


Article

Multicore Parallelized Spatial Overlay Analysis Algorithm Using Vector Polygon Shape Complexity Index Optimization

Junfu Fan ^{1,2} , Jiwei Zuo ¹, Guangwei Sun ^{1,*}, Zongwen Shi ¹, Yu Gao ¹ and Yi Zhang ^{3,4}

¹ School of Civil Engineering and Geomatics, Shandong University of Technology, Zibo 255000, China; fanjf@sdut.edu.cn (J.F.); 21407010773@stumail.sdut.edu.cn (J.Z.); 21407010766@stumail.sdut.edu.cn (Z.S.); 21507020775@stumail.sdut.edu.cn (Y.G.)

² State Key Laboratory of Resources and Environmental Information System, Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences, Beijing 100101, China

³ College of Land and Resources and Surveying & Mapping Engineering, Shandong Agriculture and Engineering University, Jinan 250100, China; cloud.zhangyi@foxmail.com

⁴ School of Geosciences and Info-Physics, Central South University, Changsha 410083, China

* Correspondence: sgw_sdut@163.com; Tel.: +86-13-32520-6076

Abstract: As core algorithms of geographic computing, overlay analysis algorithms typically have computation-intensive and data-intensive characteristics. It is highly important to optimize overlay analysis algorithms by parallelizing the vector polygons after reasonable data division. To address the problem of unbalanced data partitioning in the task decomposition process for parallel polygon overlay analysis and calculation, this paper presents a data partitioning method based on shape complexity index optimization, which achieves data equalization among multicore parallel computing tasks. Taking the intersection operator and difference operator of the Vatti algorithm as examples, six polygon shape indexes are selected to construct the shape complexity model, and the vector data are divided in accordance with the calculated shape complexity results. Finally, multicore parallelism is achieved based on OpenMP. The experimental results show that when a data set with a large amount of data is used, the effect of the multicore parallel execution of the Vatti algorithm's intersection operator and difference operator based on shape complexity division is clearly improved. With 16 threads, compared with the serial algorithm, speedups of 29 times and 32 times can be obtained. Compared with the traditional multicore parallel algorithm based on polygon number division, the speed can be improved by 33% and 29%, and the load balancing index is reduced. For a data set with a small amount of data, the acceleration effect of this method is similar to that of traditional methods involving multicore parallelism.

Keywords: overlay analysis; shape complexity; data partitioning; parallel computing; acceleration ratio; load balancing; OpenMP



Citation: Fan, J.; Zuo, J.; Sun, G.; Shi, Z.; Gao, Y.; Zhang, Y. Multicore Parallelized Spatial Overlay Analysis Algorithm Using Vector Polygon Shape Complexity Index Optimization. *Appl. Sci.* **2024**, *14*, 2006. <https://doi.org/10.3390/app14052006>

Academic Editors: Ferdinando Di Martino and Barbara Cardone

Received: 29 January 2024

Revised: 24 February 2024

Accepted: 27 February 2024

Published: 28 February 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The overlay analysis of vector polygons is one of the basic functions of geographic information system (GIS) spatial analysis [1], and it is an important and difficult problem in the field of geographic data processing [2,3]. Vector polygon overlay analysis is not only complicated in algorithm implementation but is also becoming increasingly computationally intensive and data-intensive with the rapid growth of spatial data volumes [4–6]. The computing performance of serial algorithms in dealing with large-scale spatial data superposition is showing increasing difficulty in meeting application requirements [7]. The development of computer hardware technology and programming models has given parallel computing an important role in solving intensive problems in the field of geoscience computing. The ability of a CPU to directly and quickly access internal storage also allows data sharing between tasks to be fast and unified. The OpenMP programming

model in a multicore environment is an effective way to perform parallel polygon overlay analysis [8–10].

In parallel spatial analysis, the primary task of the parallel processing of spatial data is to reasonably divide complex data and store them evenly and in a dispersed manner among multiple computing nodes [11]. Ye et al., divided vector data in accordance with the storage order of the polygon IDs, with the same number of polygons in each group [12]. Mineter, based on the spatial range division method of regular strips, extended the grid division method for raster data division to the division of vector data and divided the data in accordance with the spatial position of the data set [13]. Wang et al. proposed a quadtree partitioning method based on the spatial locations of data sets [14]. Lee et al., proposed a heuristic partitioning method using equal areas as the criterion for partitioning [15]. Zhou et al., proposed a data partitioning method based on the Hilbert curve [16]. Yang et al. proposed a data partitioning method that amounts to the topological relationships of polygons and achieved good parallel efficiency [17]. From the perspective of multicore data parallelism, Fan et al. analyzed the similarities and differences of the parallel implementation methods of eight polygon overlay analysis tools and proposed an improved grouping association minimization method to achieve data partitioning. Based on a load-balancing calculation strategy with the number of vertices as an indicator and a variety of parallel optimization methods and strategies, a parallel polygon overlay analysis toolset, including eight operations, was implemented [10]. Wu et al. proposed a parallel filling and digging algorithm for large-scale DEM data based on strip division [18,19]. These scholars' data partitioning methods have improved the efficiency of parallel spatial data processing, but these partitioning methods cannot fully consider the impacts of polygon shape, size, and structure, which can easily lead to an imbalance in the amount of calculation needed among different groups.

The data structures of the geographic polygons used in overlay analysis can greatly differ. These polygons include not only simple convex polygons but also complex concave polygons, polygons with hole islands, and self-intersecting polygons [20]. To quantitatively measure the complexity of polygons, the concept of shape complexity has been proposed [21]. Attneave first proposed constructing a shape complexity model based on geometric features such as the number of vertices, symmetry, turning number, and angular variability, but this model is effective only for concave polygons [22]. Chen et al. used global distance entropy, local angle entropy, and polygon randomness to measure the shape complexity of two-dimensional graphs [23]. Duan et al. used a combination of the area, perimeter ratio, and number of blocks of a graph to represent shape complexity [24]. Brinkhoff et al. selected structural parameters, metric parameters, and statistical parameters to construct a polygon complexity model [25]. Matsumoto et al. used the absolute curvature to quantify the surface shape complexity of objects [26]. Guo et al. used the number of polygon features and the number of vertices of each feature to quantify the complexity of objects and, based on this, achieved reasonable data partitioning in parallel computing [27]. Li et al. defined a complexity calculation method for line elements and area elements based on two indicators, namely, an area coefficient and an angle coefficient, and quantified the geometric information of sensitive elements in digital maps [28]. However, although this method partially solves the problem of measurement complexity in geographic calculations, it does not reflect the relationship between shape complexity and the computational efficiency of an overlay analysis algorithm. Zhang et al. selected 27 shape variables and used a stepwise multiple linear regression method to establish a shape complexity model [29], which effectively explained the relationship between complexity and algorithm efficiency; however, this shape complexity model was not applied to vector data partitioning.

