2016 15th International Symposium on Parallel and Distributed Computing

Distributed and Parallel Delaunay Triangulation Construction with Balanced Binary-tree Model in Cloud

Jiaxiang Lin*†, Riqing Chen*, Changcai Yang*, Zhaogang Shu*, Changying Wang*, Yaohai Lin*, Liping Wu*
*College of Computer and Information Sciences
Fujian Agriculture and Forestry University, Fuzhou 350002, China
Email: {riqing.chen, changcaiyang}@fafu.edu.cn
†Institute of Geographic Sciences and Natural Resources Research,
CAS, Beijing 100101, China
Email: linjx@fafu.edu.cn

***Abstract*—Delaunay triangulation (D-TIN) is an important graphic tool in computational geometry, which is not only widely used in many real applications, but also very significant for many spatial data mining algorithms. However, constructing Delaunay triangulation is time-consuming for most practical applications. Distributed and parallel computing mechanism is becoming a good choice to solve large scale and compute-intensive D-TIN applications. This paper proposes a novel hybrid algorithm (*HA*) for D-TIN construction in cloud computing environment, which is based on a balanced binary-tree model and an elegant data structure called quad-edge. *HA* combines the divide & conquer approach and the incremental method. Moreover, a distributed and parallel version of Delaunay triangulation computing service in cloud is designed and implemented.**

The hybrid algorithm performed in both centralised and in cloud environments are compared. Experimental results showed that the hybrid D-TIN service outperforms both the the divide & conquer one and the incremental one, and it can effectively provide higher data mining services with fundamental D-TIN construction function in cloud.

1. Introduction

Delaunay triangulation is the geometric dual of Voronoi diagram, and it has been one of the fundamental data structures of computational geometry over the past few decades [1]. For a dataset with n sites, the Voronoi Diagram is a subdivision of the plane into n regions, each region corresponds to one site. The region for a given site consists of that portion of the plane closer to it than to any other sites. Both Delaunay triangulation and Voronoi Diagram are widely used in many applications, such as geographical information system (GIS), terrain analysis, computer graphics and computer vision, virtual manufacturing, finite element analysis, road CAD technology, etc. [2], [3].

Delaunay triangulation and its dual Voronoi diagram are widely used to capture the spatial neighborhood relationships between geographical objects. Geographical knowledge can be extracted from the dataset according to the spatial proximity represented by Delaunay triangulation [4]. Hence, to design and develop graphics tools and related algorithms to compute Delaunay triangulation is interesting in the area of knowledge discovery in spatial databases.

The strategies to compute Delaunay triangulation for a set P of n sites can be divided into two categories: the direct and the indirect methods. The indirect method is to construct the Voronoi diagram ($Vor(P)$) first, and then obtains its dual Delaunay triangulation $T(P)$; while the direct methods compute Delaunay triangulation directly, typical direct methods include randomised incremental algorithm (IA), divide & conquer method (D&C), and plane sweep-line approach [5]. Time complexity of the incremental algorithm for inserting n sites is approximately $O(n^2)$, and both the divide & conquer method and the plane sweep-line approach have intrinsic predominance in term of time complexity $O(n \log n)$. However, the incremental algorithm is competitive with regard to the other methods for several reasons [6]. First, IA is much easier to implement due to its simplicity, and it is applicable to various datasets especially when their sizes increase dynamically. Second, if the site set is sampled with a uniform probability distribution, the expected time for each insertion is small and roughly independent of n, the processing time is then dominated by site location. For the simple walk algorithm [7], [8], to locate a site in an existing triangulation, the expected time is roughly $O(n^{\frac{1}{2}})$ for each site, so the total computational time of IA is $O(n^{\frac{3}{2}})$ [9]. As a result, the performance can be quite acceptable in many real-world applications, even when the sites are given in advance. Thirdly, in many practical situations successive sites tend to be close to each other. Consequently, the edge returned in the previous call can be used by the location process as its starting point, each insertion may take roughly a constant time. In such cases, the incremental method may perform better than the divide-and-conquer even for a large number of sites [10].

