A Distributed Algorithm for Computing Voronoi Diagram in the Unit Disk Graph Model

Yurai Núñez-Rodríguez[(1)] Henry Xiao[(1)] Kamrul Islam[(1)] Waleed Alsalih[(1)]
[(1)]Queen's University(yurai/xiao/islam/waleed@cs.queensu.ca)
School of Computing, Queen's University
Kingston, Ontario, Canada K7L 3N6

Abstract

We study the problem of computing Voronoi diagrams distributedly for a set of nodes of a network modeled as a Unit Disk Graph (UDG). We present an algorithm to solve this problem efficiently, which has direct applications in wireless networks. Comparing with some existing algorithms, our algorithm correctly computes the complete Voronoi diagram and uses a significantly smaller number of transmissions. Furthermore, useful geometric structures such as the Delaunay triangulation and the convex hull can be obtained through our algorithm.

1 Introduction

With the recent wave of research in wireless networks, geometric structures like the Voronoi Diagram (VD) have been studied in different computational platforms such as mobile devices and sensors [11, 10, 2]. These devices have limited battery power and usually cooperate with each other without a centralized control. Given a set of n points in the plane, it is known that the VD can be computed in $O(n \log n)$ time in a centralized fashion (see Fortune's algorithm for an example [6]). In a distributed setting, computing the VD introduces new challenges [4, 5, 10, 11, 2].

We investigate the problem of distributedly computing the VD of a network where the nodes are modeled as points in the plane. Our main goal is to compute the accurate VD while minimizing the communication cost. The communication cost is proportional to the number of transmissions between two adjacent nodes. We propose a distributed algorithm to compute the VD of a connected network. Our approach is purely based on cooperation among nodes. A preliminary version of our work has been announced in [1].

Throughout this paper, a network is modeled as a Unit Disk Graph (UDG). According to this model, an edge between two nodes v and w exists if and only if the Euclidean distance between v and w is not greater than one unit (normalized). We assume that the induced UDG is connected. The nodes are also assumed to know their geographic locations. This is usually achieved through GPS or other techniques [9, 7]. The communication of the network is not required to be synchronous.

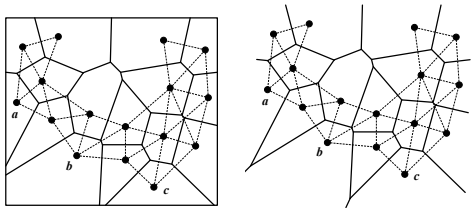


Figure 1: Voronoi diagrams of a set of nodes. Edges between nodes are represented by dashed lines. The bounded VD (left) and the complete VD (right).

In recent attempts to design distributed algorithms for computing the VD, Bash and Desnoyers [2] proposed an algorithm to compute the *bounded* VD (see Figure 1) utilizing the GPSR routing protocol ([3, 8]) for wireless sensor networks. The basic idea is to successively refine the approximations of the Voronoi cells upon discovery of other sensors in the network. Given a sensor s, their algorithm starts with the entire bounded region as the approximation of the Voronoi cell of s. A probing message is then sent to each of the vertices of the current Voronoi cell using GPSR, which will be delivered to a sensor t that is the nearest to the probed vertex. Sensor t replies to s and the current Voronoi cell of s is refined with respect to t. No more probes are sent by s once it becomes the nearest sensor to all its Voronoi vertices. Bash and Desnoyers' algorithm [2] is referred to as BD07[1] in this paper.

In the following, we describe our distributed algorithm for computing the VD and prove its correctness. We further discuss more practical environments where the correctness of our algorithm holds. Some of our simulation results in comparison with BD07 are presented in Section 4.

[1]Limited by the space, optimizations of BD07 are not reviewed here.

2 Distributed Computation of Voronoi Diagram

We propose a distributed algorithm, namely the *completely cooperative* (CC) algorithm, for computing the VD of a set of nodes in the plane. Recall that the network is modeled as a UDG. The nodes are assumed to be in general position[2].