With the continuous development of the field of spatial overlay analysis, many polygon clipping algorithms with different characteristics have emerged. Common polygon clipping algorithms include the Wang algorithm [30], the Liang–Barsky algorithm [31], the Sutherland–Hodgeman algorithm [32], the Weiler–Atherton algorithm [33], the Vatti algorithm [34], the Greiner–Hormann algorithm [35], the Liu algorithm [36], and the Mar-

tinez algorithm [37]. All of these algorithms have their own advantages and disadvantages. The Wang algorithm and the Liang–Barsky algorithm apply only to arc polygons. The Sutherland–Hodgman algorithm can be used to clip only arbitrary polygons and convex polygons. The Weiler–Atherton algorithm can be used to perform the overlay analysis of concave polygons and convex polygons, but there is a possibility of failure when dealing with the superposition of overlapping edges. Compared with the Greiner–Hormann algorithm, Liu’s algorithm offers no substantial improvement, and the algorithm proposed by Martinez et al. has not been widely used [19]. The Greiner–Hormann algorithm and the Vatti algorithm are generally recognized as effective algorithms that can address arbitrary polygon clipping problems in a limited time and obtain correct results. However, according to the experimental results of Martinez et al., the computational efficiency of the Greiner–Hormann algorithm may be lower than that of the Vatti algorithm [37]. Therefore, the Vatti algorithm is selected as the clipping algorithm in this paper.

In summary, previous scholars have carried out a lot of research on vector data partitioning and polygon shape complexity measurement and have achieved certain results, but there are some shortcomings. The current data partitioning method cannot fully consider factors such as polygon spatial characteristics and shape characteristics, resulting in unbalanced data after grouping; the measurement of shape complexity cannot fully reflect the relationship with the performance of the overlay analysis algorithm. In view of the above considerations, this paper combines shape complexity measurement with data partitioning to establish a shape complexity model that can effectively explain algorithm performance and applies it to the partitioning of vector data to make the grouped data complexity similar and achieve data balance. Using two kinds of vector data with different amounts of data, the intersection operator and difference operator of the Vatti algorithm are taken as examples for multicore parallel comparative analysis. Specifically, a variety of polygon shape indicators are selected, a stepwise multiple linear regression method is used to establish a shape complexity model, the shape complexity value of each polygon is calculated in accordance with this shape complexity model, and the vector data are then divided based on the calculated shape complexity results. Finally, multicore parallelization is performed on the divided data based on OpenMP. Moreover, the effectiveness of the data partitioning method proposed in this paper for parallel overlay analysis is verified in comparison with the traditional data partitioning method based on the number of polygons. The acceleration effect and load balancing index are tested and evaluated in environments with different numbers of parallel threads, and the most suitable number of parallel threads is determined to improve the computational efficiency of the algorithm as much as possible.

2. Data

The parallel processing algorithms discussed in this article were developed in the C++ language, and the configuration of the experimental environment is shown in Table 1.

Table 1. Experimental environment configuration.

CPU	GPU	Memory/GB	External Memory/TB	Compiler
AMD Ryzen 9 5950X 3.4 GHz 16 core	NVIDIA GeForce RTX 3090	128	4	Visual Studio 2019

The experimental data for the target polygons were obtained from land use patch data for China in OpenStreetMap and from the 2009 land use data for several counties in Ningxia. The clipping polygon uses rectangular data with holes. The amount of land use patch data for China is large: with 509,219 polygons, the number of vertices can reach 12,914,282, and the overall complexity is high, as shown in Figure 1a. The amount of land use data for several counties of Ningxia in 2009 is small, with 9129 polygons and 856,238 vertices, as shown in Figure 1b. Figure 1c shows 52,100 rectangular holes used for the overlay analysis

of the land use patches for China. Figure 1d shows 196 rectangular holes used for the overlay analysis with the land use data for several counties in Ningxia in 2009. In both cases, the target polygon data include simple polygons, concave polygons, hole-containing island polygons, and self-intersecting polygons, representative of the general characteristics and application value of planar geographic data in overlay analysis. The basic information of the experimental data is shown in Table 2.

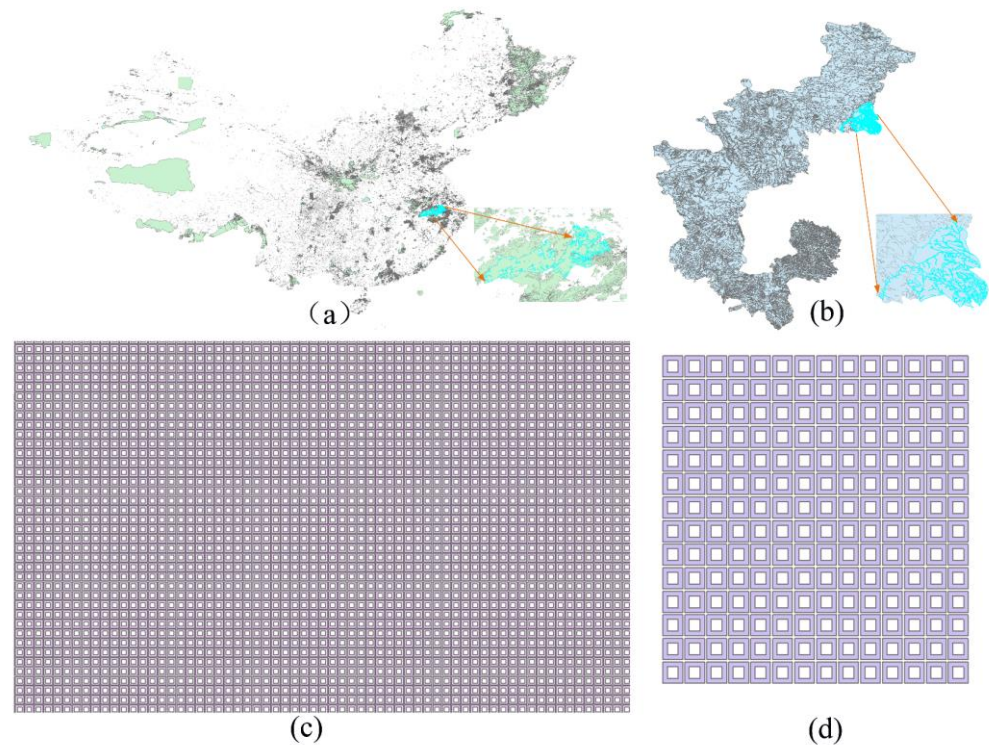


Figure 1. Diagrams of the experimental data. (a) Land use patch data from China. (b) 2009 land use data for several counties in Ningxia. (c) 52,100 rectangular data with holes. (d) 196 rectangular data with holes.

