BCG - Boletim de Ciências Geodésicas - On-Line version, ISSN 1982-2170
http://dx.doi.org/10.1590/S1982-21702014000100011

DWSI: AN APPROACH TO SOLVING THE POLYGON INTERSECTION-SPREADING PROBLEM WITH A PARALLEL UNION ALGORITHM AT THE FEATURE LAYER LEVEL

DWSI: uma abordagem para a solução do problema de alastramento de intersecção de polígonos com um algoritimo de união paralela ao nível da camada de característica

JUNFU FAN [1, 2]
CHENGHU ZHOU [1]
TING MA [1, *]
MIN JI [3]
YUKE ZHOU [1]
TAO XU [1, 2]

[1] State Key Laboratory of Resources and Environmental Information System, Institute of Geographic and Nature Resources Research, Chinese Academy of Sciences, Beijing 100101, China
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] College of Geomatics, Shandong University of Science and Technology, Qingdao, 266590, China
{fanjf; mting}@lreis.ac.cn

ABSTRACT

A dual-way seeds indexing (DWSI) method based on R-tree and the Open Geospatial Consortium (OGC) simple feature model was proposed to solve the polygon intersection-spreading problem. The parallel polygon union algorithm based on the improved DWSI and the OpenMP parallel programming model was developed to validate the usability of the data partition method. The experimental results reveal that the improved DWSI method can implement a robust parallel task partition by overcoming the polygon intersection-spreading problem. The parallel union algorithm applied DWSI not only scaled up the data processing but also speeded up the computation compared with the serial proposal, and it showed a higher computational efficiency with higher speedup benchmarks in the treatment of larger-scale dataset. Therefore, the improved DWSI can be a potential approach to parallelizing the vector data overlay algorithms based on the OGC simple data model at the feature layer level.

Keywords: Dual-way Seeds Indexing Method; Polygon Intersect-spreading; Parallel Union; Task Partition.

RESUMO

Um método de indexação de semeamento bidireccional (dual-way seeds indexing - DWSI), baseado em árvore-R e no modelo simples de característica de Consórcio Geoespacial Aberto (Open Geospatial Consortium - OGC), foi proposto para solucionar o problema de alastramento de intersecção de polígonos. O algoritmo de união paralela de polígono baseado no método DWSI melhorado e o modelo paralela de programação OpenMP foi desenvolvido para validar a usabilidade do método de partição de dados. Os resultados experimentais revelaram que o método DWSI melhorado pode implementar uma partição paralela de tarefas robustas, superando o problema de alastramento de intersecção de polígonos. O algoritmo de união paralela aplicado ao DWSI não apenas levou o processamento de dados a uma escala maior, como também acelerou a computação em comparação com a proposta serial, demonstrando uma maior eficiência computacional com referências de aceleração mais altas no tratamento de conjuntos de dados em larga-escala. Portanto, o método DWSI melhorado pode ser uma abordagem potencial para a paralelização e otimização dos algoritmos de sobreposição de dados de vetor baseados no modelo de dados simples OGC no nível de camada de característica.

Palavras-chave: Método de Indexação de Semeamento Bidireccional; Alastramento de Intersecção de Polígonos; União Paralela; Partição de Tarefas.

1. INTRODUCTION

Advances in multi-core processors can greatly improve the user experiences of computer systems by handling more work in parallel (GEER, 2005). However, the efficiency of traditional serial algorithms cannot be accelerated directly by the use of multicore processors because these algorithms use only one core rather than all of the cores on the board for each computation, which is a substantial waste. Spatial analysis algorithms in Geographical Information Systems (GIS) also face the same problems. Parallel computing is an effective approach to speeding up existing serial algorithms, improving the utilization rate of multi-core systems, and achieving high performance geo-computing in GIS (TURTON et al., 1998; CLARKE, 2003; SUTTER, 2005).

Polygon overlay is a key category of spatial analysis operations in GIS, and polygon clipping is one of the basic problem which always can be computationally intensive (GOODCHILD, 1977; WANG, 1993; SHI, 2012; AGARWAL et al., 2012). The Weiler-Atherton algorithm (WEILER and ATHERTON, 1977), the Vatti algorithm (VATTI, 1992), and the Greiner-Horman algorithm (GREINER and HORMANN, 1998) are three acknowledged polygon clipping algorithms that can generate results in a limited amount of time. Although optimizations proposed by scientists (KIM et al., 2006; LIU et al., 2007; MARTINEZ et al., 2009) can lead to improvements to the extensively validated polygon clipping algorithms, the acceleration effects are limited (ZHAO and ZHOU, 2013), especially when handling large datasets. Higher computational efficiency can be obtained directly and the handling or maintaining of a large spatial dataset can become more efficient by the parallelization of serial algorithms. Data partitioning and function partitioning are effective parallel programming techniques for most applications; the former best suits applications for which loops must perform the same operations on large sets of data (WANG, 1993), the later best suits the developing of parallel spatial work flow. We chose data parallelism with the expectation that the codes could be reused conveniently by other polygon overlap algorithms, such as the merge algorithm and the symmetrical difference algorithm.

The polygon union algorithm at the feature layer level has a wide utilization (WONG, 1997), such as calculating the coverage areas of two intermixed types of vegetation or determining the annual change in a category of land use. Because all the geometry parts of the polygon clipping results will be collected during the union operation, it’s difficult to parallelize it for the uncertainty of intersecting between polygons of different layers. The polygon intersection-spreading problem is the main factor predisposing the uncertainty and resulting in the many-to-many mapping relationships between polygons of the overlapping layers. Some data decomposition approaches, such as the regular grids and feature sequences based data division methods are cannot address the polygon intersection-spreading problem which lead to the parallelization barrier to the polygon union algorithm.

To solve these problems, a new data partition method called dual-way seeds indexing method (DWSI) was designed to eliminate potential intersections between groups of data decomposition results and to implement the parallel polygon union algorithm. In the next two sections, the related work and spatial index knowledge base will be introduced, and then, a detailed explanation about the polygon intersection-spreading problem and presentation of the DWSI method is provided.

2. RELATED WORK

Classic overlay analysis in GIS is mainly used to derive new and/or implied spatial and attribute information required by users, and such operations are usually based on pre-established topological relations. Mineter proposed a parallel vector data overlay architecture, and a series of parallel algorithms were deployed on a software framework called Topology-Stitching-Output (TSO) (MINETER et al., 1999 and 2003). In Mineter’s parallel architecture, some pre-processing steps, including polygon cutting and strip division, are necessary before parallel tasks start; the last and complex step is topology rebuilding for parallel overlay results.

However, polygon cutting and topological building are time-consuming, and topological building for a large dataset may fail. Moreover, topological relationships are not a concern of the final user in some overlap analysis tasks.

There are a variety of data decomposition strategies for parallelizing the polygon overlay operation without topological relationships, such as data partitioning by regular grids (WAUGH and HOPKINS, 1992) and strips (MINETER et al., 1999), feature sequences (AGARWAL et al., 2012), and expected balanced workloads (ZHAO and ZHOU, 2013). Shi (2012) has pointed out that different polygon overlay operations involve different relationships between polygons from the base layer and overlap layer. For polygon intersection and difference operations, each polygon establishes a 0- or 1-to-many relationship to those polygon features that are within the overlay layer. However, for polygon union and symmetrical difference operations, there are a series of many-to-many relationships that caused by the polygon intersection-spreading problem must be determined. This problem will lead to a phenomenon that disjoint polygons in the subject layer could be assigned to the same group for potential intersections with the same polygon in the overlap layer. The regular grids based data partition method needs complex and time-consuming polygon cutting and result stitching processes; the feature sequences based method and expected balanced workloads based method cannot solve the polygon intersection-spreading problem.

