# Distributed Voronoi partitioning for multi-robot systems with limited range sensors

K.R. Guruprasad and Prithviraj Dasgupta

***Abstract*—We consider the problem of distributed partitioning of an environment by a set of robots so that each robot performs its operations in the region within the corresponding cell. Voronoi partitioning is one of the most attractive techniques that has been used to solve this problem. It has been used in several distributed multi-robotic system and sensor network applications, such as sensor coverage, search and rescue, and coverage path planning. For a truly distributed implementation of such problems, each robot should be able to compute the corresponding Voronoi cell in a distributed manner. Further, in a practical application, the robots' sensors may have limited range, thus each robot may operate within a portion of its Voronoi cell constrained by the sensor range. We describe a distributed algorithm for computation of this range constrained Voronoi cell where each robot independently constructs chords corresponding to other robots that are within a distance of twice its sensor circle radius. A robot then uses a simple and fast technique to remove inessential chords to calculate the vertices of its Voronoi cell. We prove completeness and correctness of the proposed algorithm, and also provide the upper and lower bounds on the computational complexity of our algorithm. The theoretical results are validated with the help of experiments to show that for different values of sensor ranges, our proposed algorithm incurs a time complexity that is significantly lower than that of the existing full Voronoi partition computation algorithm. The maximum number of steps required by our algorithm is also shown to be within a constant times the lower bound given by the number of neighbors of each node.**

## I. Introduction

Voronoi partitioning has been used as a popular technique to partition the space in which a set of robots or sensor nodes are deployed. Different domains where Voronoi partitioning and its variants have been used include sensor coverage [1], [2], [3], [4], [5], [6], [7], search with multiple robots [8], [9], and complete coverage using multiple robots [10]. For a successful distributed implementation of the Voronoi partitioning problem, each robot should be able to compute the corresponding Voronoi cell autonomously. In the conventional method to calculate the Voronoi cell, each robot first uses the positions of all other robots in the environment to compute the entire Voronoi partition, and, then, extracts its Voronoi cell. Computing the entire Voronoi partition is not very efficient as each robot unnecessarily calculates the Voronoi cell for every other robot. In most practical situations, robots may not have information about the positions of all other robots in the environment, e.g., when subsets of robots are outside each others' communication range. In such a situation, it is not possible to compute the exact Voronoi cells in a distributed manner. However, there are several multi-robotic applications [1], [2], [8], [9] where each robot is responsible for performing operations within the portion of its Voronoi cell that lies within the range of the sensor(s) it uses to perceive its environment (e.g., camera, laser or IR sensor). Therefore, it makes sense to study the problem of distributed Voronoi partitioning in a sensor range constrained scenario. In this paper we describe a distributed algorithm where each robot in a multi-robot system autonomously computes its corresponding Voronoi cell within such a sensor range constrained scenario. To do this, each robot uses only the positions of the robots that are within a distance of double its sensor range, provided that the robot's communication range is at least double its sensor range. In our proposed technique, each robot represents the information about the positions of robots within its communication range in a polar coordinate system with its own position as reference. We provide theoretical results validating that our algorithm successfully calculates the sensor range constrained Voronoi cell for each robot. The theoretical results are validated with the help of experiments.

This work has been supported as part of the COMRADES project supported by the U.S. DoD Office of Naval Research, grant no. N0000140911174.

K.R Guruprasad is an Assistant Professor at the Department of Mechanical Engineering, National Institute of Technology Karnataka, Surathkal, India. krgprao@gmail.com

Prithviraj Dasgupta is an Associate Professor at the Department of Computer Science, University of Nebraska at Omaha, NE, USA. pdasgupta@mail.unomaha.edu

## II. Related Work

Computation of the complete Voronoi partition is a standard problem addressed in computational geometry [11]. Calculation of the Voronoi partition requires the underlying communication graph of the nodes to be connected. There are only a few existing techniques that employ a distributed computation of the Voronoi cell. In [12], an approximate Voronoi cell is constructed for each node using its four closest nodes, one from each quadrant. If this approximate Voronoi cell is not a polygon, a node located in the unbounded region is chosen randomly to get an approximate Voronoi cell. A filter-and-refine algorithm is presented in [13], where in the first phase, the sensor node computes an approximate Voronoi cell based on the nodes within its radio range. If this cell computed by a node is not bounded and the node is in the interior of the convex hull formed by all the nodes, messages are sent by the node in the direction of points that would close the cell. Once a node's cell is closed, it refines the cell by communicating with other nodes within an impact range. In [14], the authors consider a bounded region and an initial node set as a subset of the entire node set that yields a bounded Voronoi cell. Then a geographic routing protocol called GPSR is used to probe