Table 2. Details of the experimental data.

Experimental Data	Number of Polygons	Number of Vertices	Area/km ²
Land use patch data from China	509,219	12,914,282	96,390.5
2009 land use data for several counties in Ningxia	9129	856,238	8611.9

3. Methods

3.1. Polygon Clipping Algorithm

The main process of clipping polygons with the Vatti algorithm is shown in Figure 2. First, the vector polygons are read, and disjoint polygons are filtered out based on the minimum bounding rectangle. Second, each polygon is traversed from the minimum point at the bottom to the maximum point at the top by a scanning beam covering the area between two adjacent scanning lines to extract the location information of each node of the polygon, which is stored in the Edge Table, while the local minimum point is uniformly stored in the Local Minima Table, where all local minimum points are connected together in the form of a one-way linked list. Finally, the resulting polygon is constructed by calculating the intersections with the inner edges of each scanning beam and inserting the intersection points.

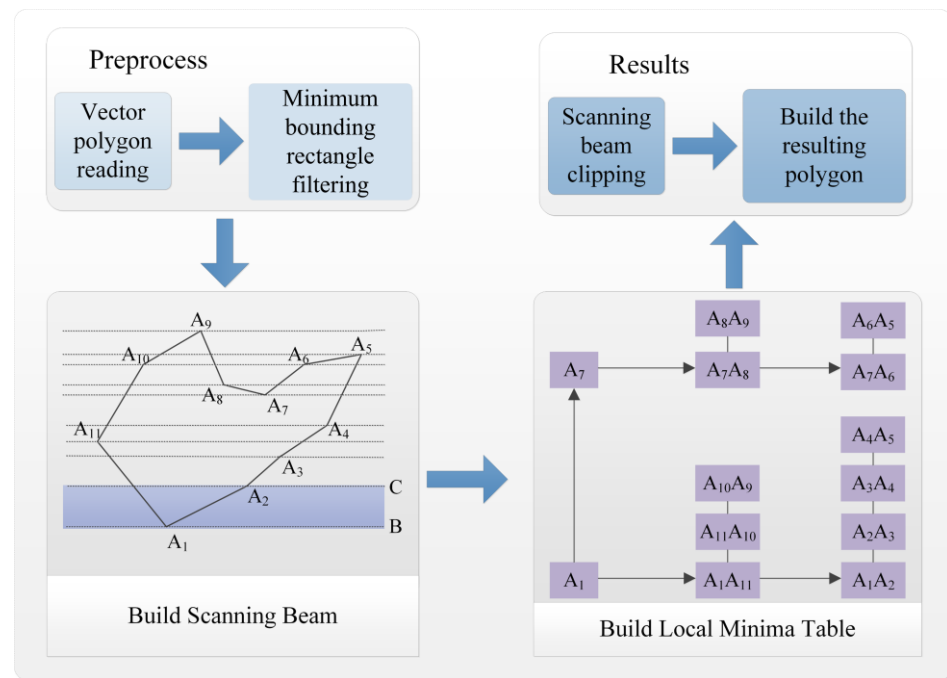


Figure 2. Flow chart of the Vatti algorithm.

3.2. Construction of the Shape Complexity Model

Zhang Peng et al. initially selected 27 shape variables, and 9 indexes were retained by a stepwise multiple linear regression method for use in constructing the shape complexity model. In this way, the relationship between the shape complexity of the polygons and the computational performance of a spatial overlay analysis algorithm was established. The model shows that the number of polygon vertices is the most important factor affecting the overlay analysis, with its coefficient in the complexity model reaching 0.97 [29]. In multiple regression analysis, as the number of independent variables increases, the model performance may improve, but this change is accompanied by greater computational overhead and obvious variable redundancy. Therefore, while ensuring the accuracy of the model, minimizing the redundancy of the variables can effectively reduce the time needed to calculate the complexity index. This paper is based on reference [29], retains the index of the number of vertices with the greatest impact, and selects 5 indexes that take less time to calculate. The final indexes are as follows: Number of vertices (NOV), Amplitude of the vibration (Ampl), Number of holes (NOH), Average nearest neighbor (ANN), Concavity (Conv), and Equivalent rectangular index (ER). Detailed descriptions of these shape indexes are shown in Table 3.

In this paper, 100,000 polygons are randomly selected from the land use patch data for China and the 2009 land use data for several counties in Ningxia for use as target polygons, and a typical complex polygon is selected as a clipping polygon. For each target polygon, an overlay analysis operation is performed with the selected clipping polygon, with the above six indicators as independent variables and the running time of the polygon overlay analysis algorithm as the dependent variable. The stepwise multiple linear regression method in the IBM SPSS Statistics 27.0.1 software is used to construct the shape complexity models for the intersection operator and difference operator of the Vatti algorithm. The model equations are (1) and (2):

$$C_1 = 0.971x_1 + 0.059x_4 + 0.018x_5 + 0.034x_6 \tag{1}$$

$$C_2 = 0.97x_1 - 0.004x_2 + 0.037x_3 + 0.06x_4 + 0.024x_5 + 0.037x_6 \tag{2}$$

where C_1 is the shape complexity of the intersection operator of the Vatti algorithm, C_2 is the shape complexity of the difference operator, x_1 is the Number of vertices, x_2 is the Amplitude of the vibration, x_3 is the Number of holes, x_4 is the Average nearest neighbor, x_5 is the Concavity, and x_6 is the Equivalent rectangular index.

Table 3. Polygon shape indexes. (*Ach*: area of the convex hull of the polygon; *Apol*: area of the polygon; *Ambr*: area of the minimum bounding rectangle of the polygon; *Ppol*: perimeter of the polygon; *Pch*: perimeter of the convex hull of the polygon; *Pmbr* perimeter of the minimum bounding rectangle of the polygon; *Pear* perimeter of equal-area rectangle of the polygon.)

