Journal of Data Intelligence, Vol. 1, No. 2 (2019) 159–175

A PARALLELIZING METHOD FOR GENERATION OF VORONOI DIAGRAM USING CONTACT ZONE

YUUHI OKAHANA
Graduate School of Natural Science and Technology, Okayama University, 3-1-1, Tsushima-naka, Kita-ku, Okayama, 700-8530, JAPAN
pmqb1t6p@s.okayama-u.ac.jp

YUSUKE GOTOH
Graduate School of Natural Science and Technology, Okayama University, 3-1-1, Tsushima-naka, Kita-ku, Okayama, 700-8530, JAPAN
gotoh@cs.okayama-u.ac.jp

Due to the recent popularization of the Geographic Information System (GIS), spatial network environments that can display the changes of spatial axes on mobile devices are receiving great attention. In spatial network environments, since a query object that seeks location information selects several candidate target objects based on the search conditions, we often use a k-nearest neighbor (kNN) search, which seeks several target objects near the query object. However, since a kNN search needs to find the kNN by calculating the distance from the query to all the objects, the computational complexity might become too large based on the number of objects. To reduce this computation time in a kNN search, many researchers have proposed a search method that divides regions using a Voronoi diagram. However, since conventional methods generate Voronoi diagrams for objects in order, the processing time for generating Voronoi diagrams might become too large when the number of objects is increased. In this paper, we propose a generation method of the Voronoi diagram by parallelizing the generation of Voronoi regions using a contact zone. Our proposed method can reduce the processing time of generating the Voronoi diagram by generating Voronoi regions in parallel based on the number of targets. Our evaluation confirmed that the processing time under the proposed method was reduced about 15.9% more than conventional methods that are not parallelized.

Keywords: Contact zone, Processing time, Voronoi diagram

1. Introduction

Due to the recent popularization of the Geographic Information System (GIS) [1], spatial network environments are receiving great attention because they can display the changes of spatial axes on mobile phones. In spatial network environments, since a query object that seeks location information selects several candidate target objects based on the search conditions, we often use a k-nearest neighbor (kNN) search, which seeks several target objects near the query object. However, since a kNN search needs to locate the kNN by calculating the distance from the query to all the objects, the computational complexity might become too large, depending on the number of objects. To reduce this computational complexity in a kNN search, many researchers have proposed a search method that divides regions using a Voronoi diagram [2, 3, 4] and accesses the spatial query using data structure [5, 6, 7].

In a search method that divides the regions using a Voronoi diagram, when there are several objects in a spatial network, the Voronoi diagram is divided into several Voronoi regions that are generated by the target object that is the closest to each object. In this method, the computational complexity is reduced for generating Voronoi regions for each object based on the positional relationships between objects. In addition, a generation method of the Voronoi diagram was proposed that reduces the processing time using the contact zone [8, 9]. However, since this method generates a Voronoi diagram for objects in order, the processing time for generating Voronoi diagrams becomes large when the number of objects is increased.

In this paper, we propose a generation method of Voronoi diagrams to reduce the processing time. In our proposed method, the processing time can be reduced by parallelizing the generation of Voronoi regions using the contact zone.

Our major contributions are:

- We describe conventional methods in generating the Voronoi diagram. In addition, we analyze the cause that increases the computational complexity in generating Voronoi diagram.
- We propose a efficient method reduces the processing time by setting the number of targets and generating Voronoi regions in parallel using a contact zone without an initial region.
- We confirm that the processing time under the proposed method is reduced more than the conventional method that is not parallelized.

The rest of the paper is organized as follows. We explain the basic Voronoi diagram in Section 2. In Section 3, we describe conventional generating methods of Voronoi diagram. We explain the conventional generation method of Voronoi diagrams using a contact zone in Section 4. Related works are introduced in Section 5. In Section 6, we describe the generation process of Voronoi regions in parallel. Our proposed method is explained in Section 7. We evaluate our proposed system in Section 8, and conclude our paper in Section 9.

2. Voronoi Diagram

2.1. *How to Generate Voronoi Diagram*

When a spatial network has several objects, the Voronoi diagram is divided into several Voronoi regions that are generated by the target objects that are the closest to each object. By generating a Voronoi diagram, the query reduces the processing time to search for its kNN. We defined the sides and vertices that compose the Voronoi region as a Voronoi side and a Voronoi vertex, respectively.

An example of generating a Voronoi diagram is shown in Figure 1. This procedure of generating a Voronoi diagram is called a simple method. When the spatial network has objects $p_0, \cdots, p_7$, a Voronoi diagram is generated by dividing it into multiple Voronoi regions based on the perpendicular bisector obtained by any two objects. When query q has a kNN search based on the generated Voronoi diagram, first randomly it selects an object. If q selects p_0, it selects p_7, which is the closest to itself for several objects in the Voronoi region adjacent

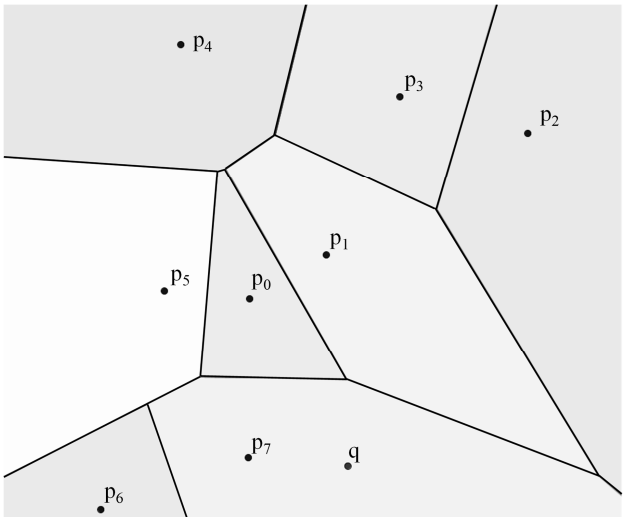

Fig. 1. Example of generating Voronoi diagram.

to p_0. Next, since q is inside the Voronoi region of p_7, the closest object to q becomes p_7, and our search for kNN is finished.

