International Journal of Hybrid Information Technology
Vol. 9, No. 5 (2016), pp. 21-30
http://dx.doi.org/10.14257/ijht.2016.9.5.03

Automatic Polygonization Algorithm on Parallel and Graph Model

Liang Wu [1, 2], Zhanlong Chen[1], Min Hao[1] and Zhong Xie [1, 2,*]

[1]College of Information Engineering, China University of Geosciences, Wuhan, 430074, China
[2]National Engineering Research Center for GIS, Wuhan, 430074, China
**College of Information Engineering, China University of Geosciences, Wuhan, 430074, China*
Tel/fax: +86-133-8750-5800
wuliang@cug.edu.cn (L. Wu), xiezhong@cug.edu.cn (Z. Xie)

Abstract

The polygon construction is the emphasis and difficulty for constructing the topological relationship of spatial data. Many scholars continuous study and innovate for the algorithm. The traditional computational process mainly involves three steps in algorithms: (1) Determining the relationship between adjacent arcs. (2) Searching polygons. (3) Determining the topological relationship between arcs and polygons. With the increase of the number of data and the need to improve the automatization, speed and complexity of algorithm, the serial algorithm has become more and more difficult in polygon construction and is often more time consuming. The algorithm of polygon construction is meeting challenge with the development of computer technology. When processing the large-scale linear data, an efficient strategy to reduce the time of complex operations in algorithm has not been proposed in spatial data field. We propose a novel algorithm to construct polygons automatically in the parallel environment base on the new IT technology. The key of the algorithm is paralleling the more time consuming operation (search, sort. etc.) and improving the speed of polygon construction. According to the characteristics of directed rings in graph model, we construct topological polygons. The experimental results on multicore computers show that the algorithm is efficient parallel performance for polygon construction.

***Keywords**: graph model, polygonization, parallel computing*

1. Introduction

Spatial Data is made up of spatial object. Spatial object consists of point, line, area, rectangle, plane, volume and high dimensional data including the time. The examples of spatial data include the location across the region such as city, rivers, roads, county town, state, crop coverage, mountains, *etc*. It is very effective for environmental monitoring, space, city planning, resource management and the application for GIS [5, 11].

There are a lot of algorithms for polygon construction in the serial environment. Yan and Cheng (2000) proposed a fast algorithm of topological polygon auto-construction based on azimuth calculation. Li *et al*. (2005) made improvements based on it. Wang (2002) proposed the raster algorithm on creation of topological relationship of polygons. Luo *et al*. (2004) proposed an algorithm to create topological relationship based on closed coordinate chain data. Chen *et al.* (2010) proposed polygon overlay analysis algorithm based on monotone chain and STR tree in the simple feature mode. Iwerks and Mitchell (2012) proposed spiral serpentine polygonization of a planar point set. Michikawa and Suzuki (2013) proposed polygonization of volumetric skeletons with junctions. Polygon

[*] Corresponding Author

construction in a parallel environment has mainly studied by Bestul (1992). Under the general scanning model [2] that data parallel of SAM model environment, he puts forward the variant of several quad-tree and how to use them to construct polygons. And shows how to carry out a certain number of operations in data parallel quad-tree. In general scanning model environment, Lindenbaum *et al*. (1995) and Hoel and Samet (2003) proposed data parallelism of polygon construction using the bucket PMR quad-tree, R Tree and R + tree.

This paper contributes as follows: We uniquely determine each polygon by the topological location relationship of elements in the plane graph. Edges are sorting by topological rule that is constructed based on the topological location relationship. Then find edges of result polygons in the collection of sorted candidate edge by a series of search operation. And form the results plane graph, finally form the result polygons. The algorithm can segmented search by a data parallel spatial data structure. It reduces the number of global search and improves the efficiency.