Polygon Shape Index	Equation	Description
Number of vertices (NOV)	$NOV = V_{pol}$	The number of vector polygon vertices
Amplitude of the vibration (Ampl)	$Ampl = (P_{pol} - P_{ch}) / P_{pol}$	The perimeter difference between the vector polygon and its convex hull
Number of holes (NOH)	$NOH = H_{hole}$	The number of inner rings of the vector polygon
Average nearest neighbor (ANN)	$ANN = \frac{\sum_{i=1}^n d_i}{n} / \frac{0.5}{\sqrt{n} / A_{pol}}$	The degree of spatial clustering of the vector polygon vertices
Concavity (Conv)	$Conv = (Ach - Apol) / Ach$	The number of concave points in the vector polygon vertices
Equivalent rectangular index (ER)	$ER = P_{ear} / P_{pol} = \sqrt{\frac{A_{pol}}{A_{mbr}}} \times \frac{P_{mbr}}{P_{pol}}$	The difference between the perimeter of a vector polygon and its equal-area rectangle

Regression analysis is widely used in the fields of econometrics, psychology, and sociology. The coefficient of determination (R^2) is mainly used to evaluate the quality of a model. The closer R^2 is to 1, the better the model fitting effect and the higher the consistency. In this paper, shape complexity models are established based on the intersection operator and difference operator of the Vatti algorithm. The coefficients of determination of both shape complexity models based on the intersection operator and the difference operator of the Vatti algorithm are greater than 0.89, indicating that the shape complexity of the input polygon is highly correlated with the calculation time. Specifically, the shape characteristics of the input polygon can explain more than 89% of the calculation time of the algorithm. The model summary is shown in Table 4.

Table 4. Model summary.

	R	R^2	Adjusted R^2
Vatti Intersection	0.943	0.890	0.890
Vatti Difference	0.947	0.896	0.896

By means of the constructed shape complexity models, the shape complexity of each polygon subjected to clipping can be calculated. In Figure 3, the calculation times for 100,000 polygons are plotted versus their shape complexities, as evaluated based on the intersection operator and difference operator of the Vatti algorithm in the form of scatter plots. Linear fits to these data are also shown. According to Figure 3, 89% of the data points are concentrated near the regression line for both the intersection operator and difference operator of the Vatti algorithm, indicating that the fitting accuracy is high.

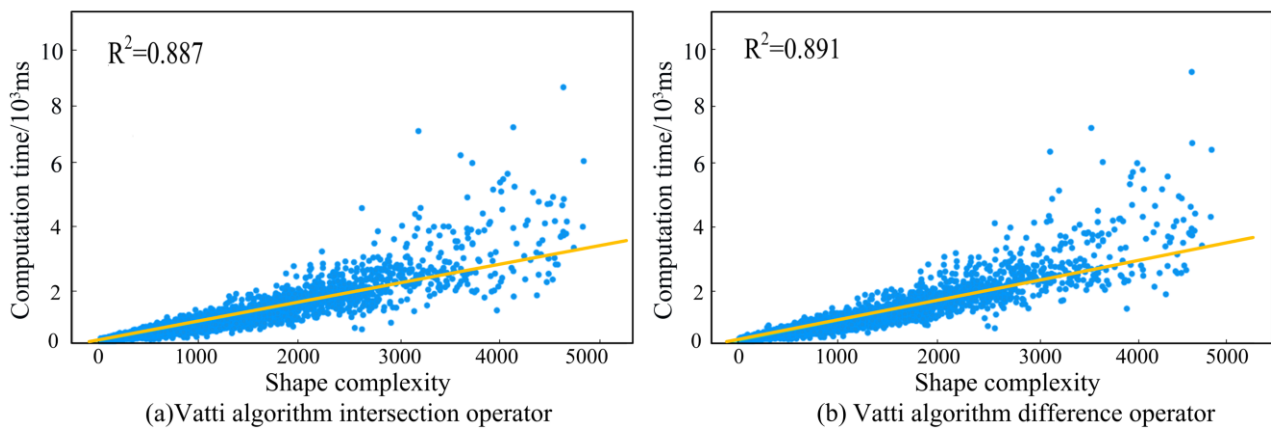


Figure 3. Scatter plots and linear fitting of calculation time versus shape complexity.

3.3. Data Partitioning and Parallelization

3.3.1. Polygon Number Division and Parallelization

The traditional polygon number division method is a direct and simple method of dividing data in accordance with the order of polygon storage. This method is easy to implement; it does not cause damage to the input data when dividing, and the number of polygons in each group after division is equal. Moreover, this method has clear termination conditions for the loop traversal clipping process of the target layer and can be easily parallelized based on OpenMP. Figure 4 shows a schematic diagram of the multicore parallel computing method based on the number of polygons. The main process is to sequentially extract each polygon element from the vector data, ensure that the number of polygon elements in each group of data is equal, distribute the divided data among different CPU threads, and perform the polygon overlay analysis operation in parallel. The complexity of each group of data grouped by this method varies greatly, and the balance is poor when multiple cores execute the polygon overlay analysis algorithm.

FID	Shape	Name	...
0			
1			
2			
3			
4			
5			
6			
7			
8			
...			
...			
...			

} CPU1

} CPU2

} CPU3

⋮

Figure 4. Polygon multicore parallel computing based on polygon number division.

3.3.2. Shape Complexity Partitioning and Parallelization

In this paper, a data partitioning method based on polygon shape complexity is designed and implemented. After partitioning, the complexity for each group of data is similar. The process of this method is as follows: ① The vector data are read, the polygons in the layer are traversed, the six indicators of NOV, Ampl, NOH, ANN, Conv, and ER of

each polygon are calculated according to the calculation formula, the shape complexity model obtained above is brought in to calculate the complexity value of each polygon in the vector data, and it is stored in the table. ② The sum of the complexity values of all polygons in the table is calculated. The sum value is denoted by M ; the number of groups, which is determined by the number of CPU threads, is denoted by n ; and the threshold on the complexity that can be assigned to each thread is denoted by L . For example, when the number of threads is selected to be 16, the data will be divided into 16 groups, and the complexity threshold of each group of data is 1/16 of the sum of the complexity values of the vector data. This complexity threshold is calculated as shown in Equation (3).

$$L = M/n \tag{3}$$

③ The vector data are grouped in accordance with the calculated complexity threshold to ensure that the shape complexity of each group of data does not exceed the threshold and that the complexities are roughly equal. ④ Each CPU thread reads the data of a completed group and calls the polygon overlay analysis algorithm for calculation. ⑤ The data for each group are output after overlay analysis, and the results for each group are combined. Figure 5 shows a schematic diagram of the proposed multicore parallel computing method based on shape complexity. This method can account for the influence of polygon shape and size on the computational efficiency of the algorithm and, thus, achieve a better load balance among CPU threads while avoiding complex element cutting and stitching operations.

