

A Theoretical Study of Parallel Voronoi Diagram

Rashid Bin Muhammad
Department of Computer Science
Kent State University
Kent, OH, U.S.A.
email: rmuhamma@cs.kent.edu

Abstract—In this paper, we concentrate on the problem of computing a Voronoi diagram using Hypercube model of computation. The main contribution of this work is the $O(\log^3 n)$ parallel algorithm for computing Voronoi diagram on the Euclidean plane. Our technique parallelizes the well-known seemingly inherent sequential technique of Shamos and Hoey, and makes use of a number of special properties of the dividing polygonal chain and the Batchers bitonic sort.

Keywords: Voronoi diagram, dividing chain, hypercube, and intersection set.

1 Introduction

The concept of Voronoi diagram has been applied in diverse fields of sciences such as anthropology, archeology, astronomy, biology, crystallography, geography, marketing, mathematics, physiology, physics, robotics, statistics, zoology, to name a few. For example, in astronomy, the concept is used to identify clusters of stars and galaxies, while in physics and chemistry, the Voronoi diagram can be used to compute domains of action or area of influence. Still, in metrology and geography, the Voronoi diagram is used to estimate ore reserves from sample of deposits by computing ‘Thiessen polygons’. More recently, Voronoi diagram has been used for location updates and routing in wireless networks and mobile computing [1]. For further discussion on applications of Voronoi diagram, see [2].

Let S be a set of n planar sites. The Voronoi diagram partitions the plane into regions such that each region contains one site. The region of a site s consists of all points closer to s than to any other sites on the plane. Thus, given an arbitrary point p in the plane, one can determine which of the n sites of S is closest to p by determining which of the n regions contains the point p . The region of a site is called the Voronoi cell or Voronoi polygon. The vertices of these polygonal regions are called Voronoi vertices and the polygonal boundaries i.e., edges of the regions, are called Voronoi edges.

The Voronoi diagram solves many fundamental proximity problems. Some of these problems include

the closest pair problem, which can be stated as find the closest pair of sites among a set of sites, the nearest neighbor problem that is find the nearest neighbor of each of a given set of sites, the convex hull the Delaunay triangulation and the Euclidean minimum spanning tree. See [2], [3] for details.

The basic properties of Voronoi diagram are as follows: The circle determined by the three nearest neighbors of S is centered at Voronoi vertex and further will not contain any of the $n - 3$ other sites of S in its interior. The converse is also true: If the circle determined by the three sites of S does not contain any of the $n - 3$ other sites of S , then the center of that circle is a Voronoi vertex. Since the Voronoi diagram is a planar graph, the number of Voronoi vertices is at most $2n - 5$ and the number of edges is at most $3n - 6$ for $n > 2$. For further discussion on properties of Voronoi diagram, see [2], [3]

An $\Omega(n \log(n))$ time worst case lower bound can be shown for this problem by reducing it to sorting [4]. Shamos [4], [5] and Shamos and Hoey [6] describe an $O(n \log(n))$ time divide-and-conquer algorithm for construction of the planar Voronoi diagram. In 1988, Aggarwal, Chazelle, Guibas, O’Dunlaing, and Yap, propose a parallel algorithm that has time complexity $O(\log^2 n)$ and uses $O(n)$ processes on *CREW PRAM* model [7]. They parallelize the dividing chain tracing step. This requires the identification of those Voronoi edges that intersect the dividing chain. They performed this task by using planar point location technique. The authors use beachline data structure to achieve this goal. In 1990, Chang-Sung Jeong [8] gave a parallel time-optimal algorithm that runs in $O(\sqrt{n})$ on an $O(\sqrt{n} \times \sqrt{n})$ mesh. The algorithm is based on the Shamos’s divided-and-conquer [4] method.

The idea of this work is derived from the Shamos and Hoey [6]. While the approach is based on the divided chain technique presented in [9], our proposed algorithm diverges in a significant aspect: the way algorithm finds the sites by first computing intersection points between bisectors and Voronoi edges that are used in the construction of the dividing chain. Reader

may consult [10].