The union-find algorithm can provide disjoint subsets, which is always used in the determination of network connectivity and image processing. Elements partitioned into a subset are connected (overlapped for polygons) with each other but disjointed with elements belonging to other subsets. The union-find algorithm using the tree data structure elegantly which lead to an extremely low time complexity (TARJAN, 1975 and 1984; CORMEN et al., 2001). Considering the polygon intersection-spreading problem in the polygon union algorithm, it can be parallelized by dividing all of the polygons into disjoint subsets and perform the same union process on multiple CPU cores. In this research, a data division method called a DWSI is designed to generate disjoint subsets of polygons and the polygon intersection-spreading problem can be solved as a result.

3. MEMORY LIMIT AND R-TREE

One important principle for parallelizing vector data overlay algorithms was that the efficiency bonus derived from parallel computing should exceed the time and complexity costs caused by the task partitioning and the results stitching (MINETER, 2003). However, the memory of commercial computers is limited and, in some spatial analysis operations, it is impossible to load all of the features into memory at one time when handling massive vector datasets. Frequent read/write commands on data entities will become bottlenecks, and consequently, the algorithms and programs running on such systems cannot obtain the maximum concurrency performance (DEL ROSARIO, 1993). Therefore, efficient data partitioning methods and economic spatial indexing data structures are necessary for the data parallelism to overcome the memory limits.

The spatial division results of the grid index and the Quad-tree (FINKEL and BENTLEY, 1974) index do not have relevance to the distribution of the features. Correspondingly, the spatial divisions of BSP-tree (FUCHS et al., 1980), KD-tree (BENTLY, 1975), R-tree (GUTTMAN, 1984), and their variants have strong relevance to the distributions of the features. Although the data structure of the grid index is simple, for sparsely distributed features, it will bring redundant data. Once the structure is established by the grid index or the Quad-tree, it is difficult to be extended dynamically. However, the abilities of dynamic expansion, efficient spatial searching, a simple data structure, and polygon geometry supporting are necessary to a possible candidate of spatial indexing data structures for the parallel polygon union algorithm. R-tree implemented in memory can be extended dynamically in a convenient way and can support the spatial indexing of a multi-dimensional dataset. The search efficiency of R-tree is higher than that of the grid index and the Quad-tree, and the complexity is lower than its variants, such as R*-tree (BECKMANN et al., 1990), HILBERT R-tree (KAMEL et al., 1994), and CR-tree (KIM et al., 2001).

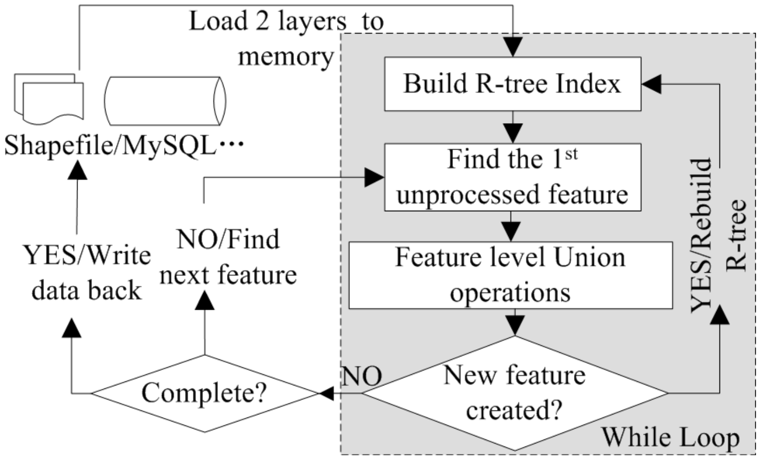
R-tree accepts an MBR as a spatial query filter and produces an identifier of qualified data objects by comparing the overlap situation of spatial extension between the input MBRs and the filter (GUTTMAN, 1984). Each MBR is stored in memory by four 8-bit double precision floating-point numbers and more than 3×10^7 MBRs can be saved in 1 GB of memory. However, the time and memory cost of loading the same number of features into memory are definitely much higher. The time cost of one overlap comparison operation of MBRs in memory is very low; as a result, the system maintains a reasonable range of time costs for building an R-tree indexing structure and executing spatial query operations for millions of vector features. It is therefore a feasible approach to implement an immediate spatial query and parallel task decomposition for whole datasets based on R-tree. However, any spatial indexing data structure with characters of high spatial query efficiency, ease of maintenance and extension, and lower memory requirements can replace it.

4. PARALLEL STRATEGIES IN THE UNION ALGORITHM

Vector data overlay algorithms with the essence of Boolean operations between spatial features are one of the core categories of functions of spatial analysis in GIS. In the parallel environment of OpenMP, overlap operations should be treated differently and should be considered fully according to their specific logical characters and the distinct traits between spatial vector data and generic data. The logical flow of the serial polygon union algorithm is presented in Figure 1.

Function division and computing data decomposition are two commonly used methods for parallelizing serial vector data analysis algorithms (GEIST et al., 1994; BRESHEARS, 2009). For the algebra operation properties of vector data overlay algorithms, their parallelization based on functional decomposition must be tightly coupled with detailed data structures. Parallel algorithms developed in this way could address their defects on generality and portability. We therefore choose the data decomposition approach to parallelize the vector data overlay algorithms.

Figure 1 - Logical flow of the serial union algorithm.



The union algorithm is characterized by the dynamic building of the R-tree index, continual geometrical splitting and attribute joining. These intensive operations are executed and finished in a *while* loop in the serial union algorithm, but because of the unpredictable iteration times of the *while* loop, it is difficult to parallelize it. In a parallel computing paradigm, the minimization of the correlations between computing data items could provide maximum concurrent computing efficiency (LIN and SNYDER, 2008). The principle is applied in this study to implement a parallel union algorithm by starting up a new thread for each feature group. The features calculated by a new thread have nothing to do with features in other groups in the parallel stages, like the disjoint-sets in the Union-Find algorithm, and the *while* loop in the serial union algorithm is executed in each new thread. Figure 2 shows the logical flow of the parallel union algorithm.

The key to the success of this approach is to develop an efficient data decomposition method to solve the intersection-spreading problem. The intersection-spreading problem refers to a phenomenon that disjoint polygons in the subject layer could be divided into one subset for the potential intersecting with the same polygon in the overlapping layer at the same time. Take polygon layers listed in Figure 3 as an example to illustrate the phenomenon.

As shown in figure 3, *A1* and *A2* are two disjoint polygons from the same layer, but they should be grouped into the same subset because their MBRs intersect with the same polygon *B1* in another layer, and *B1* will be sent to the same group as well. The meaning of intersection is not only refers to the intersection between polygon entities but also their MBRs. The uncertain intersection-spreading effect revealed in the parallel union algorithm is the main difficulty and makes it different from other parallel overlay algorithms, such as polygon intersect and difference.

Figure 2 - Logical flow of the parallel union algorithm.

