Constructing Higher-Order Voronoi Diagrams in Parallel

Henning Meyerhenke*

Abstract

We use Lee's sequential algorithm [17] to create the first parallel algorithm which constructs the order-k Voronoi diagram of a planar point set. The algorithm is developed and analyzed within two parallel models, the fine-grained PRAM and the coarse-grained CGM. Its applications include k-nearest neighbor searches in a parallel context, which are important for many applications in computational geometry. The fine-grained algorithm requires $O(k\,log^2\,n)$ time and $O(k^2 n\,log\,n)$ work on a CREW-PRAM, whereas the coarse-grained version requires the ordinary Voronoi diagram as input and then takes $O(\frac{k^2(n-k)\,log\,n}{p})$ running time and $O(k)$ communication rounds on a CGM with $O(\frac{k^2(n-k)}{p})$ local memory per processor.

1 Introduction

The Voronoi diagram is one of the most popular geometric structures studied in the field of computational geometry due to its rich algorithmic application in placement and motion planning, triangulations, connectivity graphs and closest-site problems [5]. A generalization of ordinary Voronoi diagrams are those of order k, whose regions are the locus of points with the same k nearest neighbors. Given an order-k Voronoi diagram, one can find the k-nearest point site of a query point efficiently, which is very important for generalized closest-site problems and for the closely related higher order Delaunay triangulations [15].

Since these applications from geometry and geographic information systems often benefit from parallel computing, we develop a simple parallel algorithm to construct all higher order Voronoi diagrams of some given order k and below. After the description of the sequential algorithm the parallel algorithm is developed and analyzed both in the fine-grained PRAM model (section 2) and in the coarse-grained CGM model (section 3), which is more relevant in practice.

In an ordinary Voronoi diagram of a planar point set S of size n each Voronoi region is induced by exactly one point site of S, since it is the locus of points with the same nearest neighbor in S. The concept of order k extends this perception by making each Voronoi region the locus of points which have the same k nearest neighbors (thus, an ordinary Voronoi diagram has order one). An example of the order-2 diagram of a planar point set is depicted in figure 1.

Previous works on sequential algorithms for constructing higher order Voronoi diagrams include the books of Preparata and Shamos [18] and Edelsbrunner [13]. The latter points out the duality between k-levels in arrangements and higher order Voronoi diagrams, an idea which is used in some of the sequential algorithms developed for the posed problem by Agarwal et al. [1], Aurenhammer [4], Aurenhammer and Schwarzkopf [6], Boissonnat et al. [7], Chan [8], Chazelle and Edelsbrunner [9], Ramos [19], and Lee [17].

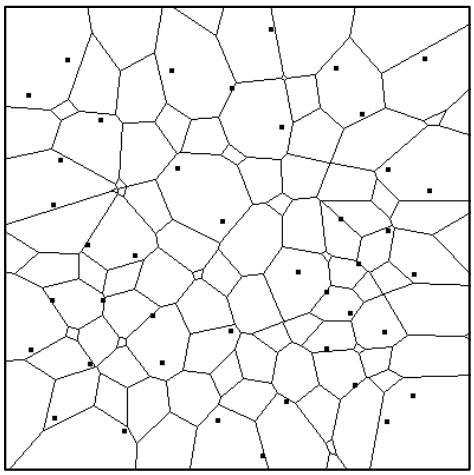

Figure 1: Voronoi diagram of order 2 of a planar point set, generated with an applet written by B. Schaudt [20].

D. T. Lee [17] describes a sequential algorithm for computing the family of all these diagrams of order $\leq k$ in $O(k^2 n\,log\,n)$ time. His method is iterative as it modifies $V_i(S)$, the Voronoi diagram of order i, in order to obtain the diagram of order $i+1$. Its space complexity for each iteration is $O(i^2(n-i))$, since the order-i diagram has $O(i(n-i))$ regions and for each of which i inducing points are stored. Meanwhile, the algorithm's time complexity has been improved to $O(n\,log\,n + nk^2)$ by Aggarwal et al. [2], whose algorithm can delete one point site from a Voronoi diagram and recompute the resulting one in linear time.