The rest of the paper is organized as follows. The Section 2 presents the model of computation. The Section 3 presents an outline of the algorithm. The Section 4 describes the merging process used in the construction of the Voronoi diagram. The main contribution of this paper is in Sections 5 and 6. The Section 7 deals with the complexity of the proposed algorithm and Section 8 presents correctness of the proposed algorithm. Lastly, in the Section 9, we draw a conclusion.

2 The Parallel Model

The hypercube is an important topology that has received tremendous attention in the theoretical literature [11] In [12], Foster describe various useful algorithms on hypercube and a parallel algorithm template in addition to a detailed description of communication structure of hypercube. The readers may refer to [13] for basic communication operations on hypercube. The d -dimensional hypercube has $n=2^d$ nodes and $d2^{d-1}$ edges. Each node corresponds to a d -bit binary string. The hypercube connects the two nodes with an edge if and only if binary strings differ in exactly one bit. As a result each node is incident to $d=\lg(n)$ other nodes. Let $n=2^d$ processors P_0, P_1, \dots, P_{n-1} be available for $d > 1$. Further, let i and $i^b \leq n-1$, whose binary representation differ only in position b , $0 \leq b \leq d$. A d -dimensional hypercube is formed by connecting each processor P_i , $0 \leq i \leq n-1$, P_i^b links, for all $0 \leq b \leq d$.

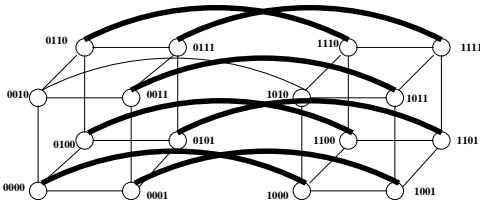


Fig. 1. The 16 nodes hypercube.

The recursive property of an hypercube is that k edges in a hypercube form a perfect matching for each edge k , $1 \leq k \leq \lg(n)$. Moreover, removal of k edges for any $k \leq \lg(n)$ results in two disjoint subhypercubes each containing $n/2$ nodes. On the other hand, an n -node hypercube by simply connecting two corresponding i^{th} nodes for $0 \leq i \leq n/2$. Additional properties of hypercube are that it has low diameter, $\lg(n)$, and high bisection width, $n/2$. Moreover, hypercube is node and edge symmetric.

3 The Outline of the Algorithm

Foremost sort the n planar sites belong to the given set S in parallel according to abscissa using $O(n)$ PEs of hypercube in $O(\lg^2 n)$ time using bitonic sort [14]. Then partition the n sites such that each partition gets

at most one site from original set S . Now map each subset or equivalently each site onto PEs such that PE_0 gets the site with the lowest x -coordinate and PE_{n-1} gets the site with the highest x -coordinate. In the merging phase all PEs compute $n/2$ sub-Voronoi diagrams. Note that in this case they are just $n/2$ bisectors. PE_i and PE_{i+1} compute the perpendicular bisector of sites s_i and s_{i+1} in S where $i=0, 2, 4, \dots, n-2$. We name the bisectors $B(s_i, s_{i+1})$, the $Vor_1(i)$ and the bisector $B(s_{i+2}, s_{i+3})$, the $Vor_1(i+1)$ where $i=0, 2, 4, \dots, n-4$. In the second stage, we merge pairs of $Vor_1(i)$ and $Vor_1(i+1)$ to form $Vor_2(i)$ and $Vor_2(i+1)$. This gives the resulting sub Voronoi diagrams. In this manner, we recursively stitch the sub-Voronoi diagrams $Vor_d(i)$ and $Vor_d(i+1)$ for $2 \leq d \leq \lg(n)$. In the last stage of merging, we stitch sub-Voronoi diagrams $Vor_{\lg(n)-1}(i)$ and $Vor_{\lg(n)-1}(i+1)$ to shape a complete (inclusive) Voronoi diagram, $Vor(S)$.