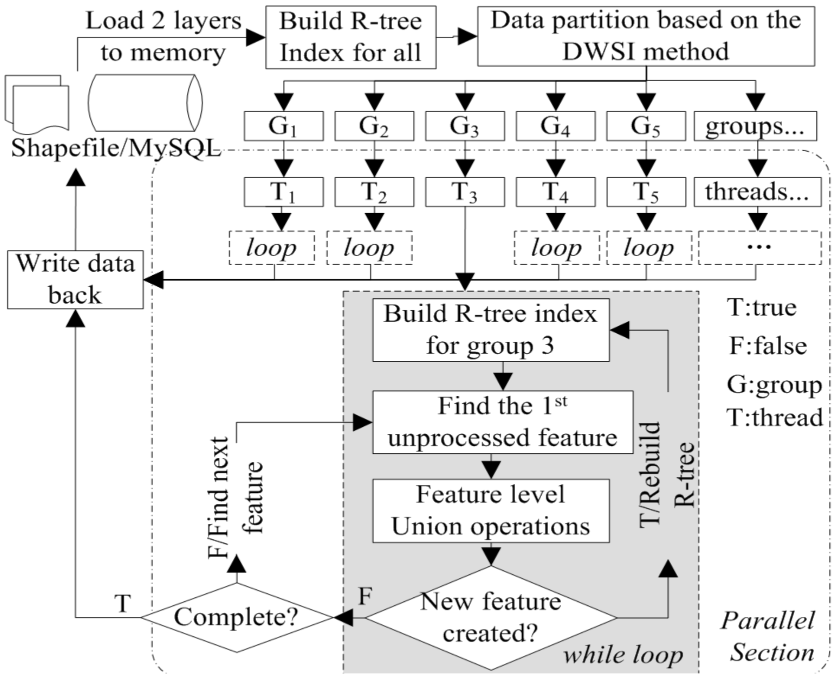
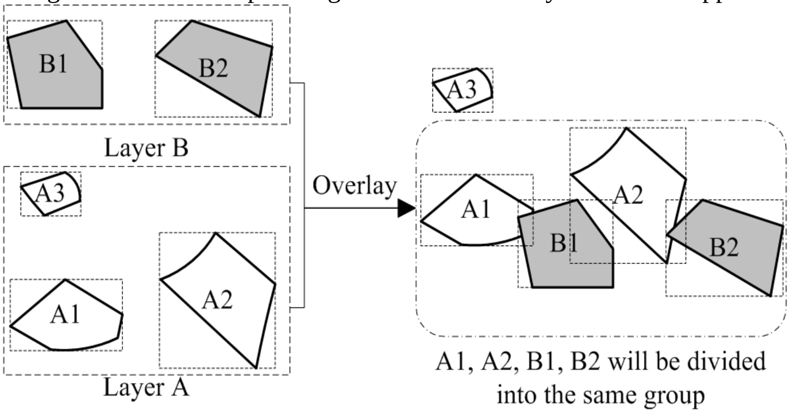


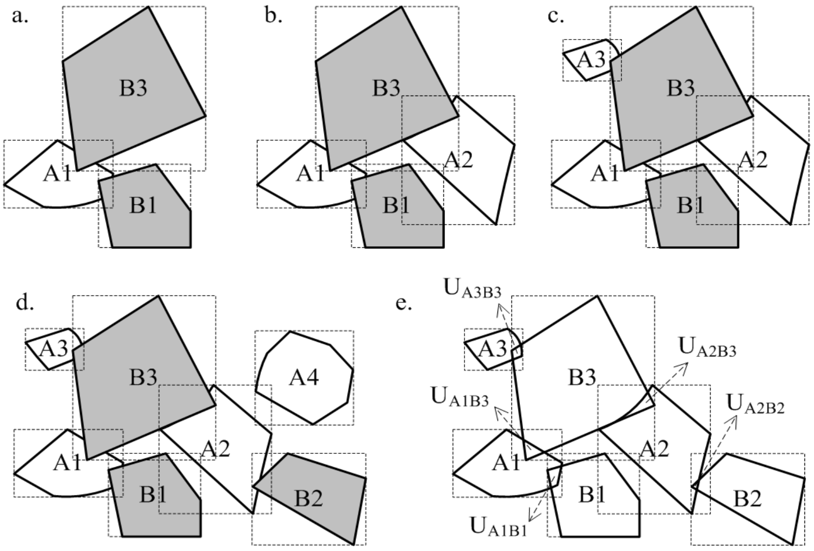
Figure 3 - Intersect-spreading effect when two layers are overlapped.



5. THE DWSI METHOD

Without loss of generality, we assume that there are no intersections between polygons in each input feature layer (actually, polygons have the possibility to intersect with each other in the same layer based on non-topological data models). However, for the intersection-spreading effect between the layers, the intersections of the polygons from different layers are unpredictable, leading to a complicated data decomposition process.

Figure 4 - Execution process and result of DWSI.



The Figure 4 shows four polygons *A1-A4* in layer A and three polygons *B1-B3* in layer B. The two layers maintain the same coordinate system and have no internal intersections. *A1*, *A2* and *A3* do not intersect in layer A, but they intersect with polygon *B3* in layer B at the same time when the two layers overlap. Moreover, *B3* and *B2* should be grouped into the same group in order to be processed together in the parallel stage, because *A2* intersects with them simultaneously. The DWSI method can achieve these goals and implement the cross-layer grouping of intersected features. The searching process of DWSI between overlapped polygon layers, as shown in figure 4-(d), is presented in figure 4-(a-e). This method is implemented by introducing two polygon searching queues, two R-tree indexing structures, and a collection container of search results.

The method is composed of six steps:

1) Build two R-tree index structures, RT_A and RT_B, for all polygons of the two input layers, and allocate memory for two search queues;

2) Insert the identifier and MBR of the first polygon *A1* in layer A into queue Q_A, and make it the search seed, as shown in figure 4-(a). Execute a spatial search in RT_B with the MBR of *A1* as a spatial filter, and polygons *B1* and *B3* will be found in layer B;
3) Add the identifiers and MBRs of *B1* and *B3* into Q_B as seeds and make *A1* be a non-seed state in Q_A. Transfer to Q_B if all of the seeds in Q_A are iterated;
4) For each seed in Q_B, execute a spatial search in RT_A iteratively, and then, make it be a non-seed state after searching and adding the identifiers and MBRs of newly found features to Q_A as new seeds. For example, after searching by *B1* and *B3*, *A2* and *A3* can be found and should be added to Q_A (figure 4-b and c) and then transferred back to Q_A.
5) For each new seed in Q_A, execute a spatial search iteratively in RT_B; then, make it be a non-seed state after searching and adding the identifiers and MBRs of newly found features to Q_B as new seeds. Ignore the features that already exist in the two queues when adding a newly found feature to them. Repeat steps 4) and 5) in Q_A and Q_B until there are no newly found features and no new seeds are found. The loop process will be terminated, and the merged results of Q_A and Q_B are the final results of one independent group as described in section 3.
6) Select the next feature in layer A, which is not grouped yet, and repeat steps 1) to 5) and a new group will be found. Finally, DWSI will be ended when all of the features in the two layers are grouped. Groups contain only one feature will be distinguished and excluded from the next step of the calculation.

The pseudo code of the DWSI algorithm, implemented according to the six steps, is presented in table 1 (do not consider intersections in the same layer).