The iterative approach of Lee's algorithm (transforming the diagram of order i into the diagram of

*Department of Computer Science, University of Paderborn, henningm@upb.de

order $i+1$) is mainly based on the following idea: Each Voronoi region r of order i, which is induced by some $H \subset S$ with $|H| = i$, is partitioned into subregions which are the respective locus of points with the same $i+1$ neighbors in S. Since all points of r share the same i nearest neighbors (namely the points in H), what remains to be done is to partition r into subregions according to their nearest neighbor in $S \backslash H$. This partitioning procedure works by merging $V_1(S \backslash H)$, the ordinary Voronoi diagram of the remaining $n-i+1$ points, appropriately with r (cf. Lee [17] for details).

2 The PRAM algorithm

As Lee has shown [17, p. 482], two adjacent regions of a Voronoi diagram of order i share $i-1$ of their i nearest neighbors. When partitioning such a region r according to $V_1(S \backslash H)$, it is split into regions w.r.t. to the $(i+1)$-nearest neighbors of r (i.e. the (possibly) different $(i+1)$-nearest neighbors of different points in r). The key observation is that only few additional information is required to do this.

Lemma 1 *The $i+1$-nearest neighbors of a region r of $V_i(S)$ all induce adjacent regions of r.*

Proof. Points on Voronoi edges of order i share the same $i+1$ nearest neighbors because they have not only one, but two i-nearest neighbors. If r is induced by the point set H, then its edges are induced by H and one additional point each. This point induces the adjacent region whose boundary the respective edge is part of. □

Corollary 2 *In order to partition a Voronoi region r of order i into subregions of order $i+1$, it is sufficient to know r's neighboring regions of order i (and their inducing points).*

This locality property is obviously useful for a parallel version of Lee's algorithm. Following its scheme, the problem mainly consists of partitioning each order-i region into subregions of order $i+1$ and merging equivalent subregions. To do this for some region r, one only needs to know the points which induce the regions adjacent to r.

For the PRAM algorithm we therefore compute for each order-i region r that part of $V_1(S \backslash H)$ (the ordinary Voronoi diagram of S without the points that induce r) which contributes to r. This can be done using the algorithm developed by Amato et al. [3], which requires $O(log^2\, n)$ time and $O(n \log n)$ work on a EREW-PRAM to compute the Voronoi diagram of n sites in the plane. After that, only some minor update operations have to be performed to complete one iteration and obtain the order-$(i+1)$ diagram.

More formally we state the algorithm as follows:

Input: Planar point set S^n, $p \leq n$ processors, k
Output: Order-k Voronoi diagram of S

1. Use Amato et al.'s algorithm [3] to compute the Voronoi diagram of S.
2. FOR $i := 1$ TO $k-1$ DO

   /* Transform the diagram of order i into one of order $i+1$ */

   (a) FOR EACH region r_j PARDO

      i. Partition r_j induced by $H_j \subset S$ into subregions according to $V_1(S \backslash H_j)$, which is computed by Amato et al.'s algorithm.

   (b) FOR EACH new region r_m not updated yet PARDO

      i. Update r_m by merging it with equivalent subregions and deleting old edges within the merged region.

Theorem 3 *The family of all order-($\leq k$) Voronoi diagrams can be computed on a CREW-PRAM using $O(k\, log^2\, n)$ time and $O(k^2 n \log n)$ work.*

Proof. Recall that an order-i Voronoi diagram has $O(i(n-i))$ regions. If an r_j of them has s_j edges, we need to compute the ordinary Voronoi diagram of s_j points. The total amount of work per outer iteration is therefore $\sum_{j=1}^{k(n-k)} s_j \log s_j = kn \log n$. (The parallel merging/deleting loop requires only constant time and a linear amount of work.) Since one region can have at most $O(k(n-k))$ edges, the algorithm takes $O(log^2(k(n-k)) = O(log^2\, n)$ time for each of the $k-1$ iterations. For this method, no concurrent write operations are necessary, but concurrent read operations are. □

Unfortunately, the work complexity does not match the improved sequential running time. For this, a parallel algorithm comparable to Aggarwal et al.'s sequential one [2] that deletes one site from an existing Voronoi diagram and can recompute the new one using a linear amount of work, would be necessary.