2.1 The CC algorithm

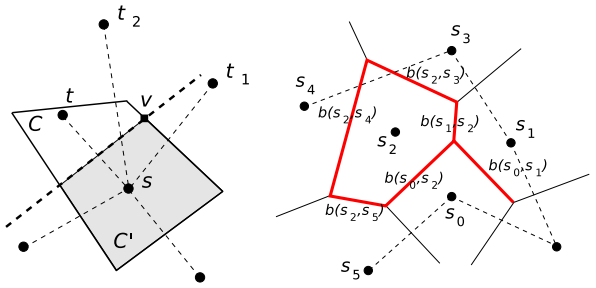
The basic idea behind the CC algorithm is that nodes do not need to send out queries to discover their Voronoi neighbours; instead, nodes are informed about possible Voronoi neighbours by other neighbours. We adopt the following terminology. Let S be a set of nodes embedded in the plane and let $G = (S, L)$ be the connected UDG induced by S, where $L \subseteq S \times S$ contains pairs of nodes that are within unit distance. Let $VD(G)$ be the VD of G. We refer to an element of S as a *node* and an element of L as a *link*, saving the term *edge* for the corresponding element in the VD. Similarly, we refer to nodes that share a link as *adjacent* and to nodes that share a Voronoi edge as *neighbours*.

Let s be a node that receives a message about a potential (Voronoi) neighbour, t, at some point during the computation. Then s computes the intersection of its current cell, C, with the half plane defined by the bisector between itself and t. We call this step the *refinement* of a cell. If the new cell C' resulting from the intersection is equal to C, t is ignored; otherwise ($C' \subset C$), t becomes a neighbour of s. In the latter case, new vertices appear on C' and some vertices of C fall outside of C'. Figure 2a illustrates the refinement process. Two adjacent vertices that fall outside of C', define a piece of bisector for a node t_2 that is then discarded. A new vertex v on C' is created by the intersection of the bisector between s and t, and the bisector between a certain node t_1 and s. Therefore, t and t_1 may be neighbours of each other since they have a common Voronoi vertex according to the cell of s. Consequently, s informs both t and t_1 about each other. This way, the information about possible neighbours flows towards the corresponding nodes until each node finds all its neighbours.

Initially, the cell of any node s is equal to the entire plane. Then all nodes broadcast their locations triggering the entire computation as explained above.

Algorithm 1 is a pseudocode description of the CC algorithm. In the description of the algorithm, node s has location $s.loc$, a field $s.cell$ that stores the description of its Voronoi cell, and a message queue $s.q$. The refinement of $s.cell$ with respect to t is done through $s.refine(t.loc)$. The method $send\_message(t_1.loc, t_2.loc)$ sends a message to node t_1 containing the location of t_2 and results in $t_2.loc$ being added to the message queue of t_1 (i.e., $t_1.q$). Vertices in $s.cell$ can be accessed through $s.cell.verts$. A vertex v of a Voronoi cell is equipped with a method $v.third(s.loc, t.loc)$ that returns the location of the third node associated to v that is neither s nor t.

[2]No three nodes are collinear and no four nodes fall on the same circle



(a) Cell Refinement (node s discovers node t)

(b) Disconnected Node

Figure 2: Illustration of the CC algorithm.

Algorithm 1: Completely Cooperative (CC)

```
// Initialize the cell
s.cell = ENTIRE_SPACE
// Broadcast the node location to all adjacent nodes
s.send_message( BROADCAST, s.loc )
// Process each (node) message in the queue
while( t.loc = s.q.get_message )
    old_Cell = s.cell
    s.cell = s.refine( t.loc )
    for each( v in s.cell.verts and not in old_Cell.verts )
      // Notify each pair of possible neighbours about each other
        s.send_message( t.loc, v.third( s.loc, t.loc ) )
        s.send_message( v.third( s.loc, t.loc ), t.loc )
    end
end
```

2.2 Proof of correctness

Theorem 1 *Let G be the induced UDG of a set of nodes in the plane. The CC algorithm computes the correct Voronoi cell of every node in G.*

Proof. The reader is referred to Figure 2b for a graphical description of this proof. It is not hard to see that the algorithm terminates after a finite number of steps given that every message sent is the result of the refinement of a cell whose area has decreased. Therefore, the computation ends as the approximations of the Voronoi cells converge to the correct cells.

In order to prove that the algorithm determines the correct cells, suppose, for the sake of contradiction, that the Voronoi cell corresponding to a node s_0, was not properly determined. This means that s_0 did not find at least one of its (Voronoi) neighbours. Let s_1 be a neighbour of s_0 that was not discovered by the application of Algorithm 1 to s_0. It would be a contradiction that $(s_0, s_1) \in L$ since adjacent nodes know about each other and must have been neighbours from the initial

refinements. Therefore, we assume that there is no link between s_0 and s_1.