for nodes that reduce the initial Voronoi cell and refine it. A similar approach is used in [15], where the sensors cooperate to refine the Voronoi cell and achieve a faster convergence. The first phase of these algorithms constructs approximate candidate Voronoi cells based on a small number of nodes while using a 'brute force' approach [13]. Also, the existing algorithms rely on communication protocols to exchange positional information on demand. In contrast our work requires robots to exchange positional information with each other only at the beginning of the algorithm. Cortes et al. [1], [2] presented a distributed algorithm for computation of both exact and range-constrained Voronoi cells, which was based on an algorithm by Cao and Hadjicostis [16]. The robot constructs its Voronoi cell by incrementally increasing its sensing radius. In contrast to most of the existing literature in this area, which compute Voronoi cells based on the position of all other robots/sensor motes (or nodes) in the environment, we address the problem of computing only the portion of the Voronoi cell that lies within each robot's sensor range, without using information about the positions of all other nodes. Although it is possible for each robot to compute its Voronoi cell based on the positions of all other robots if that information is available, and then discard the portions of the cell that are outside its sensor range, this is not a very efficient method for calculating the range constrained Voronoi cell. Further, in our proposed algorithm, every robot represents the relative position of other robots using a polar coordinate system, which enables a more structured construction of the Voronoi cell and reduces the amount of computation. Although, our proposed algorithm constructs the Voronoi cell incrementally by increasing the (pseudo) sensing range as in [1], [2], [16], it differs in this use of relative positions using polar coordinates and in a more systematic and efficient construction of the candidate Voronoi cells.

## III. Problem formulation

Consider $N$ robots in a multi-robot system (MRS) (or multi-agent system, sensor network). Let $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$ be the configuration of the MRS, where $p_i \in \mathbb{R}^2$ is the position of the $i$-th robot. Let $I_N = \{1, 2, \ldots, N\}$ be an index set. By a slight abuse of notation, we use $p_i$ to refer to both the $i$-th robot and its position in the space. Let $\mathcal{P}_i^R = \{p_j | \quad \|p_j - p_i\| < R, j \in I_N \setminus \{p_i\}\} \subseteq \mathcal{P}$. Let $\mathcal{V} = \{V_i | i \in I_N\}$ be the Voronoi partition generated by $\mathcal{P}$ as a node set, with

$$V_i(\mathcal{P}) = \{q | \|p_i - q\| \leq \|p_j - q\|, \forall j \in I_N\}$$

Nodes $i$ and $j$ are considered Voronoi neighbors or neighbors in the Delaunay graph $\mathcal{G}_D(\mathcal{P})$, if the corresponding Voronoi cells $V_i$ and $V_j$ share a common edge. Let $N_D(\mathcal{P}, p_i)$ be the set of neighbors of $p_i$ in $\mathcal{G}_D(\mathcal{P})$.

For $i$-th robot having a sensor with a range of $R$, the range-constrained Voronoi cell (RCVC) is defined as $V_i^R = V_i \cap \bar{B}(p_i, R)$. See Figure 1 for illustration. Two robots $p_i$ and $p_j$ are said to be neighbors in $\mathcal{G}_{\text{LD}}(\mathcal{P}, 2R)$, the $2R$-limited Delaunay graph, if $V_i^R$ and $V_j^R$ share a common edge [1]. Let $N_{LD}(\mathcal{P}, 2R, p_i)$ be the set of neighbors of $p_i$ in $\mathcal{G}_{\text{LD}}(\mathcal{P}, 2R)$.

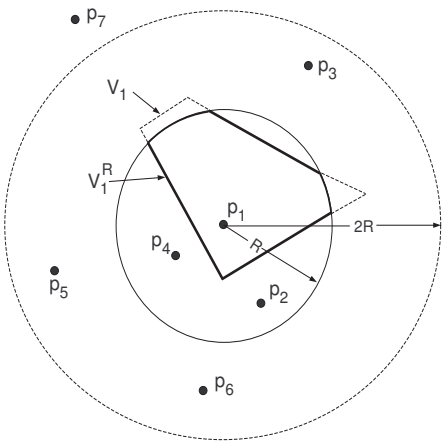



Fig. 1. The range-constrained Voronoi cell $V_1^R$ is shown bounded with dark lines. Node $p_7$ contributes to $V_1$, but not to $V_1^R$.

*Problem statement:* For each $i \in I_N$, given $\mathcal{P}_i^{2R}$, the $i$-th robot should compute the corresponding range-constrained Voronoi cell $V_i^R$, and the neighbors in $\mathcal{G}_{\text{LD}}(\mathcal{P}, 2R)$.