2.2. *Procedure for Generating Voronoi Diagram*

We can generate the Voronoi diagram based on a Voronoi region that is generated by drawing a perpendicular bisector between the target and each object. When the spatial network has objects $p_0, \cdots, p_7$, an example of generating Voronoi regions for p_0 is shown in Figure 2. First, a vertical bisector is drawn between p_0 and objects $p_1, \cdots, p_7$. Next, the region of the polygon, which is the common part of the half plane including p_0, is set as the Voronoi region. Here, the Voronoi region of p_0 is a square region surrounded by four perpendicular bisects with p_0 and p_1, p_4, p_5, and p_7.

When we make the Voronoi regions that are generated by drawing the perpendicular bisectors between the target for all objects, the computational complexity for generating a Voronoi diagram is proportional to the cube of the number of objects in a spatial network. Therefore, we need to propose a generating method of a Voronoi diagram to reduce this computational complexity.

3. Generating Methods of Voronoi Diagram

3.1. *Outline*

In spatial networks, researchers have proposed several generating methods. In this section, we explain the 2^* farthest vertex method [8], the incremental method [10], and the divide-and-conquer method [11].

3.2. 2^* *farthest vertex method*

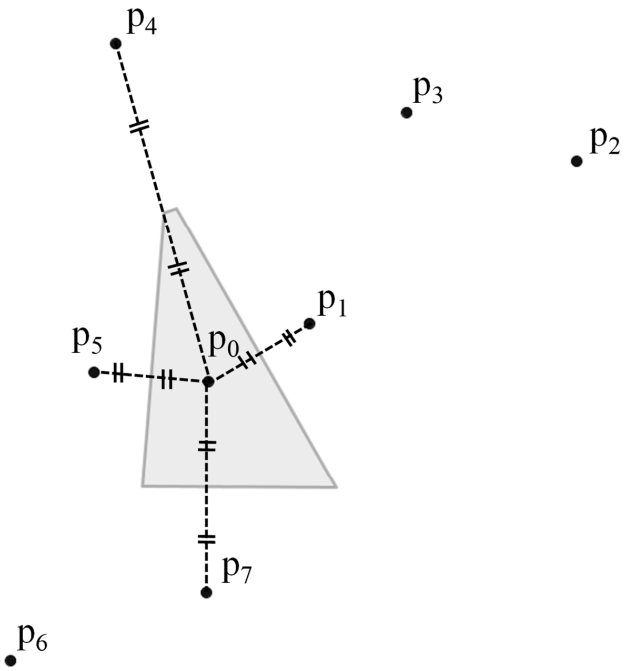

Fig. 2. Example of generating Voronoi region under simple method.

The 2* farthest vertex method [8] first draws a vertical bisector between the target and the object that is close to the target in the space network. When an initial region can be generated, the 2* farthest vertex method sets the vertices constituting the initial region and selects one vertex that is the furthest from the target. On the other hand, the Voronoi region is generated by the same procedure as the simple method.

Next, the 2* farthest vertex method generates a circle whose center is the position of the target, and the radius is twice the distance between the target and the selected vertex. A vertical bisector is drawn between the object inside the circle and the target, and the Voronoi region is updated.

An example of generating Voronoi regions using the 2* farthest vertex method is shown in Figure 3. The target is p_0, and there are seven objects $p_1, \cdots, p_7$ in the spatial network. First, three perpendicular bisectors with p_0 are drawn in the order of p_5, p_1, and p_7 that are close to target p_0, and a triangle is generated, which is the initial region of p_0. Next, the 2* farthest vertex method generates a circle whose center is p_0, and its radius is twice the distance between p_0 and selected vertex v. A vertical bisector is drawn between p_3 that is the shortest from the p_0 inside the circle and p_0, and the Voronoi region is updated. On the other hand, we excluded p_2 that exists outside the circle since it is not affected by generating the Voronoi diagram. Therefore, the 2* farthest vertex method reduces the amount of calculation for generating a Voronoi diagram more than the simple method.