In many GIS applications, datasets are distributed among different geographical locations, so the construction of Delaunay triangulation both on local datasets and global dataset are usually required [11], [12]. This constitutes a preliminary work on performing distributed spatial data mining. The main goal of this paper is to focus on a hybrid algorithm to construct local and global Delaunay triangulations simultaneously and to explore an effective mechanism and implementation of distributed & parallel D-TIN in a geographical knowledge service cloud, called GeoKSCloud. This will take a full advantage of the high-performance computing capability of cloud and provide high-level spacial knowledge gird services with complete spatial neighbourhood relation graph.

The rest of this paper is organised as follows: Section 2 describes in detail the hybrid algorithm to construct Delaunay triangulation. In Section 3, we elaborate on the design and implementation of a distributed & parallel Delaunay triangulation service in GeoKSCloud platform. Section 4 presents experimental results with contrast evaluation of the hybrid algorithm on centralised and cloud environment. Finally, we conclude in Section 5.

2. Hybrid Algorithm for Delaunay Triangulation

In this section, we introduce a hybrid technique for performing Delaunay triangulation. The technique combines the divide & conquer and the incremental algorithms. Consider a site set P of n points in the plane and let θ be a threshold which is set to a default value $\log_2 n$. The basic idea of the hybrid algorithm (HA) is described as follows:

If the scale of problem P is under threshold θ, then incremental algorithm (IA) is used directly to compute the corresponding Delaunay triangulation. Otherwise, the problem P is partitioned into two approximately equal size of sub-problems, namely left sub-problem (L) and right sub-problem (R), and the related Delaunay graphs $T(L)$ and $T(R)$ of L and R sub-problems are computed respectively. During the process of problem partition, we continue to divide the sub-problems recursively until the scale of sub-problems (L_i) and (R_i) is less than θ. Subsequently, incremental algorithm is adopted to construct Delaunay triangulations for (L_i) and (R_i), and the result Delaunay triangulations $T(L_i)$ and $T(R_i)$ for (L_i) and (R_i) are merged gradually until the global Delaunay triangulation is obtained. The principle of problem decomposition and sub-Delaunay triangulations merging for the hybrid algorithm (HA) is shown in Fig 1.

Two main steps of HA algorithm: "the divide and conquer" and "the incremental" are more adequate to run on distributed platforms and mainly for very large Delaunay Decomposition problems. Through tight combination of these two steps this new approach inherits the advantages of high-adaptability, simplicity of the incremental algorithm and the advantages of high-efficiency, autonomy of divide & conquer algorithm [13]. At the same time, it overcomes the fatal drawback of a large number of recursions involved in divide & conquer algorithm without a significant reduction in the efficiency. In particular, in a distributed environment, it avoids a large number of network communications and hence improves its performance.

3. Distributed D-TIN in Cloud Environment

GeoKSCloud currently includes all the features owned by computational cloud and data cloud. Once a knowledge service is deployed; it can be monitored, discovered, shared and called by every node of the cloud. After introducing large amounts of spatial data mining services and spatial decision-support services, GeoKSCloud has different kinds of knowledge service capabilities, such as spatial clustering, spatial association rule mining, spatial outlier detection and urban air pollutant dispersion simulation that are needed by various applications. Through years of research on spatial data mining algorithms and the mechanisms of cloud service interoperability, several main function modules have reached maturity and provide better and reliable distributed spatial knowledge services with OGSA architecture. Typical modules include cloud resource center, cloud information center, execution management center, knowledge service center, and cloud platform management center with a uniform cloud portal interface. Moreover, GeoKSCloud provides a novel problem solving environment that will enable the in-depth study of distributed Delaunay triangulation algorithms in cloud environment.