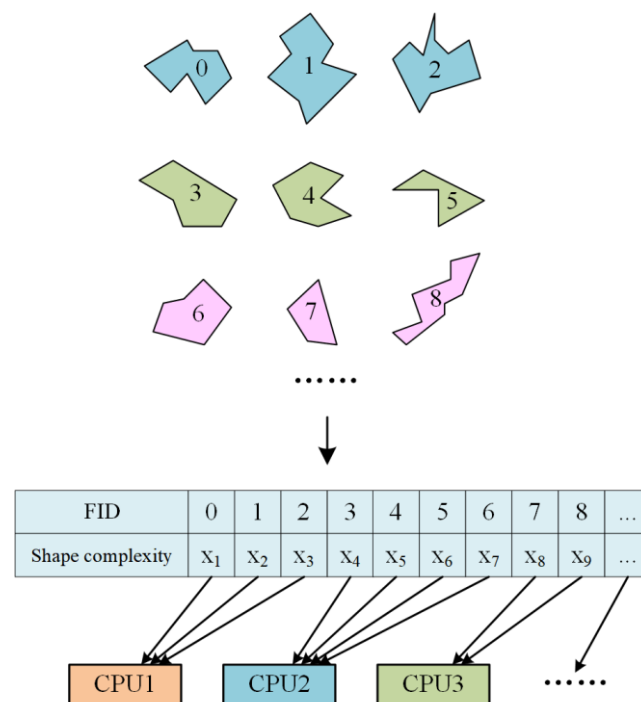


Figure 5. Polygon multicore parallel computing based on shape complexity partitioning.

4. Results

This paper reports the results of carrying out the intersection and difference operations for polygon overlay analysis via multicore parallel computing processes on land use patch data for China and land use data for several counties in Ningxia in 2009. Considering the three characteristics of running time, speedup, and the load balancing index, the performance of data partitioning based on polygon shape complexity and the performance of data partitioning based on polygon number are evaluated to verify whether the shape complexity-based method of data partitioning proposed in this paper is more advantageous for multicore parallel computing than previous methods are.

The multicore parallel running time of data partitioning based on the number of polygons refers to the total time for data partitioning, overlay analysis, and result merging. The multicore parallel running time of data partitioning based on shape complexity refers to the total time for shape complexity calculation, data partitioning, overlay analysis, and result merging.

The speedup ratio is defined as the ratio of the serial algorithm execution time to the parallel algorithm execution time and is calculated as shown in Equation (4):

$$S = T_1 / T \tag{4}$$

where S is the speedup ratio, T_1 is the running time of the serial algorithm, and T is the total running time of the parallel algorithm.

The load balancing index is defined as the ratio between the running time of the slowest thread and the running time of the fastest thread. The smaller this ratio is, the better the balance. The formula is shown in Equation (5):

$$F = T_{\max} / T_{\min} - 1 \tag{5}$$

where F is the load balancing index, T_{\max} represents the running time of the slowest thread among all threads participating in multicore parallel processing, and T_{\min} represents the running time of the fastest thread among all threads participating in multicore parallel processing.

4.1. Polygon Overlay Analysis Intersection Operator

To compare the performances of the different data partitioning methods for different amounts of data, the running time, acceleration ratio, and load balancing index were calculated. Figure 6 shows the results for the intersection operator for the polygon overlay analysis of the land use patch data for China and the land use data for several counties in Ningxia in 2009. When the number of threads is 1, the multicore parallel computing methods based on both types of data division are equivalent to the serial algorithm.

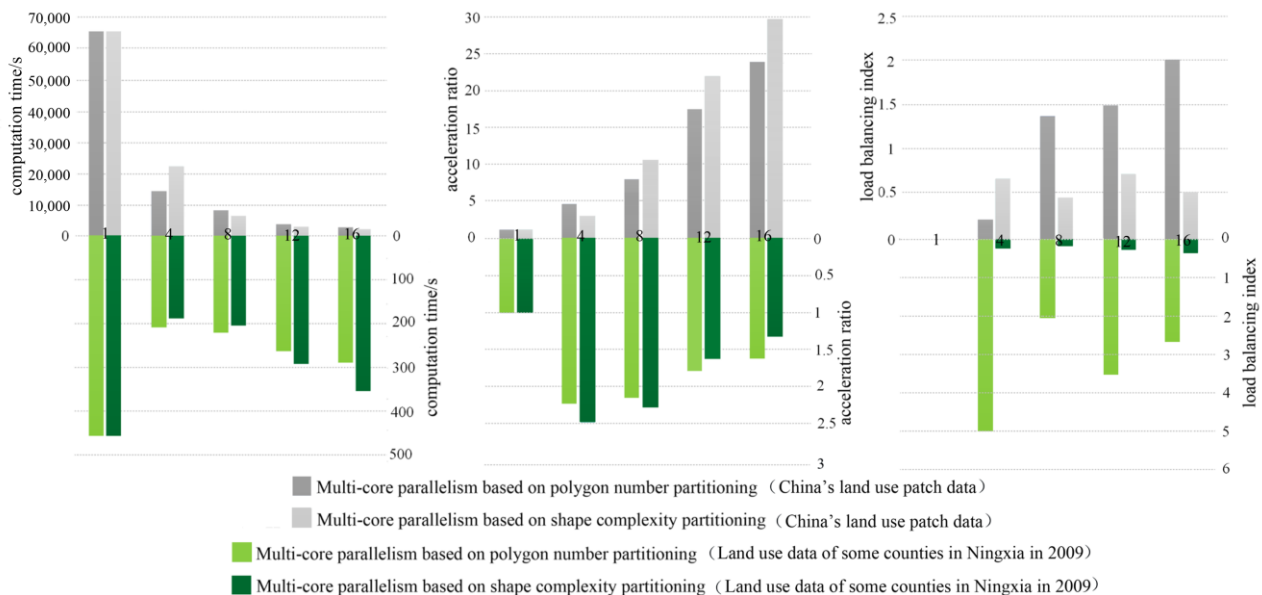


Figure 6. Multicore parallel intersection operator calculation results achieved with different methods for different amounts of data.

According to the test results for the multicore parallel intersection processing of the land use patch data for China, the serial running time of the intersection operator on the land use patch data for China is approximately 65,178 s, and the load balancing index is 0. When the number of threads is 4, the polygon number-based partitioning method is

superior to the shape complexity-based data partitioning method designed in this paper. The running time difference is approximately 7730 s, and the load balancing index is only 0.2, which indicates that the traditional polygon number-based partitioning method is more suitable when using four threads to perform intersection operations on a large amount of data. When the number of threads is greater than four, however, the method designed in this paper shows significant improvements compared with the traditional method. When 8 threads, 12 threads, and 16 threads are used, the effect is increased by 24%, 25%, and 33%, respectively. When the maximum number of threads in the parallel environment (16 threads) is reached, the running time is only 36 min, corresponding to an acceleration ratio of 29, which is 33% better than that of the polygon number-based partitioning method. According to Figure 6, when the number of threads is greater than four, the load balancing index under the proposed data partitioning method is significantly lower than that under the traditional partitioning method, which indicates that the data partitioning method based on shape complexity balances threads better than the data partitioning method based on the number of polygons. The running times of the two different data partitioning methods also gradually decrease with an increase in the number of CPU threads, and the acceleration ratio shows an upward trend.