This paper is structured as follows: first, we describe the parallel computing model and intersection calculation for line segments in Section 2. Second, we describe how to construct polygons in parallel in Section 3. Third, we make experiments to prove whether the algorithm is feasible in Section 4. Fourth, we discuss the advantage and improvement of the algorithm in Section 5. Fifth, we conclude and offer research direction in Section 6.

2. Parallel Computing Model and Intersection Calculation for Line Segments

For the set of basic operations that can be operated in any long variable (one-dimensional array) data. Scanning model used in this paper [3] is defined as the general model of parallel computing. Three basic operations (search, vector product and arrange) are mainly used to generate a result vector of equal length. Search operation [12] requires a combination operator $\oplus$, an input vector $[a_0, a_1, \ldots, a_{n-1}]$, and returns a result vector $[a_0, (a_0 \oplus a_1), \ldots, (a_0 \oplus a_1 \oplus \ldots \oplus a_{n-1})]$.

A polygon is composed of at least three points. And each point is adjacent the two edges. The generation process is not by a simple human visual sense, but needs further polygonization on the map through good logic.

As shown in Figure 1, two polygons A (A1, A2,..., A5) and B (B1, B2,..., B4) have been given in the plane, their vertex arranged clockwise, plane graph P1 consists of polygon A and plane graph P2 consists of polygon B, P1 is composed of the edge of the boundary polygon A (A1,A2,...,A5), P2 is composed of the edge of the boundary polygon B (B1,B2,...,B4). Determining the intersection of polygons A and B, remember to be $A \cap B$. $A \cap B = (q|q \in A \&\& q \in B)$, the boundary of $A \cap B$ is cross assembly by the A boundary and B boundary, Intersection of the A edge and B edge are the staggered between the A edge and B edge in $A \cap B$ boundary sequences. Check all edges of B for each edge of A, whether they intersect, If there is a point of intersection then record, the steps to find the intersection are shown as follows:

a.) Selected line segment cannot intersect by the x value of the vertex, calculating the relationship between the position of the polygon A side $C_i = A_iA_{i+1}$ and the polygon B side $D_i = B_iB_{i+1}$. If C_i is on the left of D_i, calculating the position relation of C_{i+1} on D_i; If C_i is on the right of D_i, calculating the position relation of C_{i+1} on D_{i+1}.

b.) Determine C_i and D_i whether there are identical vertices, if true, record line segments C_i and D_i.

c.) Selected line segment cannot intersect by the y value of the vertex second time. If the minimum y_{pmin} value of C_i edge is greater than the maximum y_{qmax} value of D_i edge, the maximum y_{pmax} value of C_i edge is greater than the minimum y_{qmin} value of D_i edge, then the two sides do not intersect.

d.) For possible intersecting line segments from a) and b), calculate the intersection according to the different position of line segments. The method of comparing the intersection of different situations is as follows: calculating the relationship between the point and relative line segment, then deciding intersection based on the existing intersection relationship matrix.

```
Calculation algorithm for the intersection of Ci and Di
Begin
Calculate the relationship of point Ai and Ai+1 to Di
Calculate the relationship of point Bi and Bi+1 to Ci
If point Ai and Ai+1 is not on Di edge, and distributed on both edges of the Di
Begin
If point Bi and Bi+1 is not on Ci edge, and distributed on both edges of the Ci
Calculate and record the intersection
Else if Bi is on Ci edge and Bi+1 inside of the Ci edge
Intersection=Bi
Else if Bi+1 is on Ci edge and Bi+1 inside of the Ci edge
Intersection=Bi+1
End
Else if Ai inside or outside of the Di edge and Ai+1 is on Di edge
Intersection=Ai+1
Else if Ai+1 inside of the Di edge and Ai is on Di edge
Intersection=Ai
```