3.3. *Incremental Method*

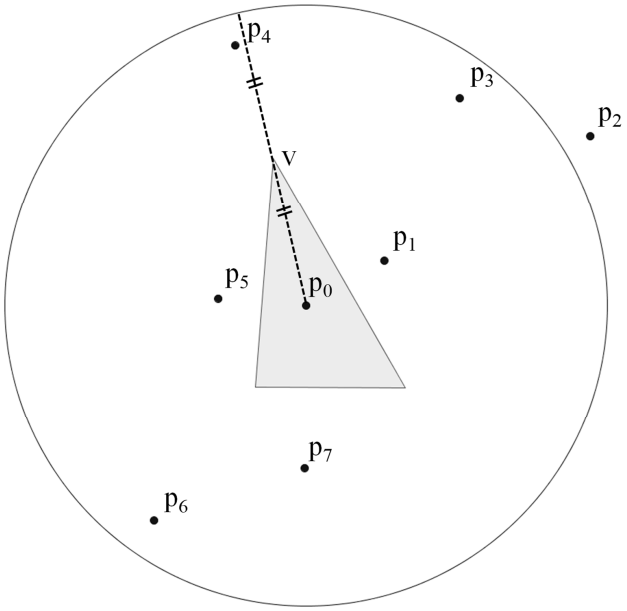

Fig. 3. Example of generating Voronoi region under 2* farthest vertex method.

The incremental method [10] first selects a certain number from a plurality of objects in a spatial network. Next, a Voronoi diagram is generated using the simple method for its selected objects. After the Voronoi regions are sequentially updated using the remaining unselected objects, the Voronoi diagram for all the objects is generated.

An example of generating Voronoi regions under the incremental method is shown in Figure 4. A Voronoi diagram is generated for its selected four objects p_1, p_2, p_5, and p_7. After the Voronoi regions are sequentially updated using the remaining unselected four objects p_0, p_3, p_4, and p_6, the Voronoi diagram for all the objects is generated. When p_0 is selected, a vertical bisector is drawn between p_0 and p_5, and the Voronoi region is updated. Finally, a Voronoi diagram is generated for p_3, p_4, and p_6. Since all Voronoi sides with p_2 do not intersect three perpendicular bisectors with p_0 are drawn in the order of p_5, p_1, and p_7, they do not affect updating the Voronoi region. Therefore, the incremental method reduces the amount of calculation for generating a Voronoi diagram more than the simple method.

3.4. *Divide-and-conquer Method*

In the divide-and-conquer method [11], first, a Voronoi region is divided by drawing an arbitrary number of straight lines perpendicular to the x axis in a spatial network, and a Voronoi diagram is generated by applying the incremental method to each divided region. Next, Voronoi diagrams in the spatial network are generated by updating them based on the several Voronoi diagrams generated in each region.

An example of generating a Voronoi diagram in the divide-and-conquer method is shown in Figure 5. When there are seven objects $p_1, \cdots, p_7$ in a spatial network, the divide-and-conquer method draws a straight line perpendicular to the x axis and divides the space into

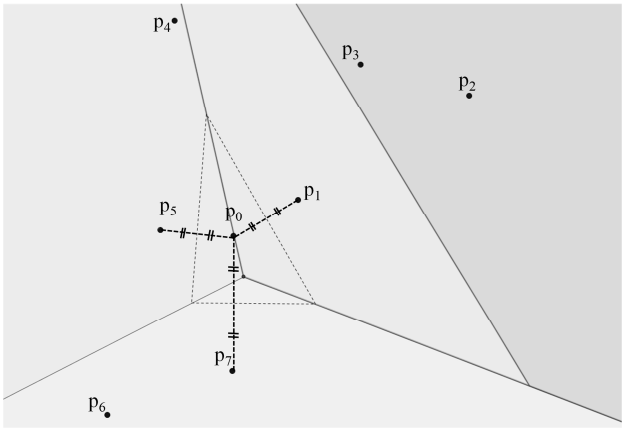

Fig. 4. Example of generating Voronoi region under incremental method.

two.

In the divide-and-conquer method, a Voronoi diagram is generated by applying the incremental method for both divided regions. Next, it is updated by combining both types of them generated in each region. The computational complexity of the divide-and-conquer method is $\mathcal{O}(n \log n)$ in both the worst and average cases.

4. Contact Zone

4.1. *Outline*

We explain the generating method of Voronoi diagrams to reduce the computational complexity using contact zone (CZ) [8, 9]. A CZ is a circle whose radius is the distance from the point that was initially set to the target. The method of setting the center point of the circle differs whether to consider a polygonal region (initial region) that is composed of the perpendicular bisectors drawn between the target object and other objects.

In the next subsections, we explain our procedure for generating a Voronoi diagram using CZ for cases where the initial region is considered and also not considered. The computational complexity in the generation method using CZ is $O(n^3)$ in the worst case.

4.2. *Generating Method of Voronoi Diagram with Initial Region*

First, similar to the simple method, a perpendicular bisector is drawn between objects on the spatial network and targets that are near the target. When an initial region can be generated with several perpendicular bisectors, the CZs whose centers are the vertex of the polygon with the initial region and whose radii are the distance from the vertex to the target are generated for each vertex. Next, a perpendicular bisector is drawn between the target and each object inside these CZs. Since the objects, which are not inside CZ, do not affect the generation of the Voronoi region, they can be excluded from the calculation for generating the Voronoi diagram. On the other hand, if the initial region can not be generated, a Voronoi region is

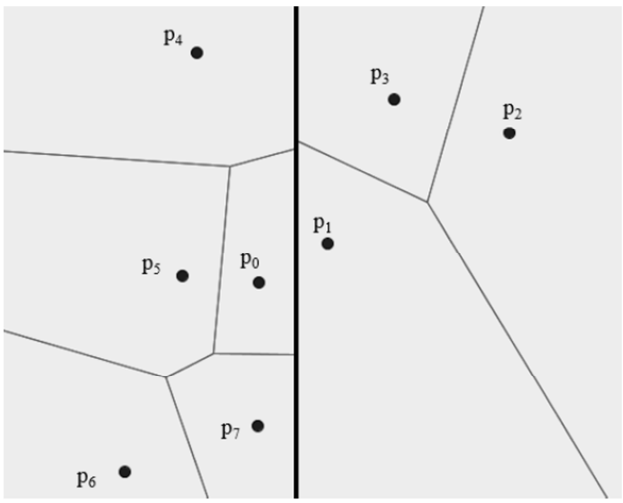

Fig. 5. Example of generating Voronoi region under divide-and-conquer method.

created by the same procedure as in the simple method.