Figure 6 also shows the test results for the multicore parallel processing of the intersection operator on land use data from several counties in Ningxia in 2009. Due to the small amount of data, the serial algorithm execution time is only approximately 451 s. When four or eight threads are used, the traditional polygon number division method is faster. When 12 or 16 threads are used, the data division method based on shape complexity designed in this paper is faster. Both data division methods can significantly shorten the running time compared to that of the serial algorithm, and their acceleration effects are similar. With an increasing number of CPU threads, the two data partitioning methods do not achieve greater computational efficiency. In terms of the load balancing index, however, the partitioning method based on shape complexity proposed in this paper has obvious advantages. Regardless of how many threads are selected for calculation, the load balancing index remains below 0.5, and the highest value of this index is only 0.33. With the data partitioning method based on the number of polygons, the load balancing index can only reach as low as 2 and can even exceed 7. This shows that for a small amount of data, the proposed data partitioning method based on shape complexity can achieve better load balancing when performing multicore parallel intersection calculations.

It can be seen from Figure 6 that when the Vatti algorithm intersection operator is executed in parallel on multiple cores, with the increase in the complexity of vector data, the data partitioning method based on shape complexity proposed in this paper can obtain higher speedup than the data partitioning method based on the number of polygons. In the face of data with different complexity, the load balancing performance of the proposed method can be significantly improved.

4.2. Polygon Overlay Analysis Difference Operator

Figure 7 shows the results obtained when applying the polygon overlay analysis difference operator to the different data sets using the different data division methods, including the running time, acceleration ratio, and load balancing index.

According to the multicore parallel computing results for the difference operator based on the land use patch data for China, the serial running time of the data is approximately 94,917 s. Figures 6 and 7 show that the calculation results are similar when performing intersection operations and difference operations on these data. When the number of threads is four, the traditional polygon number-based data partitioning method can achieve a better acceleration effect and better load balancing than the shape complexity-based data partitioning method designed in this paper. When the number of threads is 8, 12, or 16, the proposed data partitioning method can yield substantially improved results. Compared with that achieved with the traditional data partitioning method, the calculation speedup increased by 34%, 24%, and 29%, respectively, and the load balancing index is also

significantly reduced. When using the data partitioning method proposed in this paper to perform multicore parallel difference calculations with 16 threads, a 32-fold speedup can be achieved, and the running time is only 2966 s.

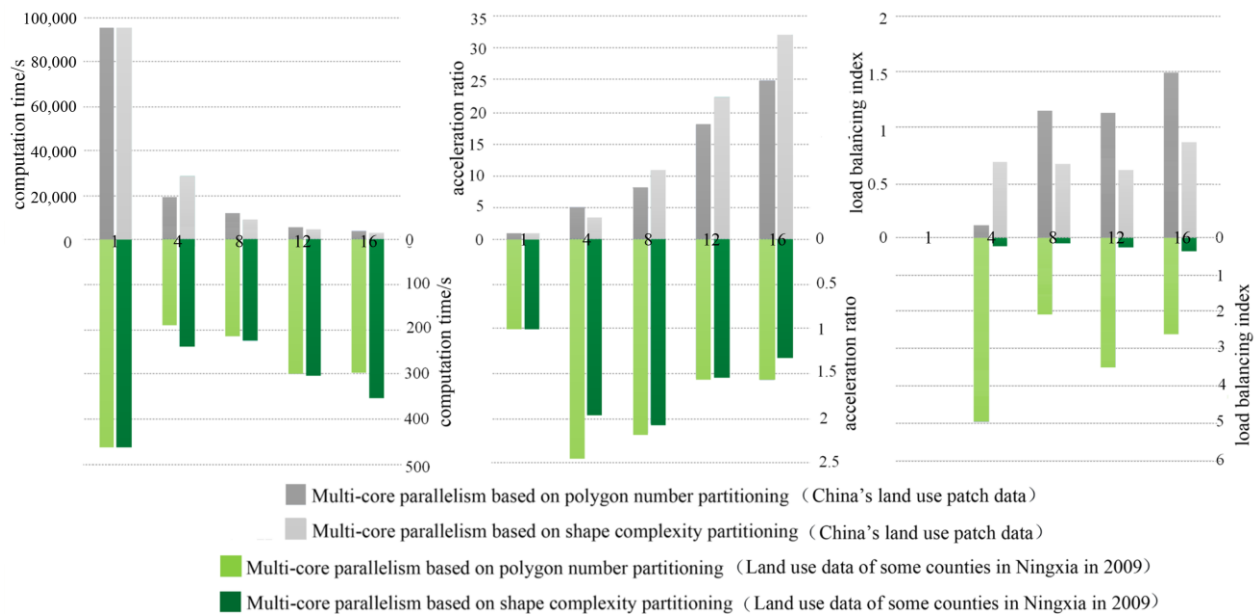


Figure 7. Multicore parallel difference operator calculation results achieved with different methods for different amounts of data.

Figure 7 also shows the multicore parallel computing results for the difference operator based on the land use data for several counties in Ningxia in 2009. The serial algorithm execution time on these data is 461 s. Figures 6 and 7 show that when either the intersection operator or the difference operator is applied to these data, both data partitioning methods improve the running time compared to that of the serial algorithm, and their improvement effects are roughly the same. However, the data partitioning method designed in this paper achieves better balancing of the multicore parallel difference calculations than does the traditional data partitioning method. The maximum load balancing index is only approximately 0.3, whereas the load balancing index of the traditional partitioning method exceeds 2.

According to Figure 7, when the multicore performs the Vatti algorithm difference operator in parallel, with the increase of the complexity of vector data, the acceleration ratio of the data partitioning method based on shape complexity proposed in this paper is significantly improved compared with the data partitioning method based on the number of polygons. Regardless of whether the data complexity is improved, the load balancing performance of this method can be significantly improved compared with the traditional method.