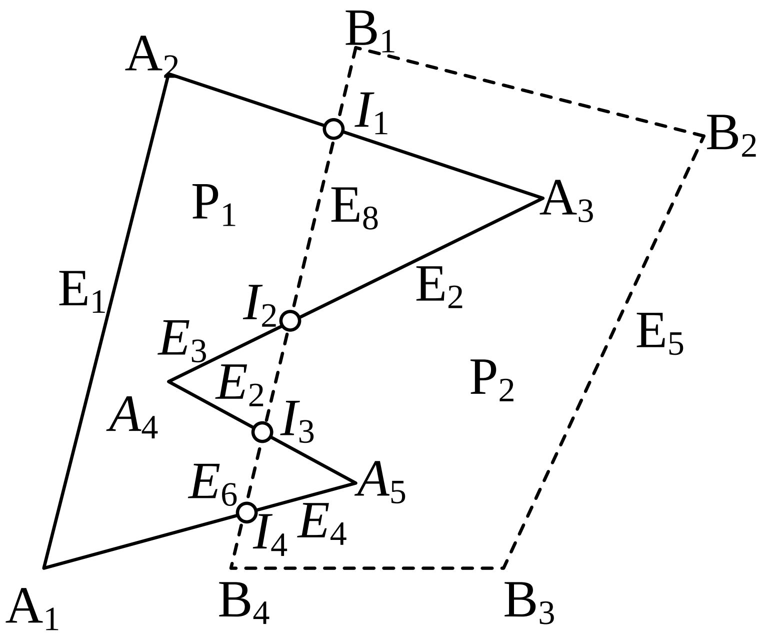


Figure 1. Intersections of Arbitrary Polygons

3. Parallel Construction

Before the polygon construction, the input line is generated a graph model. Follow these steps to add the edges: add the nodes 0 and 1 to the graph, Then add l_0 edge to graph, At the same time, add the directed edges v_0, v_1, the direction of v_0 is the same as the original edge, the direction of v_1 is the contrary as it, so called l_0 as the father edge, v_0 and v_1 are sub edge, and v_0 and v_1 are brother edge each either and their direction are contrary, add the first edge finished. Add the remaining edges accordingly, as shown in Figure 2.

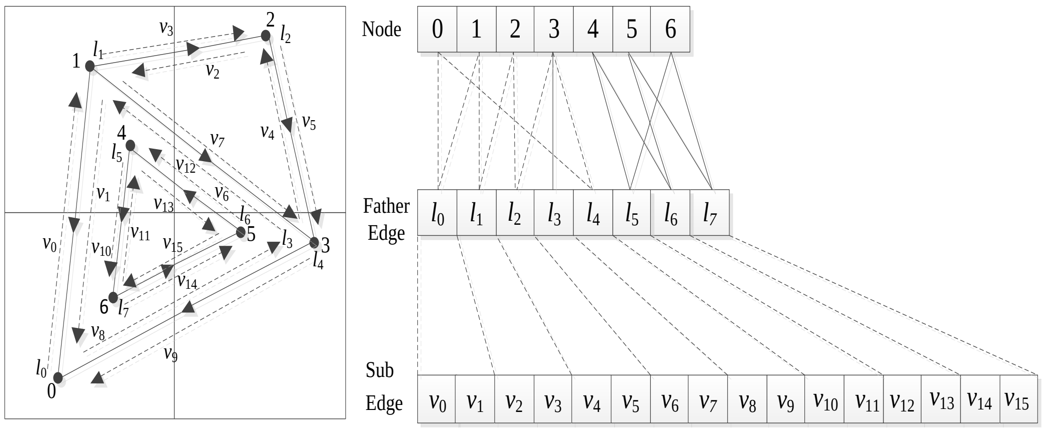

Node
0
1
2
3
4
5
6
Father Edge
l_0 l_1 l_2 l_3 l_4 l_5 l_6 l_7
Sub Edge
v_0 v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12} v_{13} v_{14} v_{15}

Figure 2. Generate Figure

3.1. Record Relationships between Edges

The point on a polygon is stored according to a certain order, for the edge of a polygon also requires a certain order recorded. Sorting the out edge of the same node according to the size of the direction angle of directed edges. Take the start node of a directed edge as the origin, the horizontal direction is X axis, the vertical direction is Y axis (with the same plane rectangular coordinate system), take the positive x axis as initial direction, Counterclockwise rotation angle to the segment composed of the starting node and the second node is the directed edge direction angle. For the out edges' direction angle of node 1 in Figure 2, the result is: v_3-v_7-v_1 after arrange their out edge, the amplified results are shown in Figure 3.