In Figure 6, we explain the example that generates the Voronoi regions using CZ with the initial region. In a spatial network, the target is p_0 and seven objects $p_1, \cdots, p_7$. First, a perpendicular bisector with p_0 is drawn in order of the objects that are closer to p_0. In Figure 6, a triangle composed of three perpendicular bisectors drawn from p_0 to p_5 and p_1, and p_7 is set as an initial region. Next, we generated three CZs whose centers are each vertex of the triangle in the initial region, and whose radii are the distance from each vertex to p_0. Since objects p_2, p_3, and p_6 are outside of the three CZs, they are excluded. Therefore, this method can generate the Voronoi regions for p_0.

4.3. *Generating Method of Voronoi Diagram without Initial Region*

We explain the generating method for a Voronoi diagram using CZ without an initial region. First, the perpendicular bisectors are drawn in order of objects for the target close to each object. When two of the perpendicular bisectors can generate intersection point C, a region, which includes C among two regions divided by perpendicular lines drawn from the target to two perpendicular bisectors, is set as a Pruning Region (PR). Next, a circle CZ is generated centered on C with a radius as the distance from the target to C. The Voronoi region is generated by drawing a vertical bisector between the target and the object in the area that satisfies outside of PR or inside of PR and inside of CZ. On the other hand, objects within an area satisfying inside of PR and outside of CZ are excluded because they do not affect the generation of the Voronoi region.

In Figure 7, we explain the example that generates the Voronoi region using PR and CZ without the initial region. In a spatial network, the target is p_0 and seven objects $p_1, \cdots, p_7$. First, two perpendicular bisectors between p_0 and p_1 and between p_0 and p_5 are drawn in order of objects for p_0 close to each object. When two of the perpendicular bisectors can generate intersection point C, a region, which includes C among two regions divided by

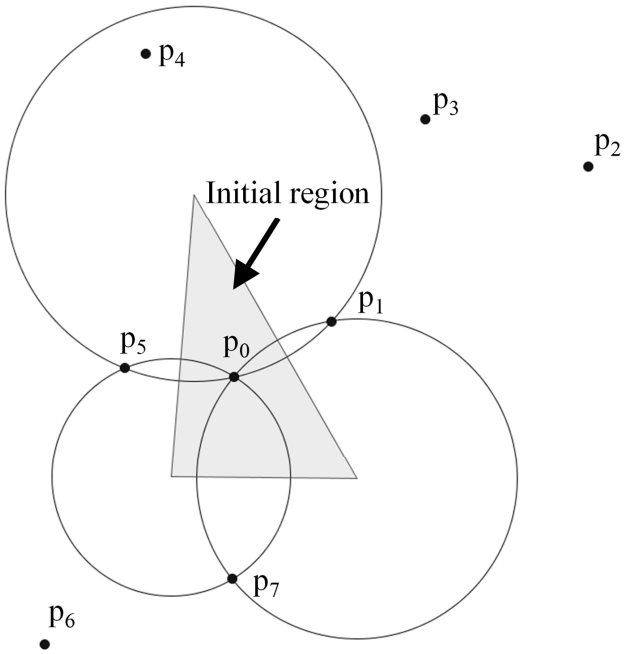

Fig. 6. Example of generating Voronoi region under method using CZ with initial region.

perpendicular lines drawn from p_1 to two perpendicular bisectors, is set as a PR. Next, two CZs centered on the C whose radius is the distance from C to p_1 and p_5 are generated. Since p_4 is inside the region of inside of PR and inside of CZ, the Voronoi region is updated by drawing a perpendicular bisector between p_0 and p_4. In addition, p_7 is selected that has the shortest distance from the target among the objects outside the PR. A perpendicular bisector is drawn between the target and p_7, and the Voronoi region is updated. Since objects p_2 and p_3 are inside of PR and outside of CZ, they are excluded. Therefore, this method reduces the calculation amount for generating a Voronoi diagram more than the simple method.

5. Related Works

5.1. *Searching Method of Nearest Neighbor using Voronoi Diagram*

The Voronoi diagram can obtain the nearest object using the spatial network represented by the Delaunay diagram [12] based on the adjacency relation of the Voronoi region. In the nearest neighbor search using Voronoi diagrams, the amount of calculation required for this search changes greatly according to the distance from one candidate object randomly selected to the query. For example, when a query takes as candidates the objects closer to itself, since the number of calculating the distance between the objects decreases, the amount of calculation required for the search also decreases.