Like sequential counterpart [6] of this algorithm, the most vital part of stitching progression is the creation of dividing chain. Our algorithm creates the dividing chain by first searching the intersection sites between bisectors and Voronoi edges. Next, we outline the course of action to build this intersection set.

4 Description of Merging

In each merging step, $2 \leq d \leq \lg(n)$, define all unbounded sites whose associated Voronoi polygons are infinite in partition sets. We call such sites unbounded. Define sites that have maximum y -coordinates and minimum y -coordinates in subsets S_i (subset strictly on the left) and S_{i+1} (subset strictly on the right) as y_{max} and y_{min} respectively. The unbounded sites arranged in the counterclockwise direction from y_{max} to y_{min} are identified as α_i sites where $y_{max} \leq i \leq y_{min}$ and those sites that are arranged in clockwise direction from y_{max} to y_{min} be defined as β_i sites, where $y_{max} \leq i \leq y_{min}$.

In subsequent merging stage, every PEs that holds α sites in subset S_i will search in parallel for β sites in the subset S_{i+1} , which has the smallest y -coordinate greater than the y -coordinate of its own. After this search, PEs of α sites will search for β sites in the strictly right subset S_{i+1} , which has largest y -coordinate that is smaller than the y -coordinate of its own. The search of these sites can easily be accomplished using binary search on unbounded sites in $O(\lg(n))$ time. Quinn [15] discussed in detail the parallel search and embedding various graphs into hypercube.

After the detection of sites, we compute the perpendicular bisector, $B(\alpha_i, \beta_{i+1})$, of unbounded sites in the strictly left partition, α_i , and unbounded sites in strictly right partition, β_{i+1} . Then find the intersection

site between Voronoi edges of sites α_i , and β_{i+1} . We can accomplish this task by using “walking method” described in incremental construction of Voronoi diagram [16], [17].

After the detection of intersection site between Voronoi edge and perpendicular bisector, $B(\alpha_i, \beta_{i+1})$, include it in the set of intersection, X . Simultaneously, algorithm runs this procedure for all α and β sites in strictly left and strictly right partitions respectively. The consequence, after running this procedure on all $lg(n)$ dimensions of $O(n)$ hypercube, is the finished Voronoi diagram of the given set of sites, S .

5 The Algorithm

This section presents the parallel algorithm for Voronoi diagram construction using the hypercube model of computation and based on [18].

Input: A set of $n=2^d$ planar sites.

Output: A set of Voronoi edges specified by set of Voronoi edges.

- 1) Sort the $n \in S$ sites by their x -coordinates using bitonic sort and map n sites s_0, s_1, \dots, s_{n-1} onto the $O(n)$ PEs such that each PE_i obtain site s_i , $0 \leq i \leq n-1$.
- 2) For all $0 \leq i \leq n-1$ do in Parallel
 - a) For all $0 \leq i \leq n-1$ do
Construct perpendicular bisector $B(s_i, s_{i+1})$ using PE_i and PE_{i+1} for $i=0, 2, 4, \dots, n-2$.
 - b) For all $2 \leq k \leq lg(n)$ do
Merge $Vor_d(S) = Vor_{d-1}(i) \cup Vor_{d-1}(i+1)$ by constructing the set of Intersection, X .
- 3) Sort intersection points in X , with respect to their y -coordinates using bitonic sort.
- 4) Add polygonal line and discard the portion of $Vor_{d-1}(i)$ which is strictly left of the polygonal line and discard the portion of $Vor_{d-1}(i+1)$ which is strictly right of the polygonal line.

Following algorithm computes the intersection points between bisectors and Voronoi edges that are used in the construction of dividing polygonal chain.