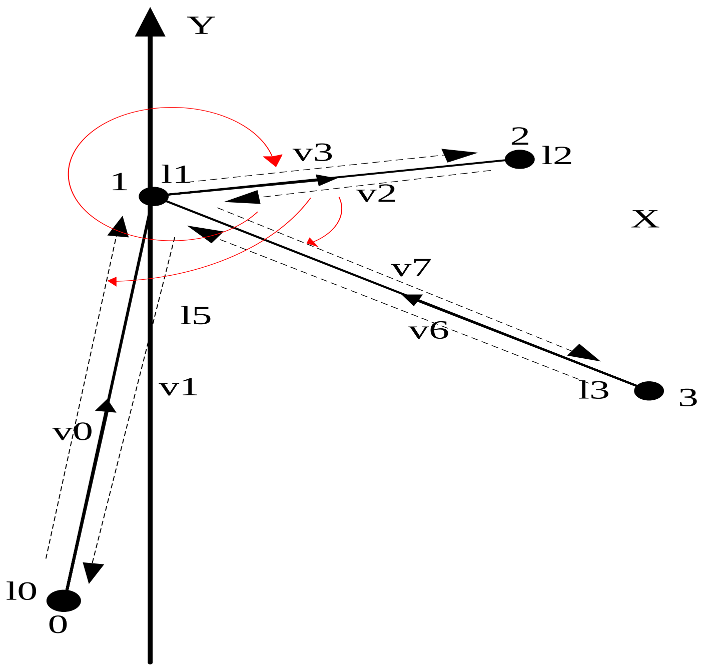


Figure 3. Out Edges' Direction Angle of Node 1

Taking packet sorting algorithm when out edges in a graph that has m nodes to be sorted out. To ensure that nodes in the same group are spatially adjacent to each other, first carry out the global space partition of the nodes in Figure 4, then sort the nodes according to the spatial location, and then group the sorted nodes, in general to ensure a group of nodes is the spatial adjacency. Supposing there are n processors (n≥m), according to the number of processors after grouping nodes, the number of nodes z= (z_1,z_2,…z_n), considering the load balance, the number of nodes in each group does not necessarily equal, but the total amount of data balance and the nodes in the same group close to each other in space.

To sort out the edge of each node according to the direction angle, for each node, the relationship between the edges mainly are: for node i, take the brother edge v_q of the out edge vp according to the out edges order after sorting, set the next edge of vq for the next out edge vm sequencing around the node i, take the brother side vn of vm, set the storage relationship between edges accordingly, until back edge vp. Therefore, the storage relationship of directed edges in Figure 4 is shown in Table 1.