5.2. *Method for Generating Highest Voronoi Diagram*

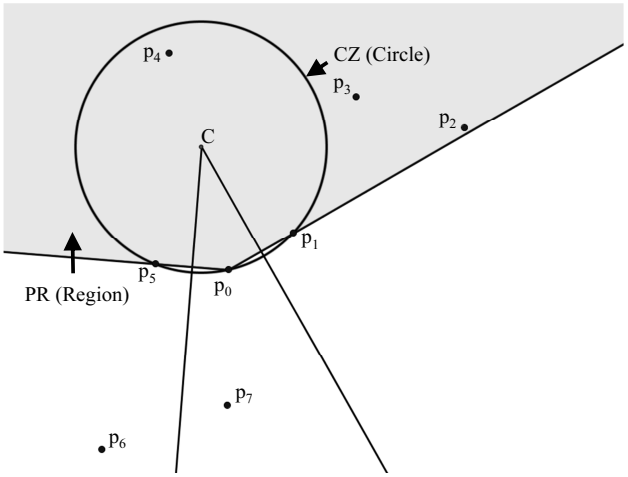

Fig. 7. Example of generating Voronoi region under method using CZ without initial region.

Highest order Voronoi diagram (HSVD) is a Voronoi diagram where every Voronoi cells have the sequence of distance order to all generator points. A searching method based on a single pre-computed HSVD has advantage in supporting multiple variations of region-based nearest neighbor problems. However, since the construction cost to makes pre-computation of HSVD is very high, this method is limited to a very small number of facility points.

By solving this problem, Christianto et. al. proposed a searching method [13] to improve the processing method of region-based Nearest Neighbor. In the experiment, their method shows that HSVD will boost the query performance for most of the common Nearest Neighbor problems in spatial queries. Therefore, their proposed method can change how the region-based Nearest Neighbor is processed.

6. Parallelizing Generation Process of Voronoi Regions

As explained in Section 4, in conventional generating methods of Voronoi diagrams, the processing time is lengthened as the number of objects is increased. In this paper, we consider the generation process of Voronoi regions in parallel for several targets.

First, we consider whether parallel processing for generating the Voronoi regions is possible. In the simple method, in the processing method using CZ with the initial region, in that using CZ without the initial region, and in the 2^* farthest vertex method [8], since the algorithm for generating the Voronoi regions is independent of each target, these processes can be parallelized. In addition, the divide-and-conquer method can be parallelized for each divided region [11]. On the other hand, the incremental method can not be parallelized to sequentially update the Voronoi diagram.

Next, we compared the computational complexity of four conventional methods that can parallelize the process for generating Voronoi diagrams. Since their parallelization process is independent for each target, we can reduce the processing time by selecting a method with low computational complexity. In this case, the computational complexity in the processing method using CZ without the initial region is the lowest. Therefore, we use it without the

initial region to parallelize the process for generating a Voronoi diagram.

7. Proposed Method

7.1. *Parallel Calculation using Multi-threading*

We propose a method that reduces the processing time in a Voronoi diagram by parallelizing the process for generating the Voronoi region of each target. Our proposed method reduces the processing time by setting the number of target objects (parallel number) for generating the Voronoi regions in parallel based on a processing method using CZ without an initial region.

A parallel calculation method using MapReduce [14] must construct an environment with multiple computers. Therefore, we implement parallel calculation using multi-threading on a single computer.

7.2. *Procedure of Generating Voronoi Region*

We show the procedure that generates the Voronoi region for each target. The procedure in the proposed method is the same as that using CZ without an initial region:

1. Select the unchosen object that is the closest to the target that is selected.
2. Draw a perpendicular bisector between the target and the object that is selected in step 1.
3. If an intersection can be created with the perpendicular bisector in step 2, go to step 4. Otherwise, go to step 1.
4. Generate CZ and PR based on the intersection generated in step 2.
5. If there is an object outside of PR or inside PR and CZ, select one that is the closest to the target, and go to step 2. Otherwise, end the process.

7.3. *Procedure of Generating Voronoi Diagram*

In this subsection, we show the procedure that generates a Voronoi diagram using the Voronoi region.

1. In a spatial network, as the target randomly select the number of objects that is the same as the parallel number.
2. Based on the procedure for generating Voronoi regions (Subsection 7.2), generate a Voronoi region in parallel for each target selected in step 1. In addition, wait until the Voronoi regions have ended for all the selected targets.
3. Repeat steps 1 to 3 until all objects generate Voronoi regions as targets.

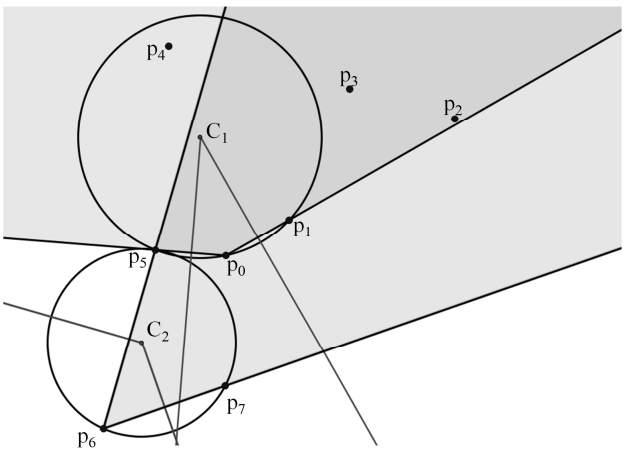

Fig. 8. Example of generating Voronoi region under proposed method.