- 1) Search unbounded sites in strictly left subset, S_L , and strictly right subset, S_R .
 - a) Search for the maximum and minimum y -coordinates sites in subsets S_L and S_R such that $y_{max}, y_{min} \in S_L$ and $y_{max}, y_{min} \in S_R$.
 - b) Identify α and β sites S_L and S_R respectively where, $y_{max} \leq i \leq y_{min}$.
- 2) In parallel, every PE that hold $\alpha \in S_L$ will search for $\beta \in S_R$, which has smallest y -coordinate greater than its own. Simultaneously, every PE

of $\beta \in S_R$ will search for $\alpha \in S_L$, which has largest y -coordinate that is smaller than its own.

- 3) In parallel, do step 2 after interchanging the role of α and β sites.
- 4) Compute the perpendicular bisector of $\alpha \in S_L$ and $\beta \in S_R$, $S = S_L \cup S_R$. That is, $B(\alpha_L, \beta_R)$.
- 5) Search the intersection points between Voronoi edges associated with Voronoi polygons of $\alpha \in S_L$ and $\beta \in S_R$ sites perpendicular bisector $B(\alpha, \beta)$.
- 6) Include intersection points found in step 5 in the intersection set, X , i.e., $X \leftarrow X \cup \{x\}$.
- 7) Return intersection set, X .

6 Construction of the Intersection Set

Now, we scrutinize the algorithm by merging two Voronoi diagrams. Suppose that we have obtained the left Voronoi diagram Vor_L and the right Voronoi diagram Vor_R of sets S_L and S_R where $S = S_L \cup S_R$.

Search all unbounded sites in strictly left and strictly right partitions S_L and S_R respectively. Consider only those sites in S_L that are arranged in counterclockwise order from highest y -value to lowest y -value. Similarly, consider only those sites in S_R that are arranged in clockwise order from highest y -value to lowest y -value. We have $\{6, 7, 3\} \in S_L$ and $\{13, 9, 8, 10\} \in S_R$.

First, we construct the semi-infinite ray by computing perpendicular bisector, $B(6, 13)$, of sites 6 and 13. Note that these sites have maximum y -coordinates in their respective sets. Then find the intersection point of bisector $B(6, 13)$ and $V(13)$. Let it be point x_1 . Similarly, construct semi-infinite ray using sites $3 \in S_L$ and $10 \in S_R$; and find the intersection point x_6 . See figure 2.

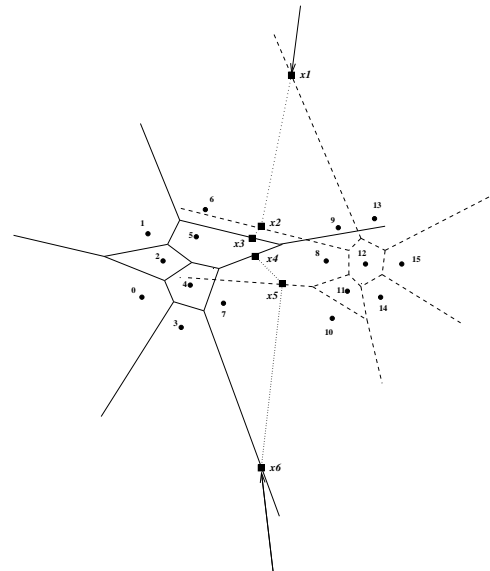


Fig. 2. Boxes indicate intersection points.