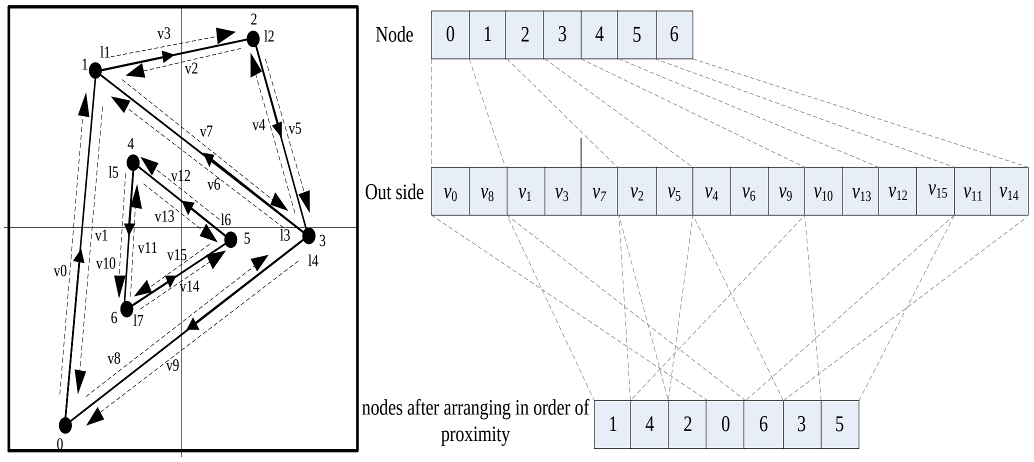


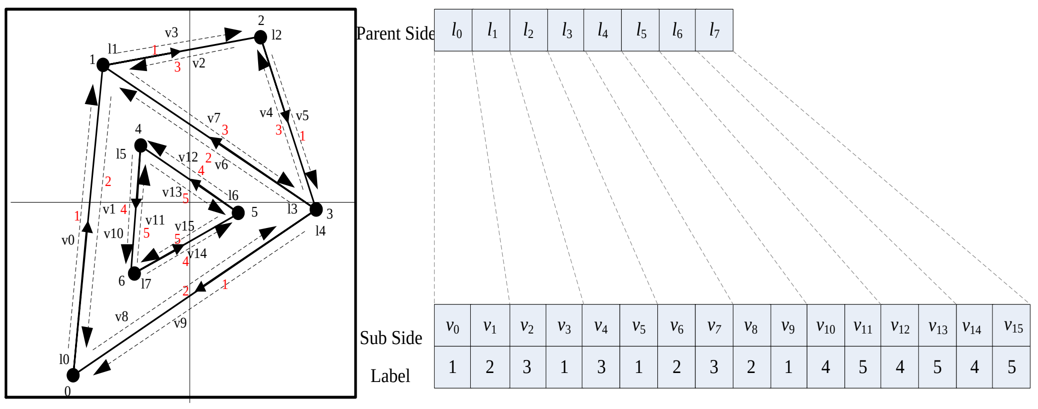
Figure 4. Nodes after Arranging In Order of Proximity

Table 1. Topology of Each Sub Edges

Number	Storage relationship of edges
1	v0→v7→v9→v0
2	v1→v8→v4→v2→v1
3	v2→v7→v4→v2
4	v7→v9→v0→v7
5	v1→v8→v6→v1
6	v10→v14→v12→v10
7	v11→v13→v15→v11

3.2. The Calculation of the Result Polygons

Mark label, sub edges will divide into several groups of input vector corresponding according to the storage relationship between edges, sub edges in each vector are unique, between groups does not exist repeat, each group vector is marked by the vector group number. Vector groups are assigned to each processor according to the number of groups and the processors. Finally, according to the group number, Vector product operation is performed in parallel for sub edges in each group and mark. From v_0, $v_0 \to v_3 \to v_5 \to v_9$, the label of these directed edges is marked as 1;from v_1, $v_1 \to v_8 \to v_6$, the label of these directed edges is marked as 2, for v_2,$v_2 \to v_7 \to v_4$ is marked as 3; $v_{10} \to v_{14} \to v_{12}$ is marked as 4. The marked results are shown in Figure5.