Table 1 - Pseudo code of the DWSI algorithm.

```
RTSpatialIdx *siRTreeA = new RTSpatialIdx(),*siRTreeB = new
RTSpatialIdx();
std::vector<std::vector<int> > fid_vec;
std::queue<int> QA, QB;
std::vector<int> tmp_fid_vec;
for (each FID in the two layers){
    if (FID is processed) continue;
    else QA.push(FID);
    while(QA or QB is not empty ){
        while(QA is not empty){
            siRTreeB->spatialSearch(First MBR in QA, a_search_list);
            for (each FID in a_search_list)
                QB.push(FID);
            QA.pop();
            }
            while(QB is not empty){
                siRTreeA->spatialSearch(First MBR in QB, b_search_list);
                for (each FID in b_search_list)
                    QA.push(FID);
                QB.pop();
        }}
        fid_vec.push_back(tmp_fid_vec);
}
```

Take the features in the two layers as shown in figure 4-(d), as an example; the execution graph and the changes in Q_A and Q_B of DWSI are described in table 2.

Table 2 - Execution flow and change details in each queue of DWSI.

Q_A		**Q_B**	
Operations	**Status**	**Operations**	**Status**
Insert seed *A1* into Q_A	***A1***[a]		*NULL*
Q_A search starts	***A1***		*NULL*
Spatial search in RT_B by *A1;* Get *B1*, *B3*	*A1*		*NULL*
Insert seed *B1* into Q_B	*A1*		***B1***
Insert seed *B3* into Q_B	*A1*		***B1***, ***B3***
Q_A search end; Transfer to Q_B	*A1*	Q_B search starts	***B1***, ***B3***
	A1	Spatial search in RT_A by *B1;* Get *A1*, *A2*, *B3*	*B1*, ***B3***
A1 already exists Ignore *A1*	*A1*, ***A2***	Insert seeds *A1*, *A2* into Q_A *and B3* into Q_B	*B1*, ***B3***
	A1, ***A2***	*B3* already exists Ignore *B3*	*B1*, ***B3***
	A1, ***A2***	Spatial search in RT_A by *B3;* Get *A1*, *A2*, *A3*, *B1*	*B1*, *B3*
	A1, ***A2***	Insert seeds *A1*, *A2*, *A3* into Q_A *and B1* into Q_B	*B1*, *B3*
A1, *A2* already exist Ignore *A1*, *A2*	*A1*, ***A2***, ***A3***	*B1* already exists Ignore *B1*	*B1*, *B3*
Q_A search starts	*A1*, ***A2***, ***A3***	Q_B search ends Transfer to Q_A	*B1*, *B3*
Spatial search in RT_B by *A2;* Get *B1*, *B2*, *B3*	*A1*, *A2*, ***A3***		*B1*, *B3*
Insert seeds *B1*, *B2*, *B3* into Q_B	*A1*, *A2*, ***A3***	*B1*, *B3* already exist Ignore *B1* and *B3*	*B1*, *B3*, ***B2***

Spatial search in RT_B by *A3;* Get *B3*	*A1*, *A2*, *A3*		*B1*, *B3*, ***B2***
Insert seed *B3* into Q_B	*A1*, *A2*, *A3*	*B3* already exists Ignore *B3*	*B1*, *B3*, ***B2***
Q_A search ends Transfer to Q_B	*A1*, *A2*, *A3*	Q_B search starts	*B1*, *B3*, ***B2***
	A1, *A2*, *A3*	Spatial search in RT_A by *B2;* Get *A2*	*B1*, *B3*, *B2*
A2 already exists Ignore *A2*	*A1*, *A2*, *A3*	Insert seed *A2* into Q_A	*B1*, *B3*, *B2*
	A1, *A2*, *A3*	Q_B search ends	*B1*, *B3*, *B2*
Q_A and Q_B both search end. Merge Q_A and Q_B to one group which will contain: *A1*, *A2*, *A3*, *B1*, *B3*, *B2*			
Start searching for a new group, Reinitialize Q_A and Q_B Find the next unprocessed feature in layer A: *A4*			
Insert seed *A4* into Q_A	***A4***		*NULL*
Q_A search starts	***A4***		*NULL*
Spatial search in RT_B by *A4* Get nothing	*A4*		*NULL*
Finally *A4* will be ignored for it is not intersected with others in layer B.			
END			

[a] Bold characters represent seed features in the moment.

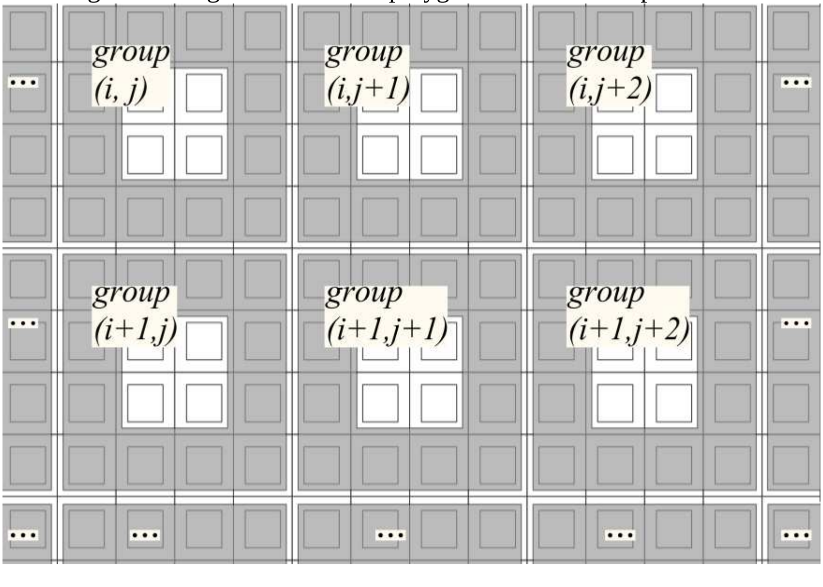
Table 2 shows that all of the features in Layer A will be iterated as R-tree searching seeds, because DWSI must guarantee that all of the possible intersections between two overlapped layers should be detected; the number of R-tree searches could be less than the count of features in the two layers because there is no need to traverse the remaining features in Layer B after all of the features in Layer A are iterated. The time cost of DWSI is in positive proportion to the number of features in the two overlapped layers and is highly related with the intersection situations between them. However, the time cost of DWSI is limited by the inherent defects of R-tree, whose search efficiency is in a linear drop with the growing number of features when the MBRs of the features are overlapped seriously in the R-tree indexing structure (BECKMANN, 1990). Hence, DWSI will encounter a similar problem in addressing such problems.

Overlapping operations based on a non-topological data model cannot guarantee that there are no intersections between the polygons that are in the same layer. If intersections between polygons in the same layer should be detected, then only one R-tree rather than two trees should be constructed to maintain all of the MBRs of the polygons in the two layers, and a simple modification of DWSI can meet the requirements.

6. EXPERIMENTS AND ANALYSIS

Keeping the computing resources unchanged, two main factors that affect the computational efficiency of the parallel union algorithm were discussed in this study, including the quantity of polygons and the total amount of points of two polygons involved in an atom operation. To begin with, special tests were conducted to analyze the data division efficiency variation of DWSI with different numbers of polygons. Then, a series of experiments and statistics were performed to analyze the relationships between parallel computational efficiency variation and the quantity of features, while the number of points in each polygon was kept constant. Finally, the variation in the time cost of one single intersection operation implemented by Vatti's algorithm (a sub-operation of the union algorithm) with the quantity of points was analyzed to determine the effects exerted to the polygon union algorithm. We use a DELL Optiplex 990 computer (i7-2600 quad-core CPU) to perform the experiments. The experimental polygons are regular distributed rectangles and each one with a hole. The detailed overlapping situation and groups divided by the DWSI method are presented in figure 5.

Figure 5 - Regular distributed polygons used in the experiments.