3 The CGM algorithm

We use now the locality property explained in the previous section to develop a coarse-grained parallel algorithm for the problem within the (slightly modified) CGM model [11]. The original CGM model consists of p processors connected by some arbitrary network, with each processor having a local memory of size $O(\frac{n}{p})$. Interprocessor communication is realized by message passing in communication rounds, during which no processor can send or receive more data items than fit into its local memory.

It was shown by Dehne et al. [11] that many collective communication primitives can be simulated on a CGM by a constant number of global sort operations. Moreover, Goodrich proved [14] that coarse-grained sorting can be performed optimally within a constant number of communication rounds if $\frac{n}{p} \geq p$.

For our algorithm we slightly modify the model by increasing the local memory bound to $O(\frac{k^2(n-k)}{p})$, which is the space complexity of the sequential algorithm divided by the number of processors.

The ordinary Voronoi diagram of the point set S is part of the input of our algorithm.[1] This diagram of order 1 must have the property to be *x-disjoint*, i.e. it is stored in such a way that S is distributed on the processors within disjoint intervals w.r.t. the x-coordinate. This enables a processor to determine efficiently which other processor stores a particular point site. (If the diagram is not x-disjoint, it can be made so via a constant number of global sort operations.)

The algorithm works as follows:

Input: $CGM(n,p)$ with $\frac{n}{p} \geq p$ and $O(\frac{k^2(n-k)}{p})$ local memory per processor; planar point set S of size n and its Voronoi diagram, which is distributed on the p processors in such a way that each processor stores $O(\frac{n}{p})$ Voronoi regions and edges; order k of the desired output.

Output: $V_k(S)$ distributed on the p processors such that every processor stores $O(\frac{k(n-k)}{p})$ Voronoi edges and regions.

1. If $V_1(S)$ is not x-disjoint, then it is rearranged by a constant number of global sorting steps to make it x-disjoint.

2. Every processor sends the locally stored point with the largest x-coordinate to all the other processors.

   With this information every processor determines by binary search for each *boundary region* (i.e. a local Voronoi region with at least one non-local neighbor region), which other processor stores its *non-local neighbor regions* (i.e. regions stored on other processors that share an edge with a locally stored region).

3. FOR $i := 1$ TO $k-1$ DO

   (a) Each processor sends necessary information about its boundary regions of order i to the processors which store their non-local neighbor regions (if present).

   (b) Using Lee's algorithm [17, p. 482] with the enhancement by Aggarwal et al. [2], each processor transforms its local part of $V_i(S)$ into the corresponding order-i Voronoi diagram.

   (c) Since some of the new Voronoi regions are computed on two processors but are to be used on only one of them, we proceed as follows: If i is even, a newly created duplicate region is only retained on the processor with the lower number, otherwise only on the processor with the higher number.

Lemma 4 *A processor creates at most $O(\frac{i(n-i)}{p})$ Voronoi regions of order $i+1$ during the transition from $V_i(S)$ to $V_{i+1}(S)$.*

Proof. An ordinary Voronoi diagram can be distributed on p processors such that each processor stores $O(\frac{n}{p})$ Voronoi regions and edges so that the claim is true for $i = 1$. Let us assume now inductively that it holds for arbitrary $i_0 \leq i$.