In GeoKSCloud environment, the new approach consists of two atomic services, namely the *divide and conquer* service and the *incremental* service. The two services are encapsulated and deployed independently as atomic cloud services on different cloud nodes in GeoKSCloud. Jointly, called `D-TIN service`, the two services constitute an integrated distributed Delaunay triangulation services. Once a cloud service is successfully started in any cloud node, it is ready for the client to invoke a cloud service everywhere. More detail information about the mechanism of parallel and distributed computing in GeoKSCloud can be found in [14], [15]. The key steps of a distributed version of the hybrid algorithm in a cloud environment are as follows:

1) Sort the sites in ascending order, that is $(x_i, y_i) < (x_{i+1}, y_{i+1})$ if and only if $x_i < x_{i+1}$, or $x_i = x_{i+1}$ and $y_i < y_{i+1}$.
2) Find an available cloud node to carry out the D-TIN construction on the site-set P. This initial cloud node should be able to invoke the hybrid D-TIN service.
3) If the problem size is smaller than a specified threshold, compute the Delaunay triangulation directly and return the result of D-TIN to the user.
4) If the problem size is greater than the threshold, partition the problem into two adjacent sub-problems of approximately equal size, and find two available cloud nodes to deal with them. GOTO Step 3.
5) Once the division of the initial problem into small sub-problems has been finished, the sub-problems

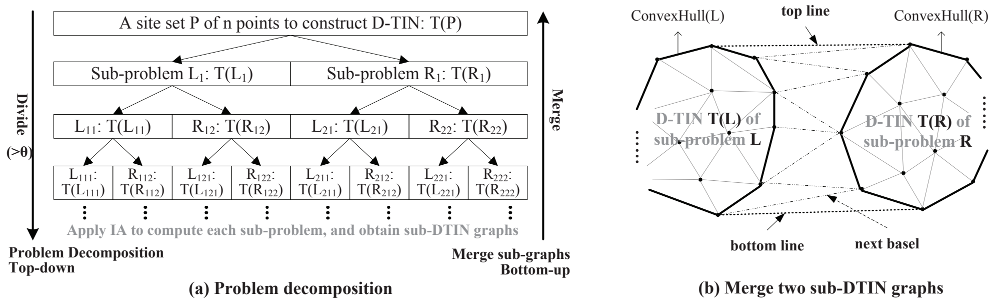


Figure 1. Problem decomposition and sub-Delaunay triangulations merging in HA.

are performed in parallel on their corresponding cloud-nodes. Each sub-problem will generate a sub-graph. This partitioning forms a binary tree, in which the leaves represent the sub-problems.

6) Merge every two adjacent sub-graphs starting from the bottom (leaf nodes) of the tree using the merge algorithm of divide & conquer. This is repeated at every level of the tree until we reach the root of the tree, which represents the initial cloud node with the whole site set P.

3.1. GeoKSCloud Processes

The execution and scheduling flowchart of distributed & parallel Delaunay triangulation in GeoKSCloud is shown in Fig 2. When a Delaunay triangulation is submitted and registered to the cloud platform, the execution management center (refers to GRAM by Globus) will find and invoke an available hybrid D-TIN service to perform the task.

In practical D-TIN construction applications, after a job has been submitted to the cloud platform, a hybrid D-TIN service can be invoked. If the problem scale is smaller than the threshold, the execution management center will invoke a cloud server to deal with the problem. If the scale of the problem is greater than the threshold, two cloud servers are invoked; one for each sub-problem.

As shown in Fig 2, the merge step keeps waiting until all the two sub-problems have been solved and the sub-graphs have been returned to their parents. In order to optimize the distributed version of the hybrid algorithm, for both computation and communication times, the cloud platform will allocate only an extra node instead of two to complete the two sub-tasks. Therefore, one needs to find a cloud server to deal with one sub-problem while the other sub-problem is processed by the local node. In this way, not only it reduces the response time due to data transmission between nodes, but also reduces the application requirements in terms of resources.