At this stage, we have set of intersection $X_1 = \{x_1, x_6\}$. Now site $9 \in S_R$ has largest y -coordinate smaller than y -coordinate of $6 \in S_L$. Compute the bisector $B(6, 9)$ and find the intersection point of bisector $B(6, 9)$ and $V(9)$, which are x_1 and x_2 . Include these intersection points in the set of intersection and we get $X_1 = \{x_1, x_2, x_6\}$. Next unbound site in the set S_L is site 7. Site $8 \in S_R$ has the smallest y -value larger than $7 \in S_L$. Again, compute the bisector $B(7, 8)$, and find the intersection point of $B(7, 8)$ and Voronoi edges of site 7 and 8, which are x_4 and x_5 . Include these points in the set of intersection, which now becomes $X_1 = \{x_1, x_2, x_4, x_5, x_6\}$. Simultaneously, processors of set S_L run the same procedure and get the set of intersection points, $X_2 = \{x_1, x_2, x_3, x_5, x_6\}$. The union of sets X_1 and X_2 gives a set of intersection $X = X_1 \cup X_2 = \{x_1, x_2, x_3, x_4, x_5, x_6\}$. Now, sort the set of intersection with respect of y -coordinate using bitonic sort and add the polygonal line $(\overline{x_1x_2}, \overline{x_2x_3}, \overline{x_4x_5}, \overline{x_5x_6})$. Finally, delete from Vor_L the part to the right of the polygonal line and delete from Vor_R the part to the left of the polygonal line and return the resultant Voronoi diagram.

7 Algorithm Complexity

Theorem 1 *The Voronoi diagram of a set of n sites in the plane can be constructed on a $O(n)$ hypercube computer with in $O(\lg^3(n))$ time.*

Proof: In sorting phase of an algorithm, we can sort n planar points on hypercube using bitonic sort in $O(\lg^2(n))$ time. The partition phase i.e., distribution of n planar sites on $O(n)$ -hypercube can be done in constant time since all geometric operations at this stage take constant amount of time. Note that this argument is based on the fact that the planar sites have already been sorted. It is easy to see that conquer phase can be performed in constant time. In merging phase, each PE builds a new Voronoi diagram by calling several sub algorithms for finding the perpendicular bisector, delete Voronoi edges, new Voronoi edges, etc. Each of these sub algorithms takes a constant time. Suppose, the running times of these sub algorithms be c_1, c_2, \dots, c_n then we have $C = \max(c_1, c_2, \dots, c_n)$. The searching in this phase for sites associated with infinite polygons take $O(\lg(n))$ time using binary search. Hence, running time of merging step is no more than $O(\lg(n))$, since C is a constant. Moreover, sorting the set of intersection with respect to y -coordinate using bitonic sort takes $O(\lg^2(n))$ time and adding polygonal line takes constant time.

Now, let $t(n)$ denote the time required by the algorithm for n sites. Then, we get the recurrence relation $t(n) = t(n/2) + O(\lg^2(n))$. This equation implies that $t(n) = O(\lg^3(n))$. Note that communication

overhead is not included in the solution. Since in all merging stages, messages are exchanged between PE s that are exactly on hop away therefore, the lengths of communication paths are always one. Clearly, sending and receiving a message from one node (PE) to other node (PE) of hypercube takes a constant time. Suppose, in d^{th} dimension, communication required time C_d . Hence, the total time required by all dimensions, $(\lg(n))$, of hypercube would be $t_{com} = C_{com}(n/2) + C_d = C_{com}(n/2) + C_d(\lg(n)) \leq 2 \times C \times \lg(n) = \lg(n)$. Hence, the total communication time to build a complete Voronoi diagram would be $t_{com} = O(\lg(n))$. From these two pieces of information, we get $t(n) = t(n) + t_{com}(n) = O(\lg^3(n)) + O(\lg(n)) = O(\lg^3(n))$. Therefore, the Voronoi diagram of a set of n sites in the plane can be computed on an $O(n)$ -hypercube computer in $O(\lg^3(n))$ time. This completes the proof. ■

8 Correctness

In this section, we prove the correctness of an algorithm by showing that the algorithm will result in the Voronoi diagram of the given set of planar sites. But before stating and proving lemmas and theorem, we outline the procedure. We need to prove that this algorithm will contain all Voronoi vertices and edges contained in $Vor(S)$, and that every vertex and edge that is retained by the algorithm is contained in $Vor(S)$. In this algorithm, the individual edges of divided chain can be obtained by joining intersection points in the set X . Since edges of divided chain are true Voronoi edges, we can derive their properties by exploiting the established properties of the Voronoi diagram [3]. In this correctness, we demonstrate that the edges of polygonal line are indeed edges of divided chain and further the algorithm constructs every edge of divided chain by using intersection set X . Following lemmas based on [3] and [18].