According to our assumption, a processor stores $O(\frac{i(n-i)}{p})$ Voronoi regions and edges of an order-i Voronoi diagram. These local regions can be seen as an independent sub-diagram, which can be influenced by at most $O(\frac{i(n-i)}{p})$ non-local Voronoi regions, since the local regions cannot have more neighbor regions than edges. The total number of all these regions is still $O(\frac{i(n-i)}{p})$, so that the order-$(i+1)$ sub-diagram which is created during the transition, has at most $O(\frac{(i+1)(n-(i+1))}{p}) = O(\frac{i(n-i)}{p})$ regions. □

Theorem 5 *Let a coarse-grained multicomputer $CGM(n,p)$ with $\frac{n}{p} \geq p$ and local memory size $O(k^2(n-k))$ be given and let the Voronoi diagram of a planar point set S be distributed on $CGM(n,p)$ such that each processor stores $O(\frac{n}{p})$ Voronoi regions and edges. Then one can compute the order-k Voronoi diagram of S with a time complexity of $O(\frac{k^2(n-k)\log n}{p})$ for local computation using $O(k)$ supersteps and communication rounds, during which every processor sends and receives at most $O(k^2(n-k))$ data items.*

Proof. Using corollary 2, we conclude that each processor can transform its locally stored Voronoi subdiagram of order i into the sub-diagram of order $i+1$ if it knows all neighbors of its locally stored regions. This is ensured by the communication in step 3a. Since all duplicate regions are eliminated in step 3c, the union of all sub-diagrams forms the Voronoi diagram of order $i+1$ after iteration i.

The communication complexity of the algorithm is clearly dominated by step 3a, during which at most

[1] The best deterministic CGM algorithm for computing the Voronoi diagram of arbitrary planar point sets is due to Diallo et al. [12] and requires $O(\frac{n \log n \log p}{p})$ local computation steps and $O(\log p)$ communication rounds. Moreover, two rather complicated randomized CGM algorithms exist, which both compute the planar Voronoi diagram with high probability in optimal time of $O(\frac{n \log n}{p})$ and with $O(1)$ communication rounds [10, 16].

$O(\frac{i(n-i)}{p})$ Voronoi regions are communicated. (Note that for each region the number of inducing points sent/received is not greater than the region's number of edges.) This can be accomplished by a total exchange operation. Hence, no more than $O(\frac{i(n-i)}{p})$ points are sent and received during one communication round.

Furthermore, the local time complexity is dominated by this communication step, too. It requires in total $\sum_{i=1}^{k-1} O(\frac{i(n-i)\log n}{p}) = O(\frac{k^2(n-k)\log n}{p})$ local computation. In contrast to this, the other steps require only $O(\frac{n \log n}{p})$ (step 1), $O(\frac{n \log p}{p})$ (step 2), and $\sum_{i=1}^{k-1} O(\frac{i(n-i)}{p}) = O(\frac{k^2(n-k)}{p})$ (both step 3b and step 3c), respectively.

Clearly, the local memory space bound of $O(\frac{k^2(n-k)}{p})$ is necessary, but never exceeded. □

4 Conclusion

In this paper we present and analyze a parallel algorithm for constructing higher order Voronoi diagrams for two parallel models, PRAM and CGM. For both models we use Lee's iterative idea [17] to transform the diagram of order i into one of order $i+1$. Although the algorithm is quite simple, it appears to be the first parallel one for the construction of higher order Voronoi diagrams.

Two possible improvements could be the object of further research: First, a parallel algorithm which deletes a point site from a Voronoi diagram and can recompute the new one using a linear amount of work. This would make our PRAM algorithm work-optimal when compared to Lee's sequential algorithm.

Secondly, the CGM algorithm is only efficient for small k. Although small k seem to be more relevant in practice, it would nevertheless be of interest to obtain a CGM algorithm which requires significantly less than $O(k)$ communication rounds.

References

[1] P. K. Agarwal, M. de Berg, J. Matoušek, O. Schwarzkopf. Constructing Levels in Arrangements and Higher Order Voronoi Diagrams. In *Proc. 10th Symp. on Comp. Geometry* (1994), pp. 67–75.