5. Discussion

Overlay analysis is one of the basic functions of GIS spatial analysis. When faced with a large amount of data, traditional serial algorithms take too much time. Parallelization after a reasonable division of data is an effective way to improve computational efficiency. Based on this idea, this paper designs a multicore parallelization method based on shape complexity-based data partitioning, which considers the influence of polygon structure, size, and other factors. By establishing a shape complexity model and applying it to divide the vector data, the grouped data can be distributed among multiple cores to execute polygon overlay analysis in parallel. Compared with the traditional multicore parallelization method based on dividing data by the number of polygons, the effectiveness of the proposed method in optimizing the execution of polygon overlay analysis algorithms is verified from three perspectives: the running time, the acceleration ratio, and the load balancing index.

The running speed of the method proposed in this study is greatly improved compared to that of the serial algorithm. Moreover, compared with those achieved with the traditional data partitioning method, the running speed and load balancing are both improved. In particular, when processing large-scale data, the speed is significantly improved, and the load balancing index is also significantly reduced. The running time of the multicore parallel algorithm depends on the running time of the slowest thread. Because the data partitioning method in this paper is more balanced, the running time of each thread is less different, which can effectively save the overall running time. Therefore, when the shape complexity of the data increases, the task time of each thread increases so that a higher speedup and a lower load balancing index can be obtained. The research results are beneficial for improving the efficiency of large-scale spatial overlay analysis and have certain reference values for the optimization of GIS spatial algorithms. Nevertheless, although the polygon shape complexity models proposed in this article accurately reflect the impact of polygon shape characteristics on the computational efficiency of overlay analysis, these models can still be improved because the structures of polygons can greatly differ. In this paper, the proposed multicore parallelization method based on shape complexity partitioning has been verified only for the intersection operator and difference operator of the Vatti algorithm, and additional in-depth research must be conducted to verify its broader applicability. In the future, we will consider calculating additional indicators that may affect the efficiency of polygon overlay analysis based on the shape characteristics and spatial characteristics of polygons and establish a shape complexity model that can better explain the observed algorithm performance. Moreover, this approach will be applied to data partitioning for various operators of other algorithms. Applying the data partitioning method based on the shape complexity model to applications for distributed parallel computing in a cloud environment and the cluster processing of large-scale vector data will also effectively improve the computing efficiency of spatial data.

6. Conclusions

In this study, six polygon shape indicators are selected to establish shape complexity models for data partitioning. Using vector data sets containing two different amounts of data, we perform multicore parallelization of the intersection operator and difference operator of the Vatti algorithm and analyze the results of the proposed multicore parallelization method in comparison with the results of the traditional method based on polygon number division. The research conclusions are as follows:

- (1) For large data sets, when the Vatti algorithm intersection operator and difference operator are executed on multicores in a parallel environment with more than four threads, the data partitioning method based on shape complexity significantly improves the speedup and load balancing performance compared with the data partitioning method based on the number of polygons.
- (2) For small data sets, multicore parallelization with data division based on shape complexity has acceleration effects similar to those of the traditional multicore parallelization method of dividing data based on polygon numbers. However, the proposed data partitioning method based on shape complexity can greatly improve the load balancing among threads.

In summary, the multicore parallelization method designed in this article based on shape complexity-based data partitioning can significantly improve the running speed of the Vatti algorithm and achieve better load balancing. When data complexity increases, higher speedups can be achieved. Relevant research results are highly important for overcoming bottlenecks such as balanced task decomposition and effectively improving the computing efficiency of parallel vector polygon overlay analysis algorithms in a big data environment.

Author Contributions: Conceptualization, J.F. and G.S.; Data curation, J.Z.; Formal analysis, J.Z. and Z.S.; Funding acquisition, J.F. and G.S.; Investigation, Z.S.; Methodology, J.F. and J.Z.; Resources, Z.S.,

Y.G. and Y.Z.; Software, J.Z. and Z.S.; Supervision, G.S.; Validation, J.Z.; Visualization, J.Z. and Y.G.; Writing—original draft, J.F. and J.Z.; Writing—review & editing, J.F., G.S. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 42171413), a grant from the State Key Laboratory of Resources and Environmental Information Systems, and the Natural Science Foundation of Shandong Province (No. ZR2020MD015).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: <http://www.openstreetmap.org>, accessed on 1 July 2023.

Acknowledgments: We would like to thank the OpenStreetMap. We appreciate the editors and reviewers for their constructive comments and suggestions.