3.2. Implementation

The most important task of the proposed distributed & parallel D-TIN construction service in a cloud is to implement two atomic D-TIN services: the Hybrid D-TIN service and the incremental D-TIN service.

Guibas and Stolfi [16], [17] proposed an elegant quad-edge data structure for representing graph embedding on two-dimensional manifolds, which simultaneously represents a structure and its dual. In this paper, this idea is adopted to represent Delaunay triangulation. The implementation of the divide & conquer algorithm, incremental algorithm and the hybrid algorithm benefitted hugely from quad-edge data structure and mainly the integration of the incremental algorithm into the divide & conquer algorithm.

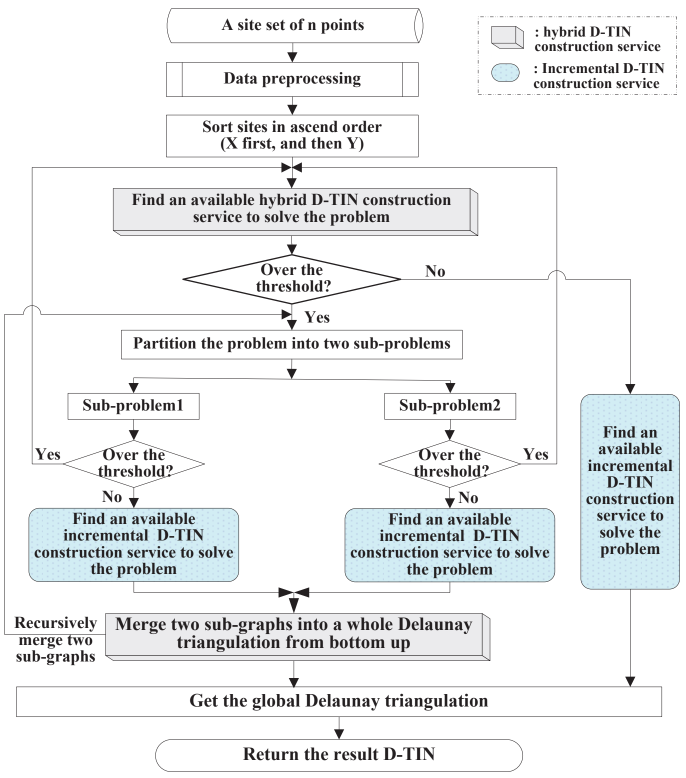
The pseudocode for the hybrid algorithm of D-TIN construction is shown in Algorithm 1, which greatly exhibits the idea of divide & conquer. The two sub-problems L & R are processed in parallel when the scale of the problem is higher than the threshold. As to the principle of cloud encapsulation and deployment of D-TIN service, it can be referred in [18], [19]

Algorithm 1 Pseudo-code of Hybrid D-TIN Service.

```
Input: A site set $s[]$ in ascending order.
        A Specified threshold $\theta$ for distributed D-TIN construction
Output: Delaunay Triangulation for the site $s[]$.
Procedure CreateDTINbyDistributedHybrid($s[]$)
return D-TIN

// if problem scale $\leq \theta$, find an incremental service to do the job
If ($size \leq \theta$) //invoke an incremental D-TIN service
        Return invokeCloudService(S, "Incremental");

// problem scale $> \theta$, divide $S$ into 2, find 2 hybrid service to solve them
D-TIN LeftDtin = invokeCloudService(L, "Hybrid"); //for L
D-TIN RightDtin = invokeCloudService(R, "Hybrid"); //for R
```



A site set of n points
: hybrid D-TIN construction service
: Incremental D-TIN construction service
Data preprocessing
Sort sites in ascend order (X first, and then Y)
Find an available hybrid D-TIN construction service to solve the problem
Over the threshold?
No
Yes
Partition the problem into two sub-problems
Sub-problem1
Sub-problem2
Yes
Over the threshold?
No
Over the threshold?
Yes
No
Find an available incremental D-TIN construction service to solve the problem
Find an available incremental D-TIN construction service to solve the problem
Find an available incremental D-TIN construction service to solve the problem
Recursively merge two sub-graphs
Merge two sub-graphs into a whole Delaunay triangulation from bottom up
Get the global Delaunay triangulation
Return the result D-TIN

Figure 2. Distributed D-TIN Construction in Cloud Environment.