Lemma 1 *Only the closest sites belong to other partition participated in the constructing of dividing chain.*

Proof: Consider a site s_j be the closest to s_i and let v be the midpoint of segment $\overline{s_i s_j}$. Now, suppose the v does not lie on the part of dividing chain corresponding to s_i and s_j . Then the line $\overline{s_i v}$ intersect some part of dividing chain corresponding to s_i and s_k (say) at point u . Which means, the length $|\overline{s_i u}| < \text{length } |\overline{s_i v}|$, therefore length $|\overline{s_i s_k}| < 2 \text{ length } |\overline{s_i v}|$. But since $2 \times \text{length } |\overline{s_i v}| = \text{length } |\overline{s_i s_j}|$, implies that the length $|\overline{s_i s_k}| < \text{length } |\overline{s_i s_j}|$. And we would have s_k closer to s_i than s_j , which is a contradiction. ■

Following lemma is based on [18].

Lemma 2 *The site(s) participated in the construction of dividing chain if and only if it is unbounded.*

Proof: Consider sites $s_1, s_2, s_3, s_i \in P_L$ such that s_i is bounded and a site $x \in P_R$. In the partition P_L , there must exist three Voronoi circles C_{12}, C_{13} and C_{23} containing the common site s_i . That is, $C_{12} = \{s_i, s_1, s_2\}, C_{23} = \{s_i, s_2, s_3\}$ and $C_{13} = \{s_i, s_1, s_3\}$. WLOG, consider circle C_{12} with vertices $\{s_i, s_1, s_2\}$ on the boundary of circle. Also consider an external arc A_{12} on the circle C_{12} between sites s_1 and s_2 not containing s_i . We claim that any site $x \in P_R$ is closer to one of the sites s_1, s_2 or s_3 than it is to s_i . Consider the segment $\overline{xs_i}$. By the Jordan curve theorem, segment $\overline{xs_i}$ must intersect one of the sides of triangle $s_1s_2s_3$. WLOG, consider it intersects $\overline{s_1s_2}$; therefore, segment $\overline{xs_i}$ also intersects arc A_{12} at point y . But y is closer to either s_1 or s_2 than to s_i , hence the claim. Since point $x \in P_R$ is closer to s_1, s_2 or $s_3 \in P_L$ than to $s_i \in P_L$, only s_1, s_2 or s_3 of partition P_L participated in construction of dividing chain with site $x \in P_R$.

Conversely assume that site s_i is bounded and let the sequence of boundary edges e_1, e_2, \dots, e_k where $k \geq 3$. Each boundary edge belongs to the part of bisector. Hence the lemma. ■

Theorem 2 *The unbounded site that is closest to some unbounded site in other partition involves in construction of dividing chain.*

Proof: Immediately from lemma 1 and lemma 2. ■

Now we show that the algorithm can construct the part of the dividing chain associated with bounded site, $s_k \in P_L$. By lemma 2, site $s_y \in P_R$ constructs the bisector (part of dividing chain) with $s_i \in P_L$ or $s_j \in P_L$ (not with $s_k \in P_L$ since it is bounded). From the intersection of bisector and Voronoi edge, we get the intersection point. WLOG assume it is x_1 . After constructing a bisector with site s_i or site s_j (whichever be the closest, see lemma 1), the remaining site becomes closest. Therefore, site $s_y \in P_R$ constructed the bisector with that site and so we have intersecting point x_2 on Voronoi edge.

Since, x_1 is a center of circumcircle corresponding to s_k, s_y and s_i ; and x_2 is a center of circumcircle corresponding to s_k, s_y and s_j (Note that s_k and s_y are common in both circles). We simply join the intersection points x_1 and x_2 to get a Voronoi edge associated with site s_k and site s_y , i.e., bisector $B(s_k, s_y)$, hence the part of divided chain.