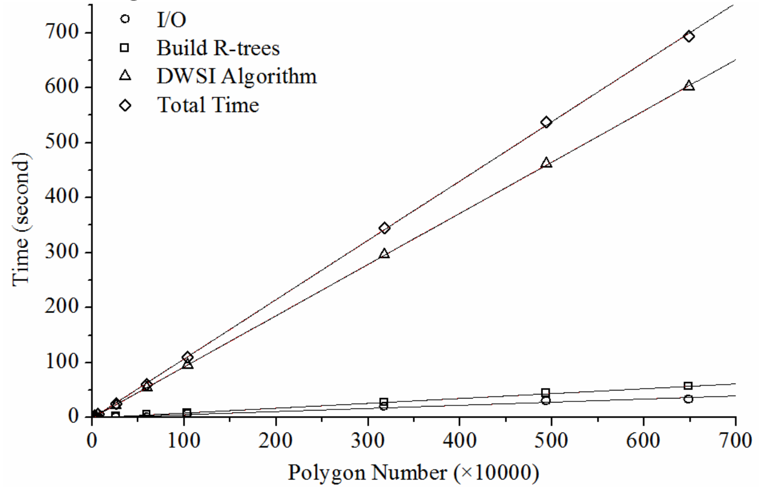
Four time costs indicators, including I/O, building and initializing of R-trees, data decomposition of DWSI, and the total time costs are statistically analyzed in each experiment, and the results are listed in table 3 and figure 6.

Table 3 - Fitting results of the four time costs metrics.

Metrics	**Regression Functions**	R^2
IO	$y = 0.0569x - 0.0345$[a]	0.9836
Build R-Trees	$y = 0.0884x - 0.2467$	0.9996
Our Algorithm	$y = 0.9316x - 0.4865$	1.0000
All Time	$y = 1.0792x - 0.7678$	0.9999

[a] *y*: the time costs (second); *x*: the amount of polygons divided by 10^4.

Figure 6 - Fitting curves of the four time costs indicators of the DWSI method.



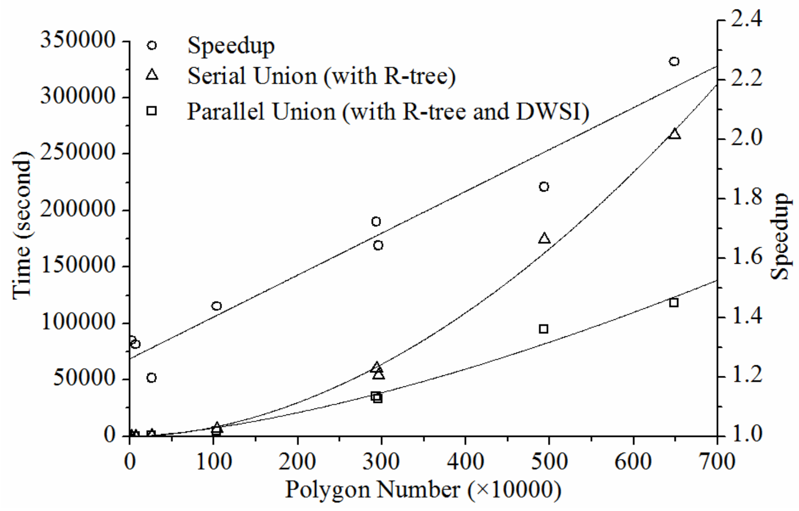
The calling of *spatialSearch* function is the key atom operation in the DWSI algorithm which means that only one time of calling of *spatialSearch* function is needed by one feature, so the computational complexity of the DWSI algorithm is *O(n)*, *n* is the feature count in the two layers. As shown in figure 6, the four indicators all present a linear uptrend with the growing number of operated polygons. Linear functions and coefficients of determination in table 3 are the fitting results of the four time costs indicators, which are cohering with the complexity analysis. It can be concluded that R-tree based DWSI demonstrates a linear complexity when searching and performing decomposition of either a small or large number of polygons. The fitting curves remain an obviously linear uptrend in the time costs when the number of polygons reaches an order of magnitude of 10^6.

DWSI was applied to the implementation of the parallel union algorithm, and experiments were conducted to statistically compare the time costs between the parallel and serial polygon union algorithms at different magnitude orders of polygon amounts. The speedup of the parallel union algorithm was calculated, and the results are presented in table 4 and figure 7.

Table 4 - Time costs of serial and parallel union algorithms and the speedup results.

Polygon Amount	Serial Time/s	Parallel Time/s	Speedup
16 200	3.268	2.469	1.324
64 800	34.096	26.044	1.309
259 200	427.774	357.419	1.197
1 036 800	6 537.462	4 545.155	1.438
2 934 726	60 216.041	34 961.461	1.722
2 962 656	54 150.452	32 971.671	1.642
4 937 760	174 293.500	94 694.790	1.841
6 485 401	266 818.100	117 830.600	2.264

Figure 7 - Statistical regression results of the union algorithms.



Compared with the serial union algorithm, the parallel version achieved calculation efficiency to a certain degree. As shown in figure 7, the acceleration effects of the data decomposition mechanism based on DWSI will become more significant with an increasing number of polygons. The algorithm even reached a speedup benchmark of *2.264* when the number of polygons was approximately 6.5×10^6, which means that the calculated time costs of the parallel union algorithm obtained by DWSI is reduced by more than *55%* than the serial union algorithm. At the same time, the time cost variation trends and the mathematical model with a varying number of polygons is not changed. Our results show that the parallel computational efficiency variation with the number of polygons still coincided with the power-law regression model, which is the same with the serial polygon union algorithm. However, as shown in table 5, the exponent of the power-law function is 1.8379 for the parallel version, which is smaller than that of the serial union algorithm (1.9156).

Table 5 - Serial and parallel fitting results of time costs and speedup variations with the number of polygons.

Metrics	**Fitting Function**	R^2
Serial Union Time	$y = 1.0413x^{1.9156}$	0.9987
Parallel Union Time	$y = 0.9313x^{1.8379}$	0.9994
Speed Up	$s = 0.0014x + 1.2623$	0.9525

y: the time costs(second); *s*: the speedup; *x:* the polygon amount divided by 10^4.

Obviously, the time costs of the DWSI method are much lower than the time costs of the serial and parallel union algorithms when handling the same amount of polygons. Therefore, DWSI will not cause time complexity increase to the parallel union algorithm. The parallel union algorithm is implemented based on the Vatti polygon clipping algorithm whose time cost variation with the total number of points in two overlapped polygons is not a linear model and should be analyzed statistically. We therefore conducted experiments to analyze the variation of time cost of one single intersection operation (which is an atom operation of the union algorithm) with the volume of points holding by the overlap polygons. The results are listed in table 6 and figure 8.

Figure 8 - Fitting curve of the time cost of the intersection operation implemented by Vatti's polygon clipping algorithm.

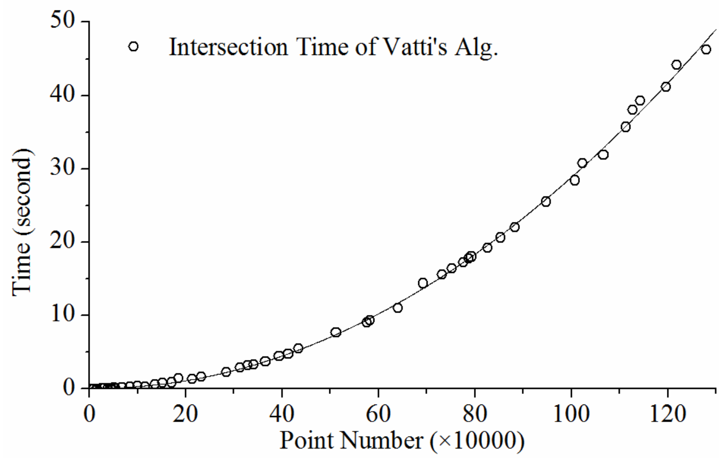


Table 6 - Time costs fitting results of the intersection operation implemented by Vatti's polygon clipping algorithm.