Algorithm 2 Pseudo-code of Incremental D-TIN Service.

```
Input: A site set $s[]$
Output: Delaunay Triangulation for the site $s[]$.

Procedure CreateDTINbyIncremental($s[]$) return D-TIN
Return getDTINbyInsertion(s); //create D-TIN by incremental
```

3.3. Cost Model

Assume that the size of D-TIN computing problem is n, and the threshold for parallel & distributed D-TIN computing is θ. Let k be the number of nodes needed to process sub-problems of size $m \leq \theta$. Then, the number of subdivisions is easy to calculate according to the mechanism of the hybrid algorithm. Based on the algorithm described above, the division of the problem consists of a binary tree. Therefore, k is of the form $k = 2^{\omega}$, such that $2^{\omega-1} < \lceil \frac{n}{\theta} \rceil \leq 2^{\omega}$.

For a site set of n points, the hybrid algorithm divides the problem recursively into two approximately equal sub-problems. Let $T_1(n)$ be the response time for the sequential version and $T_k(n)$ be the response time for the distributed version with k processors. $T_1(n)$ would be the sum of the computation time by the incremental algorithm and the merging time of the two sub-problems, while $T_k(n)$ would be the sum of the computation time by the incremental algorithm, the merging time of the two sub-problems and the communication time between the server and the clients. Let $T_{IA}(m)$ be the computation time of incremental algorithm for a problem of size m, $T_{merge}(m)$ be the time for merging two subproblems of size $\frac{m}{2}$ each and $T_{comm}(m)$ be the time for exchanging dataset of size m among the servers and clients. So the response time for the sequential version $T_1(n)$ and the distributed version $T_k(n)$ would be:

$$T_1(n) = kT_{IA}(m) + \sum_{i=1}^{\omega} \frac{k}{2^i} T_{merge}(2^i m) \tag{1}$$

$$T_k(n) = T_{IA}(m) + \sum_{i=1}^{\omega} \left(T_{merge}(2^i m) + T_{comm}(2^{i-1} m)\right) \tag{2}$$

Because Delaunay triangulation can be computed in $O(n \log n)$ by using the divide & conquer algorithm and in $O(n^2)$ by using the incremental algorithm. Therefore, time complexity for D-TIN construction in the leaf node with m points would be $O(m^2)$ by the incremental algorithm, and the corresponding time for merging two sub-problems *L* and *R* in the internal nodes would be $O(2m)$ by the hybrid algorithm.

As for the communication time in the distributed version of D-TIN construction, a data structure *QuadEdge* is adopted to fully express the spatial topology of the Delaunay triangulation and its dual, so the main source of data transmission between the clients and the servers is a series of *QuadEdge*. For a site set of n points, there are $3n + 6$ edges at most in the corresponding D-TIN. As a result, the communication time depends on the number of *QuadEdge*.

To sum up, the response time for the sequential version T_1 is as follows:

$$\begin{aligned} T_1(n) &= km^2 + [\frac{k}{2}(2m) + \frac{k}{2^2}(2^2m) + \cdots + \frac{k}{2^\omega}(2^\omega m)] \\ &= km(m + \omega), \end{aligned} \tag{3}$$

Similarly, the response time for the distributed and parallel version T_k is deduced as follows, because the sub-problems with the same level in the binary tree structure are computed in parallel.