Let $b(s_0, s_1)$ be the edge of $VD(G)$ corresponding to the bisector between s_0 and s_1. Note that VD edges are segments, lines, or semilines. $b(s_0, s_1)$ can be a line only if $|S| = 2$, but since there is no link between s_0 and s_1, G would not be connected, which is a contradiction. Therefore, $b(s_0, s_1)$ must be a segment or a semiline. In both cases $b(s_0, s_1)$ has at least one end point $(v(s_0, s_1, s_2))$ that is a Voronoi vertex of the cells associated to s_0, s_1, and a third node s_2. The CC algorithm guarantees that s_2 informs s_0 and s_1 about each other once the corresponding bisectors have been considered and the intersection point $(v(s_0, s_1, s_2))$ has been found. Since s_0 and s_1 were not informed about each other, one of three possible cases must have occurred: s_2 did not find s_0, or s_1, or both. Without loss of generality, we assume that s_2 did not find s_1. The same reasoning applied to s_0 and s_1 can be applied to s_1, s_2 and a third node $s_3 \neq s_0$. This process can be repeated until one of two stop conditions is satisfied: (1) a cycle is created when a vertex is involved twice, (2) a semiline bisector in $VD(G)$ is reached. If this process ends with a semiline between two unbounded cells, the same procedure is applied starting at the other endpoint of $b(s_0, s_1)$, if any. This process again ends because of condition (1) or (2).

In the end, this process leads to a cycle or path P consisting of missing Voronoi edges that partitions the plane into two disjoint regions and is not crossed by any link between neighbours. It is not hard to see that if P is not crossed by any link between neighbours, it can not be crossed by any other link, given that G is the induced UDG. Therefore, G would have two disconnected subgraphs (one in each region) which contradicts the initial assumption. □

2.3 Optimizations

The CC algorithm is described in its simplest form. Some optimizations can be introduced to make it more efficient. First of all, before the initial refinements, every node s broadcasts its adjacency list. Every message that involves s includes its adjacency list. If a node r, that is about to inform two nodes s and t about each other, finds s (resp. t) in the adjacency list of t (resp. s), no notification is sent. This remarkably reduces the number of transmissions. However, some lists of adjacency can be significantly large. So only the information of a bounded number of adjacent nodes is sent along. We have set this bound to 6 for our experiments. The second key optimization consists in not sending two messages simultaneously to possible neighbours s and t while trying to inform them about each other. Instead, a message is first sent to s and then it is s that informs t, if required. This also reduces the number of messages since s and t may already be neighbours by the time s receives the notification and, consequently, there is no need to inform t.

3 Discussion and Extensions

Compared to BD07, the CC algorithm computes the complete VD. The CC algorithm also provides the complete Delaunay triangulation and the convex hull without additional communication. After the computation of the VD, a node that has two consecutive neighbours separated by an angle larger than π is on the outer face of the Delaunay triangulation and, therefore, on the convex hull.

The CC algorithm does not rely on any specific routing algorithm. Also, according to experimental results (see Section 4), the CC algorithm uses a smaller number of transmissions. Recall BD07 relies the GPSR routing protocol. For the sake of fairness, we use GPSR as the underlying routing protocol for the CC algorithm as well. We believe that with a better routing algorithm, the CC algorithm may further reduce the number of transmissions.

From a practical point of view, it is desirable to extend the underlying model beyond UDG. We can extend the scope of Theorem 1, with slight modifications, to more general graphs. Let G' be an arbitrary network obtained from G by removing links, and let $DT(G')$ be the subgraph of G' that contains only the links of G' that are edges of the Delaunay triangulation of S. Theorem 2 shows that as long as $DT(G')$ is connected, the CC algorithm computes the VD correctly.

Theorem 2 *Let G' be a subgraph of G, such that $V(G') = S$ and $DT(G')$ is connected. The CC algorithm computes the correct Voronoi cell of every node in G'.*

Proof. The proof is similar to the proof of Theorem 1. Once the cycle or path P is found, by assuming that the VD was not properly constructed, a contradiction arises with respect to the connectivity of $DT(G')$. In this case no link of $DT(G')$ crosses P while $DT(G')$ should be connected. □

4 Simulation Results

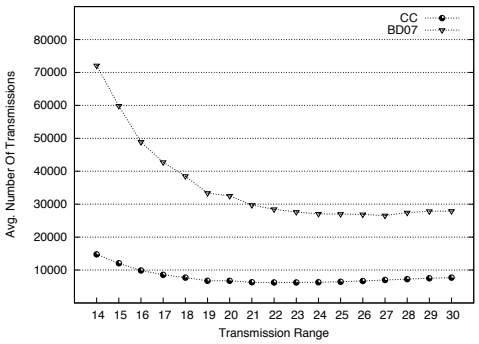
We have conducted intensive simulations on randomly generated networks to study the performance of the CC algorithm and compare the results with the BD07 algorithm [2].