7.4. *Implementation of Proposed Method*

Based on the procedure of parallel processing in the proposed method (Subsections 7.2 and 7.3), an example of generating a Voronoi region in parallel is shown when the parallel number is two (Figure 8). There are eight objects p_0, $\cdots$, p_7 in a spatial network. The proposed method selects p_0 and p_6 as targets, and simultaneously generates Voronoi regions. In the proposed method, Voronoi regions are generated using CZ without an initial region. p_5 and p_1 are selected in order of decreasing distance from p_0, and two perpendicular bisectors between these objects and p_0 are drawn. The intersection point of two perpendicular bisectors between p_0 and p_5 and between p_0 and p_1 is set to C_1, and CZ and PR are generated.

Similarly, p_7 and p_5 are selected in order of decreasing distance from p_6, and two perpendicular bisectors are drawn between these objects and p_6. The intersection points of two perpendicular bisectors between p_6 and p_7, and between p_6 and p_5 are set to C_2, and CZ and PR are generated.

In Figure 8, PR is shown in a gray region. After generating the Voronoi regions of p_0 and p_6, two targets are chosen from the unselected objects, and the Voronoi region is updated. Finally, a Voronoi diagram is generated by repeating the above processing procedure until there are no unselected objects.

8. Evaluation

8.1. *Outline*

To confirm the availability of our proposed method, we evaluated it in a simulation environment. We implemented the programs of the proposed and conventional methods in C++. The performance of the computers in our evaluation is shown in Table 1.

8.2. *Evaluation Environment*

Table 1. Computer spec.

Item	Explanation
OS	Ubuntu 14.04 LTS
CPU	Intel Core i3-2367M Processor
Number of CPU cores	2
Number of threads	4
Memory	4 GBytes
HDD	120 GBytes

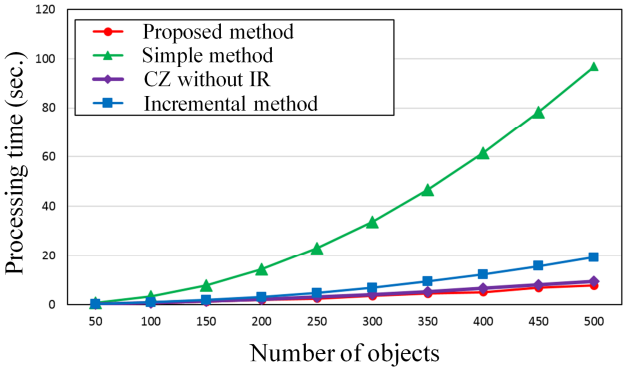

Fig. 9. Processing time for generating Voronoi diagram.

The evaluation environment of our proposed method is shown below.

1. The distance between objects is set by the Euclidean distance.
2. Based on conventional research [9], we set ten amounts of objects: 50, 100, 150, 200, 250, 300, 350, 400, 450, and 500.
3. The size of the spatial network is a two-dimensional region: 1,000 by 1,000.
4. There are five parallel numbers: four threads used in the evaluation and one for switching in the process.

Proposed method indicates the proposed method. Simple method indicates the simple method. CZ without IR indicates the conventional method for using a contact zone without the initial region. Incremental method indicates the incremental method.

8.3. *Number of Objects and Processing Time*

We evaluated the processing time under several number of objects. The result is shown in Figure 9. The horizontal axis is the number of objects in a spatial network, and the vertical axis is the average processing time for generating a Voronoi diagram. We used our proposed method, the simple method, the method using CZ without the initial region, and the incremental method. In our evaluation, we measured the processing time ten times for generating a Voronoi diagram and calculated the average value.

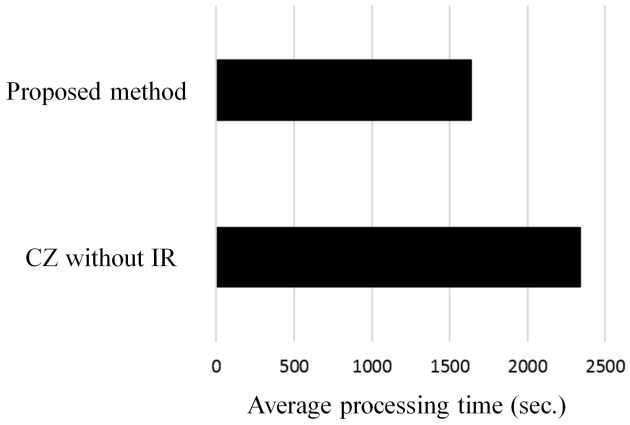

Fig. 10. Average processing time for generating Voronoi diagram (number of objects: 10,000).

In Figure 9, the processing time for generating the Voronoi diagram was reduced more than the other methods. In our proposed method, the processing time was reduced by parallelizing the generation of the Voronoi regions using CZ without an initial region. When the number of objects was 100, the processing time under the proposed method was 0.562 sec. and 0.669 sec. under the conventional method using CZ without an initial region. The processing time under the proposed method was reduced by 15.9% more than the conventional method.

Next, we evaluated the processing time for generating a Voronoi diagram under a different number of objects (Figure 10). The horizontal axis is the average processing time for generating a Voronoi diagram. We used Proposed and CZ without IR methods. In our evaluation, we measured the processing time ten times for generating a Voronoi diagram when the number of objects was 10,000 and calculated the average value.