$$\begin{aligned} T_k(n) &= m^2 + (2m + m) + (2^2m + 2m) + \cdots \\ &+ (2^\omega m + 2^{\omega-1}m) \\ &= m(m + 3(k-1)) \end{aligned} \tag{4}$$

From the equations (3) and (4), we can see that the distributed version, $T_k(n)$, outperforms the sequential version, $T_1(n)$, by a factor k, especially when the problem size becomes larger and enough cloud nodes are available for the distributed version of D-TIN computing. Moreover, according to the principle of the hybrid algorithm, it has two special cases: 1) it can degenerate into the divide & conquer algorithm if the threshold θ is smaller than 1. In this case we need as many processing nodes as the size of the problem. Therefore the communication time may dominate the whole execution, which is not effective way of distributing the algorithm. The second case is that the hybrid algorithm can degenerate into the incremental algorithm if the threshold $\theta \geq n$. In this case the number of processing nodes will be $k = 1$, and the response time will be equal to the sequential version's response time.

4. Experiments and Discussion

We perform some experiments and compare the performance of the hybrid algorithm of Delaunay triangulation executed on one machine (sequential implementation) with the distributed D-TIN service deployed in GeoKSCloud platform. In the experiments, different problem sizes were tested with a threshold $\theta \in [\frac{n}{8}, \frac{n}{4}]$. Following the process of distributed scheduling of the hybrid D-TIN service, the problem is recursively divided into two sub-problems, find and invoke $\frac{k}{2}$ different cloud nodes with the hybrid D-TIN service and k cloud nodes with the incremental D-TIN service. Fig 3 shows an example of this process with a site set of 10000 points and a threshold of 2000. In the figure, the rectangle denotes the hybrid D-TIN service, and the rounded rectangle denotes the incremental D-TIN service. Both the hybrid and the incremental D-TIN services are deployed over the GeoKSCloud nodes.

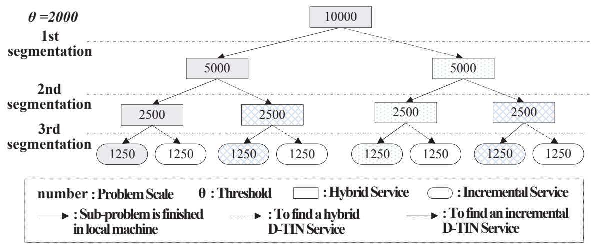


Figure 3. Example of distributed D-TIN construction with $n = 10000, \theta = 2000$.

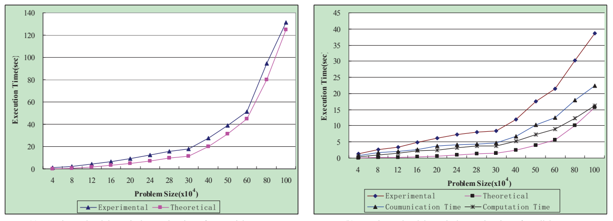
The number of sites and the corresponding executions time are listed in the Table 1. The experimental results indicate that the distributed & parallel D-TIN construction service based on the proposed hybrid algorithm can be used to compute Delaunay triangulation in cloud environment in an efficient way. One can notice from the results that the distributed version outperforms the sequential version, especially when m is becoming larger and larger. This is not really a surprise since more processing nodes are expected to perform better than one, as it is expected in the theoretical model.

Moreover the performance issue is not in the computation time, but in the communication time and the way that the algorithm was parallelized on k processing nodes. Therefore, the speed-up of the distributed version of the algorithm can be affected heavily by the overheads due to the communication times and other system activities. One can notice that for smaller sizes of the problem, the distributed version performs poorly against the sequential version, as the communications count for large part of the response time. From the last 3 columns of Table 1, we can conclude that the communication time of the distributed version counts for a significant proportion of the global response time. However, the distributed implementation is much better when the scale of the problem increases gradually.