Models	**Fitting Function**	R^2
QP[a]	$y = 0.0030x^2 - 0.0089x + 0.1229$	0.9989
POW[b]	$y = 0.0105x^{1.6792}$	0.9869

y: the time costs(second); *x:* the quantity of polygons divided by 10^4.
[a] quadratic polynomial regression model; [b] Power-law regression model.

The time costs variation of Vatti's algorithm with the total number of points in the overlapped polygons is most consistent with the quadratic polynomial regression model (figure 8 and table 6). However, Vatti's algorithm also coincides well with the power-law regression model (the POW model in table 6), which is small but close to the corresponding numbers of regression models of the serial and parallel polygon union algorithms. In the worst case, for a subject layer with *m* polygons and a overlapped layer with *n* polygons, the time complexity of Vatti's algorithm can be $O((p\text{-}2)^2)$, *p* is the average point amount contained by two overlapping polygons (GREINER and HORMANN, 1998), and the time complexity of the union algorithm is $O(m \times n \times (p\text{-}2)^2)$, meanwhile, the DWSI algorithm (with the R-tree construction process) is $O(m+n)$. Therefore, the performance of Vatti's algorithm is the main relevant factor that should be responsible for the computational efficiencies of the union algorithm, especially for the parallel one, rather than the R-tree construction and the DWSI method. Furthermore, the performance decrease of the union algorithm will be much more significant when the *p* parameter grows than *m* or *n* grows.

An extreme situation is that all of the features may be grouped into a single group, and the DWSI will fail on data partition when handling such datasets. This will occur when all or most of the MBRs intersect with each other, which is also caused by the intersection-spreading problem. In this case, the performance of the parallel polygon union algorithm will be same or even poorer than the serial version. We improved the DWSI algorithm to solve the problem completely by employing of segmentation polygons between divided groups and the restriction of expect group size. The results of experiments and discussion are presented in the next section.

7. DEFICIENCIES AND IMPROVEMENTS IN THE DWSI

One important deficiency of the DWSI algorithm is that the parallel union algorithm based on it will not work or lose the balance of task loads in most instances if all of the features in the two layers are assigned to one single group, for example, all of the polygons in figure 9-(a). We improved the DWSI method to solve this problem by recording segmentation features repeatedly. Segmentation features are special features that intersect with more than one feature in different groups, such as polygon *B2* in figure 9-(b). The key to our improvements is that the expected group size should be specified in advance. A new group will be established when the number of features reaches the specified group size, and features in the two queues will be recorded as segmentation features rather than be processed as seeds. The non-intersection parts of the segmentation features will be abandoned temporarily in the parallel union process; all of them will be recalculated at the end of the parallel union algorithm. Figure 9 is the logic flow of the parallel union algorithm based on the improved DWSI method.

We conducted several experiments with residential region data of Changchun city to verify the parallel robustness of the union algorithm based on the improved DWSI method. Each of the two input geographical layers contains 3959 polygons, and they are listed in figure 10 and figure 13.

The comparison results between parallel union algorithms based on the improved and un-improved DWSI methods using polygon layers in figure 10-(a) are presented in figure 11.

It can be deduced from the results presented in figure 11 that the performance of the un-improved DWSI based parallel union algorithm is similar to the serial algorithm. The reason is that most of the features in the two layers are sent to one group which resulting the parallel failure. With the application of our improved DWSI method, both of the serial and parallel union algorithms are speeded up definitively. The direct reason for the low efficiency of our union algorithm based on the un-improved DWSI is that too many non-intersected polygons are involved in the iterative overlay calculating process in the *while* loop. Therefore, both the serial and parallel union algorithm can be benefited from the improved DWSI method. We conducted additional experiments to find out the best group size for the two feature layers shown in figure 10; the results are listed in figure 12.

Figure 9 - Logic flow of parallel union based on the improved DWSI method.

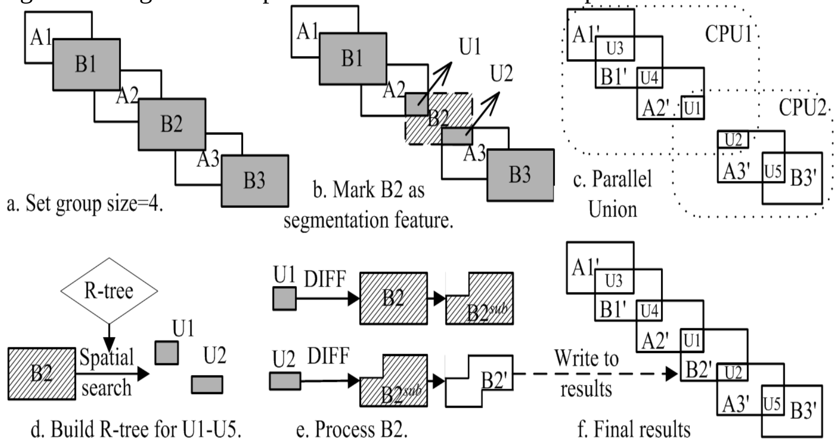


Figure 10 - Real geographical data and experiment results, (**a**. part of residential areas of Changchun city and its shifted data; **b**. parallel union results).

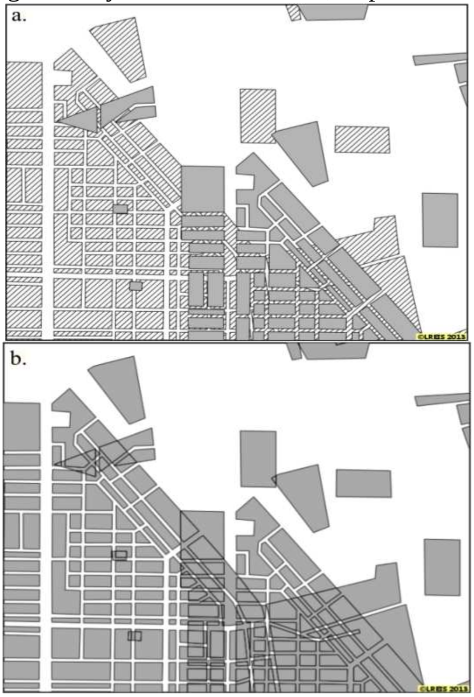
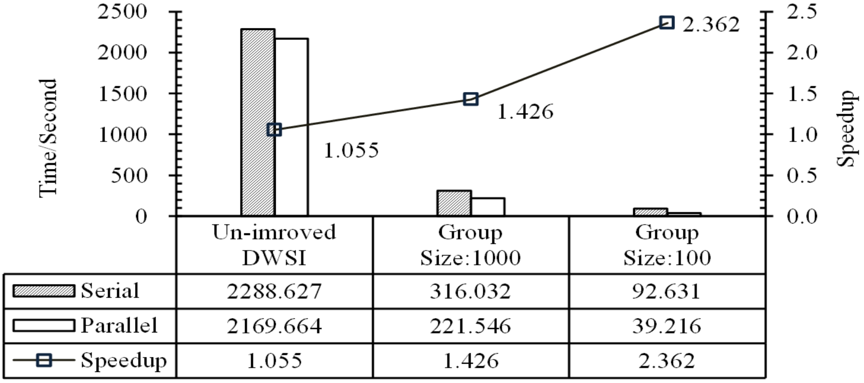
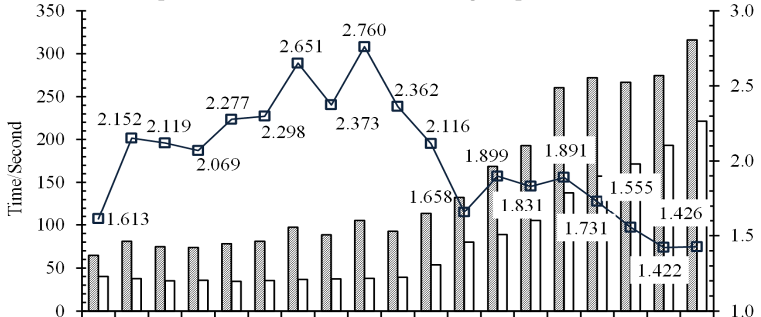