In Figure 10, compared with the case where the number of objects was 500 or fewer, the effect of parallelization by the proposed method increased. When the number of objects was 10,000, the processing time under the proposed method was 1635.2 sec. and 2338.5 sec. under the conventional method using CZ without an initial region. Therefore, the processing time under the proposed method was reduced 30.0% more than the conventional method.

8.4. *Parallel Number and Average Processing Time*

We calculated the average processing time for generating a Voronoi diagram under a parallel number (Figure 11). The horizontal axis is a parallel number, and the vertical axis is the average processing time for generating it. In our evaluation, when the number of objects was 500, we calculated the average values ten times.

In Figure 11, the average processing time was reduced by increasing the parallel number. In our evaluation, the computer generated a Voronoi diagram using four threads. Therefore, when the parallel number was 4 or less, the processing time was reduced by increasing of the parallel number. When the parallel number was 5 or more, since the processing of each thread did not end at the same time, the waiting time described in Subsection was reduced. When the parallel number is 5 or more, the processing was reduced by increasing it.

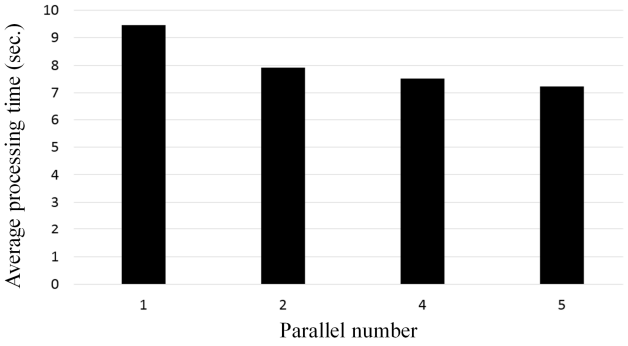

Fig. 11. Average processing time for generating Voronoi diagram.

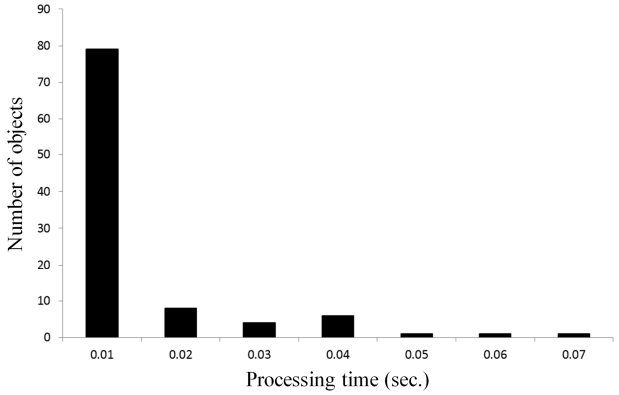

Fig. 12. Distribution of processing time for generating Voronoi region.

8.5. *Distribution of Processing Time for Objects*

In the conventional method using CZ without an initial region, when the Voronoi region has a side that reaches infinity, the processing time is lengthened. When the number of objects in a spatial network is 100, we evaluated the distribution of the processing time for generating a Voronoi diagram. The result is shown in Figure 12. The horizontal axis is the average processing time, and the vertical axis is the number of objects.

In Figure 12, the proposed method has bias in the number of objects based on the processing time for generating the Voronoi region. In parallel processing, when a thread occurs that increases the processing time to generate the Voronoi region, waiting time also occurs because the processing of other threads waits until the processing of this thread is completed. Therefore, the efficiency of parallel processing decreases.

Next, we confirmed the waiting time of each thread in a parallel process. When the proposed methods used five threads, we evaluated the processing time of the Voronoi diagram of each thread (Figure 13). In the horizontal axis, thread 1, $\cdots$, thread 5 is the processing time of the Voronoi diagram. Total is the processing time until all the threads have been processed. The vertical axis is all of the threads. The number of objects on the spatial networks was

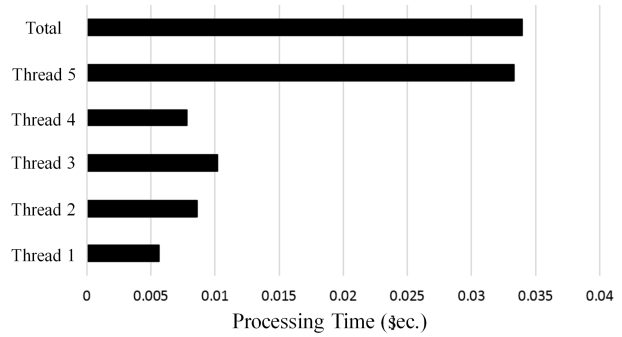

Fig. 13. Processing time of each thread for generating Voronoi region.

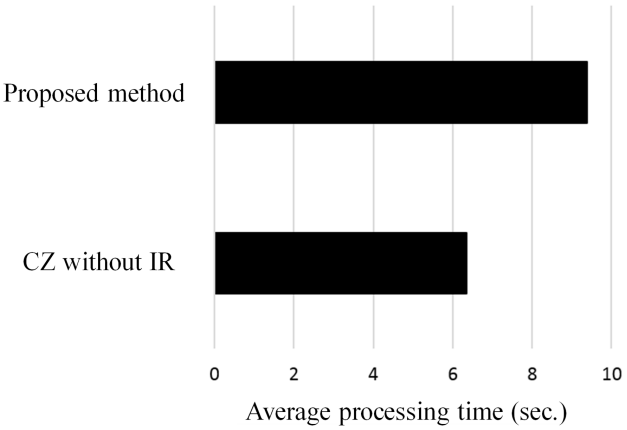

Fig. 14. Average processing time for generating Voronoi region.