Now we show that each vertex, v , of the Voronoi diagram produced by proposed algorithm for a set S is the center of a circle $c(v)$ going through just three points from the set S .

Following lemmas are based on properties of Voronoi diagram in [2].

Lemma 3 *At each Voronoi vertex produced by the algorithm, only three of its edges meet.*

Proof: Suppose that a Voronoi vertex v is the intersection points of more than 3 Voronoi edges. The proposed algorithm numerated the edges e_i in clockwise direction.

Let us numerate the Voronoi regions including v in such a way that the edge e_i is common to the Voronoi regions Vor_i and Vor_{i+1} , $i = 0, \dots, k - 1$. Also let the point s_i from the set S belong to Vor_i . Then the vertex v is equidistant from the points s_i and s_{i+1} , $i = 0, \dots, k - 1$, see line 4 of construction of dividing polygonal chain. Since v_i lies on the perpendicular to the segment $s_i s_{i+1}$ at its middle points. Therefore, all point s_i lie on the circle center at v , and by its assumption, $k \leq 3$.

On the other hand, 2 Voronoi edges cannot meet at Voronoi vertex, because if they did, the edges e_0 and e_1 would belong to both Vor_0 and Vor_1 and lie on the same straight line, hence v would not be a Voronoi vertex. ■

Lemma 4 *Let v_1 and v_2 be two arbitrary vertices of the Voronoi diagram for the s . Then the triangle $T(v_1)$ have no common inner points.*

Proof: Consider the circle $C(v_i)$ centered at v_1 . By definition of Voronoi diagram these circles circumscribe the $T(v_i)$. Clearly if the region bounded by $C(v_i)$ do not intersect each other, the corresponding triangles also do not intersect. Suppose that the regions bounded by circles $C(v_i)$ intersect. We assert that then it is easy that one of these regions cannot entirely contain the other one. Therefore, the circle $C(v_i)$ intersect at a pair of points.

Let l be the part of polygonal line going through these intersection points. Our goal is to show that the triangles $T(v_i)$ lie on different sides of the divided chain l . Consider the triangle $T(v_1)$ and suppose that some of its inner points are on the same half plane bounded by polygonal line as the vertex v_2 .

Then there exists a vertex of $T(v_1)$ lying on this half-plane, and this vertex lies inside the region bounded by the circle $C(v_2)$. This contradicts our assertion. ■

Kruskal et al. [19] introduced a class *EP* of efficient parallel algorithms, in which they measured the performance of parallel algorithms relative to that of sequential algorithms. *EP* is defined as follows. Let the time complexity of a sequential algorithm and a parallel algorithm using $p(n)$ processors for instances of size n be given by $t(n)$ and $T(n)$, respectively. An algorithm is polynomially fast and has constant efficiency, if $T(n) = O(t(n)^\epsilon)$ with $\epsilon < 1$ and $T(n) \times p(n) = O(t(n))$. The class that contains these algorithms is called *EP*. Using the definition of *EP* class, we established the following.

Lemma 5 *The algorithm belongs to the EP class.*

Proof: Here we need to show that $t_p = O(t_1^\epsilon)$ for $\epsilon < 1$, where t_1 and t_p are the times required using one and p processors, respectively. From Theorem 1, the required by the algorithm to construct the Voronoi diagram is $O(\lg^3(n))$ time on a $O(n)$ hypercube.

Therefore it holds that

$$\begin{aligned} t_p &= O(\lg^3(n) \cdot n \lg n) \\ &= O(n \cdot n^\delta) \text{ for } \delta > 0 \end{aligned}$$

Now for any δ such that $0 < \delta < v$, and $\frac{1+\delta}{1+v} \leq \epsilon < 1$ holds.

$$t_p = O(n^{1+\delta}) = O(n^{(1+v)t}) = O(T_1^\epsilon)$$

Thus, the algorithm belongs to the class *EP*. ■

9 Conclusion

This paper addresses the problem of constructing the planar Voronoi diagram in parallel on the hypercube model of computation. While the more novel results have been in the area of parallel algorithms, real focus of this paper is to combine theoretical and practical results to develop a practical parallel algorithm by exploiting the characteristic properties of computational model. In this paper, we have shown that the problem of constructing planar Voronoi diagram can be solved using hypercube model of computation with n processors in $O(\lg^3(n))$ time. The fact that the bitonic sort algorithm bounds the time complexity of the algorithm also showed that sorting helps in the constructing of Voronoi diagram.