Fig 4 shows the experimental and theoretical execution times of sequential and parallel versions of D-TIN based on the experimental results reported in Table 1 and the cost model developed in Section 3.3. Fig 4 (a) and (b) follow exactly what was expected in theory (see equations 3 and 4). Particularly, the communication time between nodes in the distributed version is well inline with the experimental results. The difference between the theoretical and the experimental results is due to some other system activities, such as the scheduling time of cloud services. Moreover, we considered in our theoretical model that the communication time is proportional to the size of the problem (linear progression), which is not really the case on cloud platforms due to the network traffic and heterogeneity of their resources. This is also reflected in these results. Overall results of the distributed version of the hybrid algorithm are promising and mainly for very large size of the problem.

Prob.Size	Sequential Impl		Para & Dist Impl ($\theta = \lceil \frac{n}{8} \rceil$)		
(n)	$\theta = \log_2 n$	$\theta = \lceil \frac{n}{8} \rceil$	Comm time	Comp time	Total time
40000	1.422	0.879	0.837	0.454	1.292
80000	2.797	2.291	1.603	0.939	2.541
120000	4.349	4.137	1.985	1.457	3.442
160000	6.265	6.428	2.590	2.291	4.880
200000	7.677	9.203	3.662	2.410	6.073
240000	9.563	12.164	4.053	3.213	7.266
400000	16.411	27.793	6.724	5.182	11.906
600000	26.349	51.533	12.480	8.994	21.474
800000	35.693	94.398	17.888	12.274	30.162
1000000	44.505	131.182	22.462	16.163	38.625

TABLE 1. TIME CONSUMPTION OF D-TIN CONSTRUCTION UNDER DIFFERENT CONDITIONS (TIME: SEC).



(a) Experimental and theoretical execution time of sequential D-TIN (b) Experimental and theoretical execution time of parallel D-TIN

Figure 4. Experimental and theoretical execution time of sequential and parallel D-TIN.

5. Conclusion

In this paper, a novel hybrid algorithm to construct Delaunay triangulation in the plane was presented, which combines the algorithms of divide & conquer and the incremental. Through both theoretical and experimental results of the approach on stand-alone and in cloud environment, the hybrid algorithm outperforms the incremental algorithm, but it is a bit weaker than the divide & conquer algorithm, which also have an time complexity of $O(n \log n)$ for smaller sizes of the problem.

An important advantage of the hybrid algorithm lies in its simplicity to be transformed and deployed in a distributed environment. Therefore, we designed and implemented a distributed Delaunay triangulation construction service in GeoKSCloud, to provide a general cloud service of D-TIN construction for some higher knowledge services in the platform. Although, there are many restrictions on getting more efficient distributed applications in GeoKSCloud, the recent advances in software cloud middleware will provide practical distributed real-world applications, and with better distributed & parallel Delaunay triangulation construction services in the future.

6. Acknowledgement

This research is supported partially by the NSF of China under Grants 41401458, the National Sci. & Tech. Development Program under Grant 2013BAH28F00, the EC FP7-2009-People-IRSES under Grant 247608, the NSF of Fujian under Grants 2014J05045 and Fujian Sci. & Tech. program under Grant 2010I0008, and Young teachers' scientific research projects of FAFU under Grant k13xjj05a.

References

[1] M. D. Berg, O. Cheong, M. V. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. Germany: Springer-Verlag, 2008.

[2] K. F. Mulchrone, "Application of delaunay triangulation to the nearest neighbour method of strain analysis," *Journal of Structural Geology*, vol. 25, no. 5, pp. 689–702, 2003.

[3] X. Li, H. Wu, and H. Xue, "A new large-scale terrain real-time simplification algorithm based on delaunay triangulation," in *2007 Second International Conference on Space Information Technology*, vol. III. Wuhan, China: SPIE, Bellingham, Wash., USA., 2007, pp. 67 953W.1–67 953W.7.