Figure 11 - Efficiency comparison between serial/parallel union algorithms based on un-improved and improved DWSI methods.



	Un-imroved DWSI	Group Size:1000	Group Size:100
▨ Serial	2288.627	316.032	92.631
□ Parallel	2169.664	221.546	39.216
─□─ Speedup	1.055	1.426	2.362

Figure 12 - Efficiency comparison between serial/parallel union algorithms based on improved DWSI with different group sizes.



GroupSize×100:	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	2	3	4	5	6	7	8	9	10
▨ Serial	65	81	75	74	78	81	97	89	105	93	113	133	169	193	261	272	267	274	316
□ Parallel	40	38	35	36	34	35	37	37	38	39	54	80	89	105	138	157	172	193	222
—⊡— Speedup	1.6	2.2	2.1	2.1	2.3	2.3	2.7	2.4	2.8	2.4	2.1	1.7	1.9	1.8	1.9	1.7	1.6	1.4	1.4

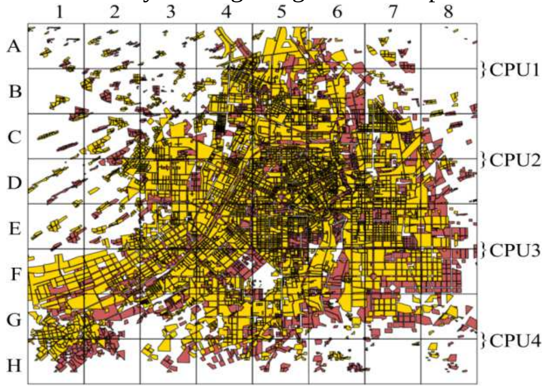
The experimental results show that the highest speedup reached 2.760 when the expected group size was specified to 90. When the group size was specified to 50, the parallel union algorithm obtained the highest efficiency with the speedup of 2.277, and both of the values are higher than the records in figure 7. However, the determination of the optimal group size for two overlapped feature layers may need many experiments or experiences.

Although the improved DWSI can lead to higher efficiency than the old one for both of the serial and parallel union algorithms, several special cases should be concerned carefully. First and foremost, intersected segmentation features will lead to additional features with the same geometries, which will bring topological errors to the result layer. We use points comparing to find out the same geometries and to eliminate such errors from the final results. Besides, because the non-intersection parts of the segmentation features will be abandoned, all of the intersection parts between a segmentation feature and non-segmentation features should be calculated at one time. Last, groups that contain only one segmentation feature should be skipped.

8. COMPARISON WITH REGULAR GRID BASED DATA DIVISION

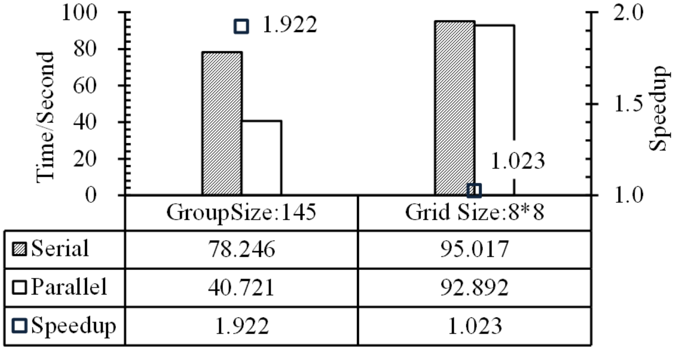
Regular grid or strip based data decomposition methods are classic approaches to reduce the relationships between data groups and to realize parallelization (MINETER, 1999 and 2003). We divided the feature layers in figure 10 into 64 parts with regular grids, as shown in figure 13; features that cross more than one grid would be cut into multiple pieces by grid lines. After cutting, the features in the different grids will not intersect with each other anymore, and the parallel union algorithm can be implemented conveniently.

Figure 13 - Data division by 8*8 regular grids for the parallel union operation.



On average, there are approximately 145 polygons in each of the 64 groups in figure 13. We therefore specified the expected group size to be 145 and conducted experiments to compare the efficiencies of the union algorithms based on the regular grid data division and our improved DWSI data division. The experimental results are listed in figure 14.

Figure 14 - Efficiencies comparison between serial/parallel union algorithms based on the improved DWSI method and regular grid data division.



	GroupSize:145	Grid Size:8*8
▨Serial	78.246	95.017
□Parallel	40.721	92.892
□Speedup	1.922	1.023

The results reveal that both of the serial and parallel union algorithms based on DWSI are faster than the corresponding ones based on the regular grid data division. The parallel speedup of the regular grid-based union algorithm is much lower than 1.922, which is the speedup of the DWSI-based parallel union algorithm. The parallel efficiency of the DWSI method based union algorithm is improved by 56% than the regular girds based parallel union algorithm, and the gap between serial algorithms is about 18%. We believe that the reason is that the uneven spatial distribution of the features leads to an unbalanced task partition for the regular grids, which also shows that the DWSI method can give more parallel adaptabilities and robustness to the GIS algorithms when handling datasets that have different spatial distributions. Furthermore, the parallel union algorithm based on DWSI avoided destruction of input data and sewing of the results data, which is much easier to implement than the regular grid-based data decomposition approach.

9. CONCLUSIONS

In this study, we proposed a data decomposition method called a DWSI to solve the intersection-spreading problem and implement the parallel polygon union algorithm at the feature layer level. The experimental results verified the usability of DWSI for implementing computing-intensive parallel algorithms in GIS. The results show that the polygon union algorithm can be parallelized and accelerated based on DWSI, and the efficiency improvement will be much more significant with a larger amount of data. The time complexity of the DWSI is much lower than the Vatti algorithm. As a result, the DWSI method does not increase the time costs and complexity of the union algorithms.

We improved the DWSI method to completely solve the problems of parallel failure and load imbalance, which are caused by a situation that most features are divided into the same group. The experimental results show that the improvements that are based on a specification of the expected group size and the recording of the segmentation features enhanced the parallel robustness of the union algorithm. The improved DWSI method brings a degree of acceleration for both of the serial and parallel union algorithms. For the element amounts in the disjoint subsets generated by the improved DWSI method are under control, load balancing can be achieved by assigning approximately equal quantity of polygons to each thread. Compared with the regular grid based data partition method, the optimized DWSI algorithm can lead to 56% performance improvement to the parallel union algorithm and about 18% to the serial one.