Parent Side	l_0	l_1	l_2	l_3	l_4	l_5	l_6	l_7

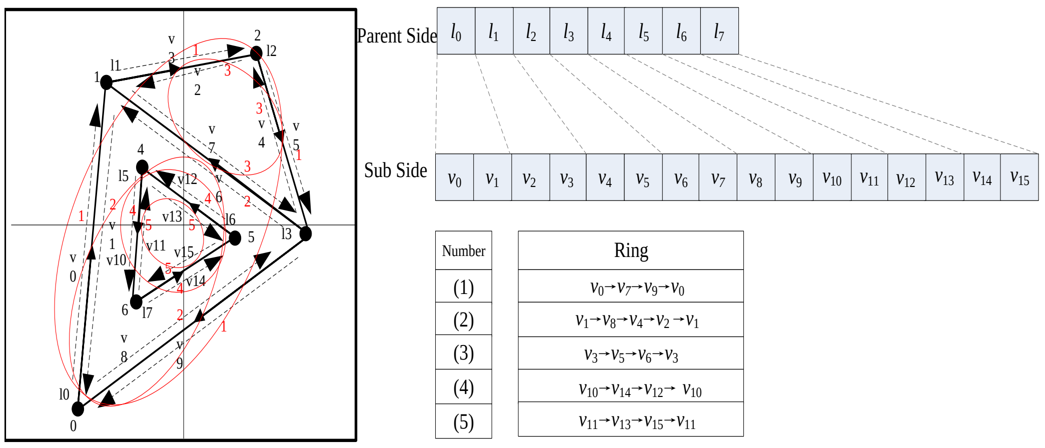
Sub Side	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}
Label	1	2	3	1	3	1	2	3	2	1	4	5	4	5	4	5

Figure 5. Sign Label

The purpose of marking each sub edge is to get special mark line, special mark line refers to sub edge having the same label and belongs to the same parent side, and these special mark lines will be deleted. There are three main steps to delete mark line, which are basic operations in parallel scanning model: a) exclusive scanning forward; b) logical reduction; c) sort

3.3. Generate a Valid Ring

The ring is generated according to the storage relationship between edges, closed ring is a valid ring, determine whether the ring is closed is mainly to determine whether the head edge and the tail edge of the ring are same, the same head and tail edge of ring is closed ring. There are at least two rings in a polygon, so the number of required to judging the ring is at least two times the number of polygon, this process takes a long time to judge. All rings will be assigned to each processor according to the number of available processors in parallel construction, each processor handles a group of adjacent rings, and parallel operation causes the judgment time greatly reduced. Finally, according to the handling results of each group, retain valid ring, delete the invalid ring. Figure2 total generates six rings, the rings are valid after executing judgment operation, and the valid ring generated is as shown in Figure 6.



Parent Side	l_0	l_1	l_2	l_3	l_4	l_5	l_6	l_7

Sub Side	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}

Number	Ring
(1)	$v_0 \rightarrow v_7 \rightarrow v_9 \rightarrow v_0$
(2)	$v_1 \rightarrow v_8 \rightarrow v_4 \rightarrow v_2 \rightarrow v_1$
(3)	$v_3 \rightarrow v_5 \rightarrow v_6 \rightarrow v_3$
(4)	$v_{10} \rightarrow v_{14} \rightarrow v_{12} \rightarrow v_{10}$
(5)	$v_{11} \rightarrow v_{13} \rightarrow v_{15} \rightarrow v_{11}$

Figure 6. Generate a Valid Ring

The final step to construct polygon is to judge the valid ring is shell or whole according to the direction of the ring, when the direction is counterclockwise, the ring is hole, when the direction is counterclockwise, and the ring is shell. This judgment process needs to spend a lot of time like the judgment of the valid ring, so use the same method as the above judgment, ring will be grouped according to the number of processors, then carry out concurrent judgment on all ring groups.

Table 2. Holes and Shells

Shells	Holes
v0→v7→v9→v0	v1→v8→v4→v2 →v1
v3→v5→v6→v3	v11→v13→v15→v11
v10→v14→v12→ v10	

Table 3. The Experimental Results under Different Number of Cores

The Number Of Cores	Time/s	The Number Of Lines	Number Into Area
1	30	93664	34123
2	22	93664	34123
4	16	93664	34123
6	18	93664	34123
8	18	93664	34123

4. Experiments

Under the premise of maintaining the correctness, the final purpose of parallel polygon construction is to improve the operational speed significantly when constructing polygon operation to a certain amount of space data. Table 3 describes operation time of parallel polygon construction on the scale for linear data of 93664 objects, which is under the different number of cores.