[2] A. Aggarwal, L. J. Guibas, J. Saxe, P. W. Shor. A Linear-Time Algorithm for Computing the Voronoi Diagram of a Convex Polygon. In *Discrete & Comput. Geometry* **4** (1989), pp. 591–604.

[3] N. M. Amato, M. T. Goodrich, E. A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th Annual IEEE Symp. on Foundations of Comp. Science* (1994), pp. 683–694.

[4] F. Aurenhammer. A New Duality Result Concerning Voronoi Diagrams. In *Discrete & Comput. Geometry* **5** (1990), pp. 243–254.

[5] F. Aurenhammer. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. In *ACM Comput. Surv.* **23** (1991), no. 3, pp. 345–405.

[6] F. Aurenhammer, O. Schwarzkopf. A Simple On-Line Randomized Incremental Algorithm for Computing Higher Order Voronoi Diagrams. In *Proc. 7th Symp. on Computational Geometry* (1991), pp. 142–151.

[7] J.-D. Boissonnat, O. Devillers, M. Teillaud. A Semi-Dynamic Construction of Higher Order Voronoi Diagrams and its Randomized Analysis. In *Algorithmica* **9** (1993), pp. 329–356.

[8] T. M. Chan. Random Sampling, Halfspace Range Reporting, and Construction of ($\leq k$)-Levels in Three Dimensions. In *SIAM J. Comput.* **30** (2000), no. 2, pp. 561–575.

[9] B. Chazelle, H. Edelsbrunner. An Improved Algorithm for Constructing kth-Order Voronoi Diagrams. In *IEEE Transactions on Computers* **C-36** (1987), no. 11, pp. 1349–1354.

[10] F. Dehne, X. Deng, P. Dymond, A. Fabri, A. A. Khokhar. A Randomized Parallel Three-Dimensional Convex Hull Algorithm for Coarse-Grained Multicomputers. In *Theory of Computing Systems* **30** (1997), no. 6, pp. 547–558.

[11] F. Dehne, A. Fabri, A. Rau-Chaplin. Scalable parallel computational geometry for coarse grained multicomputers. In *Int. J. Comput. Geometry* **6** (1996), no. 3, pp. 379–400.

[12] M. Diallo, A. Ferreira, A. Rau-Chaplin. A Note On Communication-Efficient Deterministic Parallel Algorithms for Planar Point Location and 2d Voronoi Diagram. In *Parallel Processing Letters* **11** (2001), no. 2-3, pp. 327–340.

[13] H. Edelsbrunner. Algorithms in Combinatorial Geometry. Springer-Verlag, 1987.

[14] M. T. Goodrich. Communication-Efficient Parallel Sorting. In *SIAM J. on Computing* **29** (1999), no. 2, pp. 416–432.

[15] J. Gudmundsson, M. Hammar, M. J. van Kreveld. Higher Order Delaunay Triangulations. In *Comp. Geometry - Theory and Applications* **23** (2002), no. 1, pp. 85–98.

[16] U. Kühn. Lokale Eigenschaften in der algorithmischen Geometrie mit Anwendungen in der Parallelverarbeitung. Dissertation Westfälische Wilhelms-Universität Münster, 1998.

[17] D. T. Lee. On k-nearest neighbor Voronoi diagrams in the plane. In *IEEE Transactions on Computers* **31** (1982), no. 6, pp. 478–487.

[18] F. P. Preparata, M. I. Shamos. Computational Geometry - An Introduction. Springer-Verlag New York, 1985.

[19] E. A. Ramos. On range reporting, ray shooting and k-level construction. In *Proc. 15th Symp. on Computational Geometry* (1999), pp. 390–399.

[20] B. Schaudt. Higher Order Voronoi Diagrams: A Java Applet. Website available at www.msi.umn.edu/~schaudt/voronoi/voronoi.html.