The fundamental problem in the constructions of sequential Voronoi diagram is the construction of polygonal line quickly. This paper shows that this fundamental problem remains unchanged even in the parallel setting. One of the features of the algorithm presented is to construct a polygonal dividing chain from intersection points. Hence, we have shown that it is certainly possible to construct the polygonal chain without explicitly computing all part of it.

References

- [1] I. Stojmenovic, Location updates for efficient routing in ad hoc networks, in I. Stojmenovic(Ed.), *Handbook of Wireless Networks and Mobile Computing*, Wiley, New York, 2002, pp. 451-471.
- [2] A. Okabe, B. Boots, K. Sugihara, and S. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, Wiley, Chichester, England, 2000.
- [3] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer Verlag, New York, 1985.
- [4] M. I. Shamos, "Geometric complexity," *Proc. 7th Annual ACM Symposium on Automata and Computability Theory*, 1975, pp. 224-233.
- [5] M. I. Shamos, "Computational geometry," Thesis, Dept. Computer Science, Yale University, USA, 1978.
- [6] M. I. Shamos and D. Hoey, "Closest point problems," *Proc. IEEE Symposium on Foundations Computer Science*, 1976, pp. 208-215.
- [7] A. Aggarwal, B. Chazelle, L. Guibas, C. O'Dunlaing and C. Yap, "Parallel geometry," *Algorithmica*, 3(3), pp. 293-328, 1998.
- [8] C. S. Jeong and D. T. Lee, "Parallel geometric algorithms on a mesh connected Computers," *Algorithmica*, 5(2), pp. 155-178, 1990.
- [9] L. Chen, "Constructing the Voronoi diagram on hypercube multicomputers," Thesis, Dept. Computer Science, University of Missouri-Columbia, USA, 1993.
- [10] R. B. Muhammad, "The parallel Voronoi diagram on hypercube model of computation," Thesis, Dept. Computer Science, Kent State University, USA, 2003.
- [11] D. E. Culler, J. P. Singh and A. Gupta, *Parallel Computer Architecture: a Hardware/Software Approach*, Morgan Kaufmann, San Francisco, 1999.
- [12] I. Foster, *Designing and building parallel programs: concepts and tools for parallel Software engineering*, Addison-Wesley, NY, 1994.
- [13] A. Grama, A. Gupta, G. Karypis and V. Kumar, *Introduction to Parallel Computing* Addison-Wesley, NY, 2003.
- [14] K. E. Batcher, "Sorting networks and their application," *Proc. of the AFIPS Spring Joint Computer Conference*, vol. 32, AFIPS Press, Reston, VA, 1968, pp. 307-324.
- [15] M. J. Quinn, *Parallel computing: theory and practice*, McGraw-Hill, Inc., USA, 1994.
- [16] P. J. Green and R. R. Sibson, "Computing Dirichlet tessellations in the plane," *Computer Journal*, 21(2), pp. 168-173, 1978.
- [17] M. Iri, K. Murota and T. Ohya, Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms, *J. Operational Research Society, Japan*, 27(4), pp. 306-336, 1984.
- [18] R. B. Muhammad "A parallel Voronoi diagram on hypercube computers," *Proc. of the Sixteenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, November 9-11, 2004, Cambridge, USA, pp. 542-547.
- [19] C. P. Kruskal, L. Rudolph, and M. Snir, "A complexity theory of efficient parallel algorithms," *Theoretical Computer Science*, (71):95-132, 1990.