Through improving a part of the calculation, the resulting performance improvement can be reflected by the Amdahl law quantitatively. The Amdahl law defines the speedup achieved by using a kind of optimization or parallel program.

$$\text{Speedup} = \frac{\text{The running time of the whole task before parallel}}{\text{The running time of the whole task after parallel}}$$

Through speedup formula, the experimental results can be got, speedup line chart is shown in Figure7.

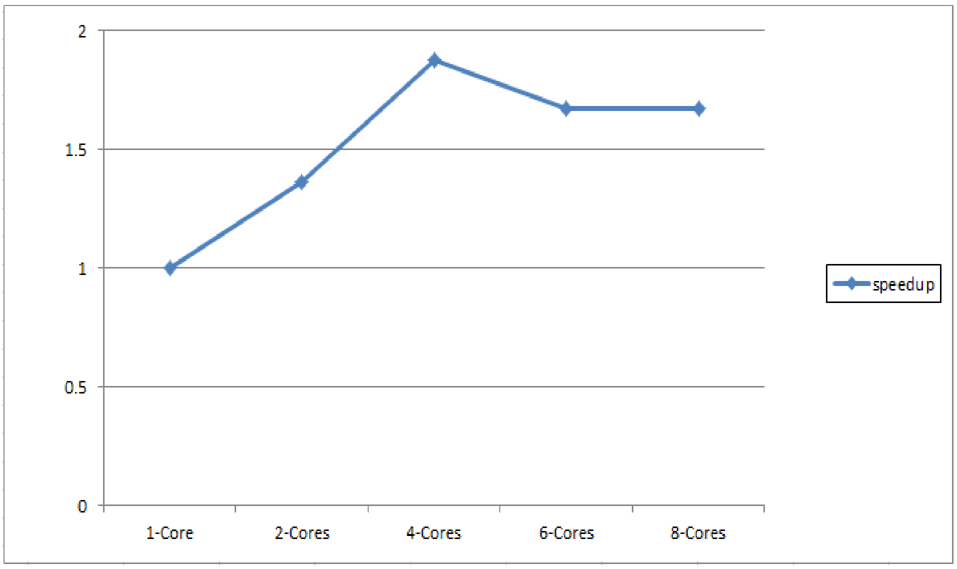


Figure 7. Speedup Line Chart

The line chart shows that the speedup of quad-core is the highest, reached 1.875, while the speedup of six-core and eight-core are reduced to 1.66, this shows that the more cores used in parallel, the more overhead. But on the whole, the parallel speedup is quite good, achieve the purpose of improving the efficiency. Therefore, this new polygon construction approach in the parallel environment is an effective way to improve efficiency.

5. Discussion

The polygon construction is one of the key issues for constructing the topological relationship of spatial data. Scholars have optimized and improved the automatization, speed and complexity of the algorithm. This paper proposes a new algorithm for polygon construction in the parallel environment. So the algorithm in this paper has more advantages than general algorithms as follows: (1) it is automated and no manual interention. (2) By the topological location relationship of elements in the plane graph, we can uniquely determine each polygon. The spare situation when searching polygons in general algorithms is avoided. Thus polygons are topology constructed faster. (3) The key of the algorithm is paralleling the more time consuming operation (search, sort. *etc*.). And generate directed rings faster. Then construct polygon for a set of line segments. So it reduces the number of global search and improves the running speed. It also improves efficiency of polygon construction.