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Li, L.Q.; Deng, M.; Liu, B.; Li, J. Design of an optimal algorithm of realizing spatial overlap analysis within GIS. *Shandong Keji Daxue Xuebao/J. Shandong Univ. Sci. Technol. (Nat. Sci.)* **2002**, *21*, 62–64. [\[CrossRef\]](#)
- Agarwal, D.; Puri, S.; He, X.; Prasad, S. A system for GIS polygonal overlay computation on linux cluster—an experience and performance report. In Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, Shanghai, China, 21–25 May 2012; pp. 1433–1439. [\[CrossRef\]](#)
- Shi, X. System and Methods for Parallelizing Polygon Overlay Computation in Multiprocessing Environment. U.S. Patent US20120320087A1, 14 June 2012.
- Ma, M.Y.; Wu, Y.; Chen, L.; Li, J.; Jing, N. Interactive and online buffer-overlay analytics of large-scale spatial data. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 21. [\[CrossRef\]](#)
- Zhou, C.; Chen, Z.J.; Li, M.C. A parallel method to accelerate spatial operations involving polygon intersections. *Int. J. Geogr. Inf. Sci.* **2018**, *32*, 2402–2426. [\[CrossRef\]](#)
- Dowers, S.; Gittings, B.M.; Mineter, M.J. Towards a framework for high-performance geocomputation: Handling vector-topology within a distributed service environment. *Comput. Environ. Urban Syst.* **2000**, *24*, 471–486. [\[CrossRef\]](#)
- Liu, Y.M.; Yang, J.; Puri, S. Hierarchical filter and refinement system over large polygonal datasets on cpu-gpu. In Proceedings of the IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC), Hyderabad, India, 17–20 December 2019; pp. 141–151. [\[CrossRef\]](#)
- Cramer, T.; Schmidl, D.; Klemm, M.; Mey, D.A. OpenMp programming on intel r xeon phi tm coprocessors: An early performance comparison. In Proceedings of the Many-Core Applications Research Community Symposium 2012, Aachen, Germany, 29–30 November 2012; pp. 38–44.
- Geer, D. Chip makers turn to multicore processors. *Computer* **2005**, *38*, 11–13. [\[CrossRef\]](#)
- Fan, J.F.; Ma, T.; Ji, M.; Zhou, Y.K.; Xu, T. Implementation and optimization of eight parallel polygon overlapping tools with OpenMP at the feature layer level in GIS. *Prog. Geogr.* **2013**, *32*, 1835–1844. [\[CrossRef\]](#)
- Jiang, Y.Y. Efficient Storage and Parallel Overlay Analysis of Massive Vector Data in Cloud Computing Environment. Master's Thesis, Kunming University of Science and Technology, Kunming, China, 2021. [\[CrossRef\]](#)
- Ye, J.Y.; Chen, B.; Chen, J.; Fang, Y.; Wu, L. A spatial data partition algorithm based on statistical cluster. In Proceedings of the 19th International Conference on Geoinformatics, Shanghai, China, 24–26 June 2011; pp. 1–6. [\[CrossRef\]](#)
- Mineter, M.J. A software framework to create vector-topology in parallel GIS operations. *Int. J. Geogr. Inf. Sci.* **2003**, *17*, 203–222. [\[CrossRef\]](#)
- Wang, S.W.; Cowles, M.K.; Armstrong, M.P. Grid computing of spatial statistics: Using the TeraGrid for G (d) analysis. *Concurr. Comput. Pract. Exp.* **2008**, *20*, 1697–1720. [\[CrossRef\]](#)
- Lee, C.K.; Hamdi, M. Parallel image processing applications on a network of workstations. *Parallel Comput.* **1995**, *21*, 137–160. [\[CrossRef\]](#)
- Zhou, Y.; Jiang, L. Hilbert curve based spatial data declustering method for parallel spatial database. In Proceedings of the 2nd International Conference on Remote Sensing, Environment and Transportation Engineering, Nanjing, China, 1–3 June 2012; pp. 1–4. [\[CrossRef\]](#)
- Yang, Y.Z.; Wu, L.X.; Guo, J.T.; Li, Z.F.; Liu, S.J. A method of spatial data partition for efficient parallel computing of topological relation. *Geogr. Geo-Inf. Sci.* **2013**, *29*, 25–29.
- Wu, X.Q.; Wu, Y.; Chen, L.; Jing, N. A parallel cut-fill algorithm for largescale DEM data. *Geomat. World* **2019**, *26*, 21–25.
- Zhang, Z.K.; Fan, J.F.; Xu, S.B.; Chen, Z. VCS Optimization Method of Vatti Algorithm for Polygon Overlay and Parallelization Using GPU. *J. Geo-Inf. Sci.* **2022**, *24*, 437–447. [\[CrossRef\]](#)

20. Jiang, Y.Y.; Jin, B.X.; Zhao, K.; Zhou, S.Y. Research on measurement of polygon shape complexity in overlay calculation. *Sci. Surv. Mapp.* **2020**, *45*, 177–184. [[CrossRef](#)]
21. Tilove, R.B. Line/polygon classification: A study of the complexity of geometric computation. *IEEE Comput. Graph. Appl.* **1981**, *1*, 75–88. [[CrossRef](#)]
22. Attneave, F. Physical determinants of the judged complexity of shapes. *J. Exp. Psychol.* **1957**, *53*, 221. [[CrossRef](#)] [[PubMed](#)]
23. Chen, Y.P.; Sundaram, H. Estimating complexity of 2D shapes. In Proceedings of the IEEE 7th Workshop on Multimedia Signal Processing, Shanghai, China, 30 October–2 November 2005; pp. 1–4. [[CrossRef](#)]
24. Duan, X.Y.; Deng, X.X.; Zuo, Q.Y. The research on the complexity of progressive die edge. *J. Eng. Graph.* **2006**, *5*, 94–97. [[CrossRef](#)]
25. Brinkhoff, T.; Kriegel, H.P.; Schneider, R.; Braun, A. Measuring the Complexity of Polygonal Objects. In Proceedings of the ACM-GIS, Baltimore, MD, USA, 28 November–2 December 1995; Volume 109.
26. Matsumoto, T.; Sato, K.; Matsuoka, Y.; Kato, T. Quantification of “complexity” in curved surface shape using total absolute curvature. *Comput. Graph.* **2019**, *78*, 108–115. [[CrossRef](#)]
27. Guo, M.Q.; Guan, Q.F.; Xie, Z.; Wu, L.; Luo, X.G.; Huang, Y. A spatially adaptive decomposition approach for parallel vector data visualization of polylines and polygons. *Int. J. Geogr. Inf. Sci.* **2015**, *29*, 1419–1440. [[CrossRef](#)]
28. Li, A.B.; Chen, Y.; Yao, M.M.; Wu, S.S. Quantitative measurement of geometrical information for sensitive features in secret-related vector digital maps. *J. Geo-Inf. Sci.* **2018**, *20*, 7–16. [[CrossRef](#)]
29. Zhang, P.; Fan, J.F.; Zhang, P.P.; Zhang, Z.K.; Chen, Z.; Han, L.S. Comparative Study on the Effect of Shape Complexity on the Efficiency of Different Overlay Analysis Algorithms. *IEEE Access* **2021**, *9*, 144179–144194. [[CrossRef](#)]
30. Wang, Z.J.; Lin, X.; Fang, M.E.; Yao, B.; Peng, Y.; Guan, H.B.; Guo, M.Y. Re2l: An efficient output-sensitive algorithm for computing Boolean operations on circular-arc polygons and its applications. *Comput.-Aided Des.* **2017**, *83*, 1–14. [[CrossRef](#)]
31. Liang, Y.D.; Barsky, B.A. An analysis and algorithm for polygon clipping. *Commun. ACM* **1983**, *26*, 868–877. [[CrossRef](#)]
32. Sutherland, I.E.; Hodgman, G.W. Reentrant polygon clipping. *Commun. ACM* **1974**, *17*, 32–42. [[CrossRef](#)]
33. Weiler, K.; Atherton, P. Hidden surface removal using polygon area sorting. *ACM SIGGRAPH Comput. Graph.* **1977**, *11*, 214–222. [[CrossRef](#)]
34. Vatti, B.R. A generic solution to polygon clipping. *Commun. ACM* **1992**, *35*, 56–63. [[CrossRef](#)]
35. Greiner, G.; Hormann, K. Efficient clipping of arbitrary polygons. *ACM Trans. Graph. (TOG)* **1998**, *17*, 71–83. [[CrossRef](#)]
36. Liu, Y.K.; Gao, Y.; Huang, Y.Q. An efficient algorithm for polygon clipping. *J. Softw.* **2003**, *14*, 845–856.
37. Martinez, F.; Rueda, A.J.; Feito, F.R. A new algorithm for computing Boolean operations on polygons. *Comput. Geosci.* **2009**, *35*, 1177–1185. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.