Experiments were done with test sets consisting of 100 nodes randomly placed in a 100×100 unit grid. The density of the graph is controled by different transmission ranges (14 to 30 units). Also two different error rates, 0% and 20% are considered. The error rate is

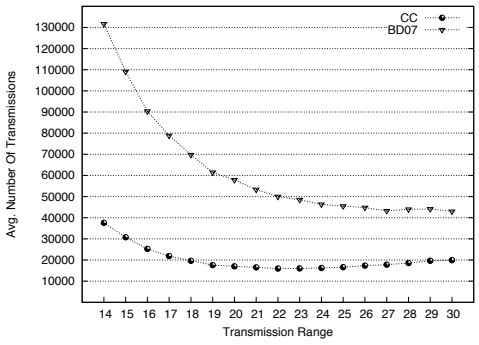
the probability of a transmission to fail. Hence, when a link temporarily fails, the transmission is repeated. For each transmission range and each error rate two algorithms (CC and BD07) are run with 1000 randomly generated networks as defined above. We also incorporate 50 randomly placed opaque obstacles in the form of bars of length 5. Special care is taken such that each graph generated contains a connected subgraph of the Delaunay triangulation as required by Theorem 2.

The entire number of simulations per algorithm is equal to [number of networks] $\times$ [number of transmission ranges] $\times$ [number of error rates] $= 1000 \times 17 \times 2 = 34,000$.

The graphs shown in Figures 3a and 3b provide the total number of transmissions in average for each transmission range with 0% and 20% error rates. For small values of the transmission range, BD07 requires a much larger number of transmissions than the CC algorithm. This is because the network is sparser and the GPSR protocol performs poorly. Because the CC algorithm does not require probing, it is not as affected as BD07 by small transmission ranges.



(a) 0% error rate



(b) 20% error rate

Figure 3: Simulation results.

5 Conclusion

We propose a novel distributed algorithm to compute the VD of a given network. The CC algorithm offers two significant advantages over previous works: (1) it computes the complete VD, and thus, can provide other useful structures such as the Delaunay triangulation and the convex hull; (2) it is more efficient in terms of the number of transmissions as verified through a large number of simulations.

Interesting problems remain open regarding the distributed construction of the VD. A natural question is whether it is possible to find a non-trivial efficient distributed algorithm for constructing the VD of arbitrarily connected networks.

Acknowledgments

We thank Selim Akl and David Rappaport for discussions and comments.

References

[1] W. Alsalih, K. Islam, Y. Núñez-Rodríguez, and H. Xiao. Distributed voronoi diagram computation in wireless sensor networks. In *SPAA '08: 20th ACM Symposium on Parallelism in Algorithms and Architectures*, Munich, Germany, June 2008.

[2] B. A. Bash and P. J. Desnoyers. Exact distributed voronoi cell computation in sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 236–243, New York, NY, USA, 2007. ACM Press.

[3] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[4] J. Byers, J. Considine, and M. Mitzenmacher. Geometric generalizations of the power of two choices. In *16th ACM Symp. on Parallel Algorithms and Architectures*, 2004.

[5] W. Chen, J. Hou, and L. Sha. Dynamic clustering for acoustic target tracking in wireless sensor networks. *IEEE Transaction on Mobile Computing*, 3(3):258–271, 2004.

[6] S. Fortune. A sweepline algorithm for voronoi diagrams. In *SCG '86: 2nd Annual Symposium on Computational Geometry*, pages 313–322. ACM Press, 1986.

[7] L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *IEEE/RSJ Conference on Intelligent Robots and Systems*, 2001.

[8] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *ACM MobiCom*, pages 243–254, 2000.

[9] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *6th Annual International Conference on Mobile Computing and networking*, pages 32–43, New York, NY, USA, 2000.

[10] M. Sharifzadeh and C. Shahabi. Supporting spatial aggregation in sensor network databases. In *12th International Symposium of ACM GIS*, 2004.

[11] Z. Zhou, S. Das, and H. Gupta. Variable radii connected sensor cover in sensor networks. In *IEEE SECON*, 2004.