The algorithm is proved under the background of multicore computers. The experimental results have proved that construct polygon in parallel environment is feasible. The experiments prove a massive reduction in the running time of the same size data in the multicore environment. However, the more cores are used, the more overhead. The direction of improvement of the algorithm is how to improve the speedup and scalability at the same time. So it can be applied in practical application.

6. Conclusion

This paper put forwards an automatic parallel polygonization algorithm based on the graph model, improve the basic operation efficiency of spatial analysis. The purpose of this paper is to parallel the basic operation such as the inquiry, arrangement and other time-consuming operation in the calculation to improve efficiency. On the basis, considering the proximity of spatial data distribution and load balance between parallel nodes. In order to prove the feasibility and effectiveness of the algorithm, this paper has conducted a group of tests. Automatic parallel polygonization algorithm based on the graph model has achieved the desired effect. The experimental results prove that the algorithm is feasible. The main direction of future research is to enhance the scalability and improve speedup, and realize some other operation, such as space intersection.

Acknowledgments

This work was supported by the National Natural Science Foundation of China(41401443) and Open Research Fund of State Key Laboratory of Geography Information Engineering (SKLGIE2013Z41).We thank anonymous reviewers for their constructive comments on the earlier version of this paper.

References

[1] T. Bestul, "Parallel Paradigms and Practices for Spatial Data", College Park: University of Maryland, **(1992)**.
[2] G. E. Blelloch, "Scans as Primitive Parallel Operations", IEEE Trans on Computers, vol. 38, no. 11, **(1989)**, pp. 1526-1538.
[3] G. E. Blelloch and J. J. Little, "Parallel Solutions to Geometric Problems on the Scan Model of Computation", Cambridge, MA: MIT, **(1988)**.
[4] Z. L. Chen, X. C. Wu and L. Wu, "Polygon Overlay Analysis Algorithm Based on Monotone Chain and STR Tree in the Simple Feature Mode", Geodaetica Et Cartographica Sinica, vol. 39, no. 1, **(2010)**, pp. 102-108.
[5] R. H. Guting and D. Papadias, "Advances *in* Spatial Databases", Proceeding of the 6th International Symposium on SSD, **(1999)**.
[6] E. G. Hoel and H. Samet, "Data-parallel Polygonization. Parallel Computing", vol. 29, no. 10, **(2003)**, pp. 1381-1401.
[7] D. J. Li, B. Liu and B. G. Zhao, "An Improvement Algorithm of Topological Polygon Auto-Construction, Computer Engineering and Applications", vol. 41, no. 16, **(2005)**, pp. 80-82.
[8] M. Lindenbaum, H. Samet and G. R. Hjaltason, "A Probabilistic Analysis of Trie-based sorting of Large Collections of Line Segments. College Park", University of Maryland, **(1995)**.
[9] F. Luo, T. H. Ai and H. Wang, "An Algorithm to Create Topological Relationship Based on Closed Coordinate Chain Data", Editorial Board of Geomatics and Information Science of Wuhan University, vol. 29, no. 6, **(2004)**, pp. 558-561.
[10] T. Michikawa and H. Suzuki, "Polygonization of Volumetric Skeletons with Junctions", Computer-Aided Design, vol. 45, no. 4, **(2013)**, pp. 822-828.
[11] M. Scholl, "Advances in Spatial Databases", Proceedings of the 5th International Symposium on SSD, **(1997)**.
[12] J. T. Schwartz, "ACM Trans on Programming Languages and Systems", Ultracomputers, vol. 2, no. 4, **(1980)**, pp. 484-521.
[13] J. C. Wang, "The Raster Algorithm on Creation of Topological Relationship of Polygons", Geodaetica Et Cartographica Sinica, vol. 31, no. 3, **(2002)**, pp. 249-254.
[14] H. W. Yan and Q. G. Cheng, "A Fast Algorithm of Topological Polygon Auto-Construction Based on Azimuth Calculation", China Image and Graphics, vol. 5, no. 7, **(2000)**, pp. 563-567.