[4] X. Wang, C. Rostoker, and H. J. Hamilton, "Density-based spatial clustering in the presence of obstacles and facilitators," *Lecture Notes in Computer Science: Knowledge Discovery in Databases (PKDD 2004)*, vol. 3202, pp. 446–458, 2004.

[5] K. Buchin and W. Mulzer, "Delaunay triangulations in o(sort(n)) time and more," *Journal of the ACM (JACM)*, vol. 58, no. 02, pp. 1–27, 2011.

[6] J.-D. Boissonnat, O. Devillers, and S. Hornus, "Incremental construction of the delaunay triangulation and the delaunay graph in medium dimension," in *2009 Twenty-fifth Annual Symposium On Computational Geometry (SCG'09)*, Aarhus, Denmark, 2009, pp. 208–216.

[7] M. A. Mostafavi, C. Gold, and M. Dakowicz, "Delete and insert operations in voronoi/delaunay methods and applications," *Computers & Geosciences*, vol. 29, no. 4, pp. 523–530, 2003.

[8] H. Zhao and M. Bikdash, "Algorithm to locate points in a delaunay triangulation," in *2006 38th Southeastern Symposium on System Theory*, Cooksville, TN, Canada, 2006, pp. 211–215.

[9] O. Devillers, "Improved incremental randomized delaunay triangulation," in *1998 fourteenth annual symposium on Computational Geometry*. Minneapolis, Minnesota, USA: ACM New York, NY, USA, 1998, pp. 106–115.

[10] J. W. Laarhoven and J. W. Ohlmann, "A randomized delaunay triangulation heuristic for the euclidean steiner tree problem in r_d," *Journal of Heuristics*, vol. 17, no. 04, pp. 353–372, 2011.

[11] G. Simon, M. Steiner, and E. Biersack, "Distributed dynamic delaunay triangulation in d-dimensional spaces," Technical report, Institut Eurecom., Tech. Rep., 2005.

[12] Y. Zhang, Q. Gao, L. Gao, and C. Wang, "imapreduce: A distributed computing framework for iterative computation," *Journal of Grid Computing*, vol. 10, no. 1, pp. 47–68, 2012.

[13] Y. Rui, J. Wang, C. Qian, J. Liu, and X. Li, “A new compound algorithm study for delaunay triangulation construction,” in *2007 15th International Conference on Geoinformatics: Cartographic Theory and Models*, vol. 6751, Nanjing, China, 2007, pp. B7510–B7510.

[14] J. Lin, C. Chen, D. Ye, and W. Wang, “Research on grid based spatial outliers mining and its implementation,” in *2008 3rd International Conference on Intelligent System and Knowledge Engineering (ISKE 2008).* Xiamen, China: IEEE, 2008, pp. 324–328.

[15] J. Lin, C. Chen, Q. Wang, W. Wang, and J. Wu, “Distributed spatial data mining in geospatial knowledge service grid,” in *2010 2nd International Conference on Advanced Geographic Information Systems, Applications, and Services (GEOProcessing 2010)*, P. Kellenberger, Ed. St. Maarten, Netherlands Antilles: IEEE, 2010, pp. 80–87.

[16] L. J. Guibas, D. E. Knuth, and M. Sharir, “Randomized incremental construction of delaunay and voronoi diagrams,” *Algorithmica*, vol. 7, no. 1, pp. 381–413, 1992.

[17] L. Guibas and J. Stolfi, “Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams,” *ACM Transactions on Graphics*, vol. 04, no. 04, pp. 74–123, 1985.

[18] C. Chen, J. Lin, X. Wu, and J. Wu, “Parallel and distributed spatial outlier mining in grid: Algorithm, design and application,” *Journal of Grid Computing*, vol. 13, pp. 139–157, 2015.

[19] N. Muthuvelu, I. Chai, E. Chikkannan, and R. Buyya, “Qos-based task group deployment on grid by learning the performance data,” *Journal of Grid Computing*, vol. 12, no. 3, pp. 465–483, 2014.