	***Query Time***	***Query Result***
1	100	0.56	37	1.17	0.31	0.13	37
2	400	1.61	149	1.89	0.42	0.33	149
3	800	2.95	285	2.94	0.49	0.67	285
4	1600	5.58	582	4.42	0.61	1.14	582
5	2400	7.89	857	6.66	0.69	1.92	857
6	3200	10.70	1153	8.39	0.70	2.45	1153
7	4800	15.66	1732	10.77	0.80	3.58	1733
8	6400	20.91	2307	13.58	0.88	5.22	2308
9	8000	26.77	2924	16.72	0.95	6.41	2926
10	9600	32.63	3581	19.69	1.00	8.17	3583
11	11200	38.45	4258	23.47	1.05	10.09	4261
12	12800	44.30	4907	27.12	1.09	12.05	4904

a. The time unit is second.

We set TU_i as the time used in the query which the UGI-SQ algorithm in section $i \in [1, 12]$ group of test data, TA_i as the time used in the query, including the boundary and internal grid index construction time, which the AGI-SQ algorithm in section $i \in [1, 12]$ group test data. From the table above we see that the first three group test data have $TU_i < TA_i$ $i \in [1, 3]$, the small amount of data UGI-SQ algorithm performance is better than AGI-SQ algorithm, However starting from the fourth set of test data have $TU_i > TA_i$ $i \in [4, 12]$, which increases with the size of the data, AGI-SQ algorithm gradually showed a better efficiency of spatial query, or AGI-SQ algorithm to better adapt to the expansion of scale of the data, as shown in Fig. 6.

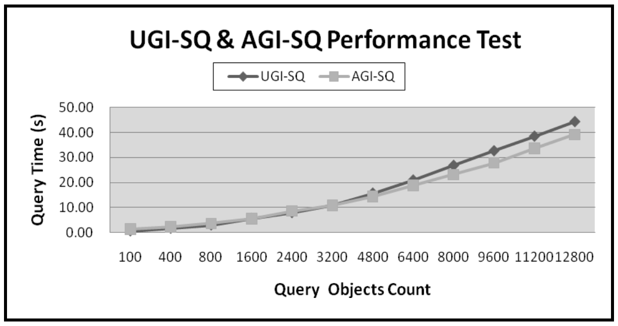

Figure 5. Query performance test 1

On the other hand, the AGI-SQ algorithm index construction result can be stored as intermediate results, and can be used in some application scenarios which spatial query frequently used, we can load the intermediate results index information to further enhance the AGI- SQ algorithm performance of spatial query, we define TQA_i as the time used in the query, including the index load time, which the AGI-SQ algorithm in section $i \in [1, 12]$ group of test data, then all the test data has $TU_i > TQA_i \; i \in [1, 12]$, its increase results as shown in Fig. 7.

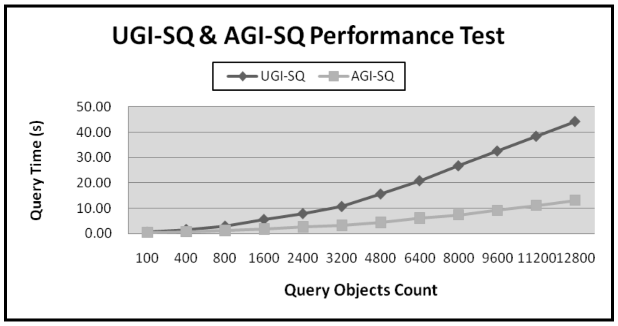

Figure 6. Query performance test 2

When compare with the effectiveness of query results, we can see the first 6 sets of test results are consistent between AGI-SQ and UGI-SQ, when the query data is greater than 5000 there are a number of individual results are inconsistent, but the error rate is always less than 0.1%.

Therefore, seen from the above experiment, AGI-SQ algorithm has a good adaptability for expansion of the size of data, the overall performance superior to the traditional UGI-SQ algorithm, and it is more suitable for large scale data applications under the frequent spatial query, and the result is authentic.

VI. CONCLUSION

In this paper, we propose the adaptive grid index spatial query (AGI-SQ) algorithm based on traditional uniform grid index spatial query (UGI-SQ) algorithm, introduce the main flow of the algorithm, data structure and the four core process, including the boundary grid index construction, the internal grid index construction, the primary spatial query and advanced spatial query, and analyze the time complexity of this algorithm at the same time, and make multiple sets of different sizes real data experiments to demonstrate the effectiveness of this algorithm. Follow-up study is mainly for core data structure and algorithm optimization and parallel index construction approach to further enhance the performance and stability of the algorithm.

ACKNOWLEDGMENT

The authors would also like to express appreciations to colleagues in our laboratory for their valuable comments and other helps.

REFERENCES

[1] A. Kumar, W. Muhanna, and R. Patterson, "Mean-variance analysis of the performance of spatial ordering methods," International Journal of Geographical Information Science, vol. 12(3), 1998, pp. 269-289.

[2] ESRI. Understanding GIS-the ARC/INFO Method, Redland: ESRI Press Inc, 1996.

[3] L. Lin, "From spatial database querying to GIS spatial object querying," Journal of Wuhan Technical University of Surveying and Mapping, vol. 20(Supplement), 1995, pp. 93-97.

[4] C. Yan and X. Zhao, "Review of GIS Spatial Index Approach," Geography and Geographic Information Science, vol. 20(4), 2004, pp. 23-26.

[5] J. Jiang, G. Han, and J. Chen, Navigation Geodatabase, Beijing: Science press, 2003, pp. 90-106.

[6] C. Cheng, "Research on multi-scale spatial index method of vector data," Information Science of Wuhan University academic journal, vol. 34(5), 2009, pp. 597-601.