100.

In Figure 13, the processing time of thread 5 was lengthened. In addition, since the processing time using all of the threads was 0.339 sec. and 0.333 sec. under thread 5, the processing time in thread 5 was more lengthened than the other threads; waiting time occurred with threads 1 to 4.

8.6. *Processing Time of Generating Voronoi Region Considering Multi-threads*

In evaluation environment, we use the simulation computer in parallel with six threads, such as one main thread for generating other threads, and five sub-threads for generating the Voronoi regions.

We evaluate the average processing time for generating the Voronoi region. The result is shown in Figure 14. The vertical axis is generating methods such as the proposed method and the method using CZ without the initial region. The horizontal axis is the average processing time. The number of objects in spatial network is 100.

In Figure 14, the processing time under the proposed method is more lengthened than

the generating method using CZ without the initial region. In generating the Voronoi region, the average processing time for generating the Voronoi region is also lengthened due to the processing time occurs by generation of the thread, destruction of it, and switching with two threads. The average processing time under the proposed method is 9.37 msec., 6.36 msec. under the method using CZ without initial region.

However, the effect for increasing of processing time in Section is small. Therefore, the total processing time under the proposed method is more reduced totally than conventional methods.

9. Conclusion

In this paper, we proposed a method to reduce the processing time of Voronoi diagrams by parallelizing the process for generating the Voronoi region of each target. The proposed method reduces the processing time by setting the number of targets and generating Voronoi regions in parallel using a contact zone without an initial region. In our evaluation, when the number of objects was 100 and the parallel number was five, we confirmed that the processing time under the proposed method was reduced by about 15.9% more than the conventional method that is not parallelized.

In the future, we will evaluate increasing the parallel number and using MapReduce.

Acknowledgements

This work was partially supported by the Wesco Scientific Promotion Foundation. In addition, this work was supported in part by the Telecommunications Advancement Foundation.

References

1. Ministry of Internal Affairs and Communications, Japan (2018), *WHITE PAPER on Information and Communications in Japan*, http://www.soumu.go.jp/johotsusintokei/whitepaper/eng/WP2018/2018-index.html.
2. A. Okabe, B. Boots, K. Sugihara, S. Chiu (2000), *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, Wiley Series in Probability and Statistics (England).
3. S. Yang, M.A. Cheema, X. Lin, W. Wang (2015), *Reverse k Nearest Neighbors Query Processing: Experiments and Analysis*, Proc. VLDB Endowment, Vol.8, Issue 5, pp.605-616.
4. B. Costello (2015), *Calculating Voronoi Diagrams Using Simple Chemical Reactions*, Proc. Parallel Processing Letters, Vol.25, Issue 1, pp.1-26.
5. A. Guttman (1984), *R-trees: A Dynamic Index Structure for Spatial Searching*, Proc. 1984 ACM SIGMOD Int. Conf. on Management of Data, Vol.14, Issue 2, pp.47-57.
6. A. Gionis, P. Indyk, R. Motwani (1999), *Similarity Search in High Dimensions via Hashing*, Proc. 25th Int. Conf. on Very Large Data Bases, pp.518-529.
7. P. Ciaccia, M. Patella, P. Zezula (1997), *M-tree: An Efficient Access Method for Similarity Search in Metric Spaces*, Proc. 23rd Int. Conf. on Very Large Data Bases, pp.426-435.
8. W. Wu, F. Yang, C.Y. Chan, K.L. Tan (2008), *FINCH: Evaluating Reverse k-Nearest Neighbor Queries on Locational Data*, Proc. VLDB Endowment, Vol.1, Issue 1, pp.1056-1067.
9. K.M. Adhinugraha, D. Taniar, M. Indrawan, *Finding Reverse Nearest Neighbors by Region*, Concurrency and Computation, Practice and Experience, Vol.26, Issue 5, pp.1142-1156.
10. P.J. Green, R. Sibson (1978), *Computing Dirichlet Tessellations in the Plane*, Proc. Computer Journal, Vol.21, Issue 2, pp.168-173.

11. O. Aichholzera, W. Aignera, F. Aurenhammer, T. Hack, B. J 端 ttler, E. Pilgerstorfer, M. Rabl (2010), *Divide-and-conquer for Voronoi Diagrams Revisited*, Computational Geometry, Vol.43, Issue 8, pp.688-699.
12. Okabe, A., Boots, B., Sugihara, K. and Chiu, S. : Spatial Tessellations Concepts and Applications of Voronoi Diagrams, 2nd Edition, John Wiley Sons, Chechester, England (2000).
13. D.S. Christianto, K.M. Adhinugraha, A. Herdiani, S. Alamri (2017), *Highest Order Voronoi Diagram Optimization* Proc. 5th International Conference on Information and Communication Technology (ICoIC7), pp.1-7.
14. J. Dean, S. Ghemawat (2008), *MapReduce: Simpli fied Data Processing on Large Clusters*, Proc. Communications of the ACM, Vol.51, Issue 2, pp.107-113.