Therefore, the improved DWSI method can solve the problem of polygon intersection-spreading and implement data decomposition independently in a more robust way compared with regular grid and feature sequence based data division approaches. Based on the DWSI method, some serial spatial analysis algorithms can be parallelized, such as the polygon merge algorithm and polygon symmetrical difference algorithm. This data partition algorithm has been applied successfully with MPI to parallel the polygon overlay operations in a cluster environment. We assume that the DWSI algorithm can be a potential approach to implementing data decomposition and thereby to parallelizing the polygon union algorithm and some other similar overlap algorithms in GIS at the feature layer level.

ACKNOWLEDGEMENTS

This study was supported by National Key Technology R&D Program (No. 2011BAH06B03 and No. 2011BAH24B10). Additional supports were provided by Chinese Academy of Sciences (No. KZZD-EW-07). We would like to gratefully thank the anonymous reviewers for their insightful and helpful comments to improve the manuscript.

BIBLIOGRAPHICAL REFERENCES

AGARWAL D., PURI S., HE X., et al. A System for GIS Polygonal Overlay Computation on Linux Cluster – An Experience and Performance Report. In: *IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, Proceedings, p. 1433-1439, 2012.

BECKMANN N., KRIEGEL H. P., SCHNEIDER R., AND SEEGER B. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In: *1990 ACM SIGMOD Conference on Management of Data*, Proceedings, 19(2), p. 322-331, 1990.

BENTLEY J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9), p. 509-517, 1975.

BRESHEARS C. *The Art of Concurrency: A Thread Monkey's Guide to Writing Parallel Applications*. O'Reilly Media Inc. Sebastopol, CA, USA. p.304, 2009.

CLARKE K. C. Geocomputation's Future at the Extremes: High Performance Computing and Nanoclients. *Parallel Computing*, 29(10), p. 1281-1295, 2003.

CORMEN T., LEISERSON C., RIVEST R., et al. Chapter 21: Data structures for Disjoint Sets, *Introduction to Algorithms (Second ed.)*, MIT Press, Cambridge, MA USA, p. 498-524, 2001.

DEL ROSARIO J. M., BORDAWEKAR R., CHOUDHARY A. Improved Parallel I/O via a Two-phase Run-time Access Strategy. *ACM SIGARCH Computer Architecture News*, 21(5), p. 31-38, 1993.

ENVIRONMENTAL SYSTEMS RESEARCH INSTITUTE, Inc. *ESRI Shapefile Technical Description*, 2012. Accessed 28/11/2012. http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

FINKEL R. A. and BENTLEY J. L. Quad Trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, 4(1) , p. 1–9, 1974.

FOSTER I. *Designing and Building Parallel Programs*. Addison-Wesley Publishing Company, Reading, MA, USA, p.430, 1995.

FUCHS H., KEDEM Z. M., NAYLOR B. F. On visible surface generation by a priori tree structures. In: *SIGGRAPH '80 of the 7th annual conference on Computer graphics and interactive techniques*, Proceedings, p. 124-133, 1980.

MARTINEZ F., RUEDA A. J. and FEITO F. R. A new algorithm for computing Boolean operations on polygons. *Computers & Geosciences*, 35(6), p. 1177-1185, 2009.

GRASS DEVELOPMENT TEAM. *GRASS 7 Programmer's Manual,* 2013. Accessed 28/10/2013. *http://grass.osgeo.org/programming7/index.html*.

GEER D. Chip makers turn to multicore processors. *Computer*, 38(5), p. 11-13, 2005.

GEIST A., BEGUELIN A., DONGATRA J., JIANG W., et al. *PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Network Parallel Computing*, MIT Press, Cambridge, MA, USA. p. 299, 1994.

GOODCHILD M. F. *Statistical aspects of the polygon overlay problem. In: Harvard Papers on Geographic Information Systems,* Addison-Wesley Publishing Company, Reading, MA, USA, 6, p. 1-30, 1977.

GREINER G., HORMANN K. Efficient clipping of arbitrary polygons. *ACM Transactions on Graphics*, 17(2), p. 71-83, 1998.

GUTTMAN A. R-TREES: A Dynamic Index Structure for Spatial Searching. In: *1984 ACM SIGMOD Conference on Management of Data*, Proceedings, 14(2), p. 47~57, 1984.

KAMEL I., and FALOUTSOS C. HILBERT R-tree: An Improved R-tree using Fractals. In: *20th International Conference on Very Large Data Bases*, Proceedings, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, p. 500-509, 1994.

KIM D. H., KIM M. An Extension of Polygon Clipping To Resolve Degenerate Cases. *Computer-Aided Design & Applications*, 3(1-4), p. 447-456, 2006.

KIM K., CHA S. K., and KWON K. Optimizing Multidimensional Index Trees for Main Memory Access. In: *2001 ACM SIGMOD international conference on Management of data*. Proceedings, p. 139-150, 2001.

LIN C., SNYDER L. *Principles of Parallel Programming*. Addison-Wesley Publishing Company, Reading, MA, USA, p. 352, 2009.

LIU Y. K., WANG, X. Q., BAO S. Z., et al. An algorithm for polygon clipping, and for determining polygon intersections and unions. *Computers & Geosciences*, 33(5), p. 589-598, 2007.

MINETER M J., DOWERS S. Parallel processing for geographical applications: A layered approach. *Journal of Geographical Systems*, 1(1), p. 61-74, 1999.

MINETER M. J. A software framework to create vector-topology in parallel GIS operations. *International Journal of Geographical Information Science*, 17(3), p. 203-222, 2003.

SHI X. *System and methods for parallelizing polygon overlay computation in multiprocessing environment.* US Patent, Pub. No.: US 2012/0320087 A1, Dec. 20, 2012.

SUTTER H. A Fundamental Turn Toward Concurrency in Software. *Dr. Dobb's Journal*, 30(3), p. 16-19, 2005.

TARJAN R. Efficiency of a Good But Not Linear Set Union Algorithm. *Journal of the ACM*, 22(2), p. 215-225, 1975.

TARJAN R., van Leeuwen J. Worst-case analysis of set union algorithms. *Journal of the ACM*, 31(2), p. 245-281, 1984.

TURTON I., OPENSHAW S. High-performance computing and geography: developments, issues, and case studies. *Environment and Planning: A*. 30, p. 1839-1856, 1998.

VATTI B. R. A Generic Solution to Polygon Clipping. *Communications of the ACM*, 35(7), p. 56-63, 1992.

WANG F. A parallel intersection algorithm for vector polygon overlay. *Computer Graphics and Applications, IEEE*, 13(2), p. 74-81, 1993.

WAUGH T. C. & HOPKINS S. An algorithm for polygon overlay using cooperative parallel processing, *International Journal of Geographical Information Systems*, 6(6), p. 457-467, 1992.

WEILER K., ATHERTON P. Hidden surface removal using polygon area sorting. In: *SIGGRAPH '77 Proceedings of the 4th annual conference on Computer graphics and interactive techniques*. Proceedings, New York: ACM Press, p. 214~222, 1977.

WONG K. M., STRECKER E. W., STENSTROM M. K. GIS to Estimate Storm-Water Pollutant Mass Loadings. *Journal of Environmental Engineering*, 123(8), p. 737–745, 1997.

ZHAO S. S., ZHOU C. H. Accelerating polygon overlay analysis by GPU. *Progress in Geography*, 32(1), p. 114-120, 2013.

(Recebido em agosto de 2012. Aceito em novembro de 2013).