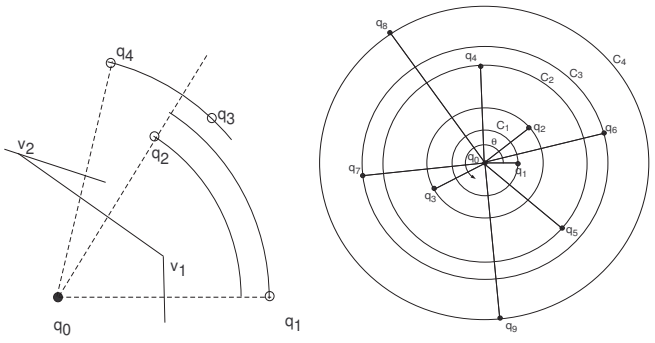
*Lemma 1:* For a given configuration $\mathcal{P}, V_i^R(\mathcal{P}) = V_i^R(\mathcal{P}_i^{2R})$.

*Proof.* Let us assume that $p_j \notin \mathcal{P}_i^{2R}$ and $p_j \in N_D(\mathcal{P}, p_i)$. The perpendicular bisector of line joining $p_i$ and $p_j$ does not intersect $C(p_i, R)$, the circle of radius $R$, centered at $p_i$. Thus, $p_j \notin N_{LD}(\mathcal{P}, 2R, p_i)$. Now as $N_{LD}(\mathcal{P}, 2R, p_i) \subseteq N_D(\mathcal{P}, p_i)$, if a node $p_k \notin \mathcal{P}_i^{2R}$ is such that $p_k \notin N_D(\mathcal{P}, p_i)$, then $p_k \notin N_{LD}(\mathcal{P}, 2R, p_i)$. □

## IV. Computation of range-constrained Voronoi cell

The Voronoi cell $V_i$ and range-constrained Voronoi cell (RCVC) $V_i^R$, of robot located at $p_i$, depend on the positions and orientations of the other robots. While calculating the Voronoi cell of each robot, selecting the set of its Voronoi neighbors based on Euclidean distances with positions of robots represented in Cartesian coordinates might lead to complicated analysis and calculations. For example, in Figure 2(a), robots $q_1$ and $q_2$ are the Voronoi neighbors for robot $q_0$. Robots $q_3$ and $q_4$ are equidistant from $q_0$ but it is easy to see that $q_3$ is not $q_0$'s Voronoi neighbor, while $q_4$ could be a Voronoi neighbor, depending on the relative position and orientation of other robots. In contrast, representing relative robot positions using a polar coordinate system provides a more succinct way to enable the computation of Voronoi cells.

By Lemma 1, the $i$-the robot only needs $\mathcal{P}_i^{2R}$ to compute $V_i^R$. Thus the communication range of robots should be at least $2R$. The $i$-th robot, $p_i$, constructs an ordered set of robots to represent the configuration of robots in $\mathcal{P}_i^{2R}$ in the polar coordinate space with itself as the origin. The set ${}^iQ = \{{}^iq_1, {}^iq_2, \ldots, {}^iq_{N-1}\}$ contains the robots that are within a distance of $2R$ from $p_i$, sorted in order of increasing distances (radii) from $p_i$. Ties in radii are broken by ordering the robots equidistant from $p_i$ in increasing order of angles




(a) (b)

Fig. 2. a) Motivation for using relative configuration in polar coordinate system. b) The robot $p_i$ re-indexes all robots based on the relative position in a polar coordinate system with $p_i = q_0$ as center.

with the line joining $p_i$ and ${}^iq_1$, the closest robot to $p_i$. If more than one robot lie at the shortest distance from $p_i$, then one of these robots is randomly chosen as $q_1$ [1]. For the sake of legibility, we rename robot $p_i$ as $q_0$. An example illustrating the construction of the sets ${}^iQ$ is illustrated in Figure 2(b). With $q_0(= p_i)$ as center, let ${}^iC_1, {}^iC_2, \ldots {}^iC_K$, $K \leq N-1$, denote virtual circles of increasing radii passing through one or more robots in ${}^iQ$; circle ${}^iC_1$ passes through ${}^iq_1$ and so on. Let ${}^ir_k$ be the radius of circle ${}^iC_k$. In the following, when the context is unambiguous, we drop the superscript $i$ to simplify the notation and refer to ${}^iq_j$ as $q_j$, ${}^iC_k$ as $C_k$, and ${}^ir_k$ as $r_k$. For brevity, we use $q_k$ to refer to both the robot itself and its position. Further, by a slight abuse of notation, we use notation $C_k$ to refer to both the virtual circle, and the set of robots on the circle $C_k$.

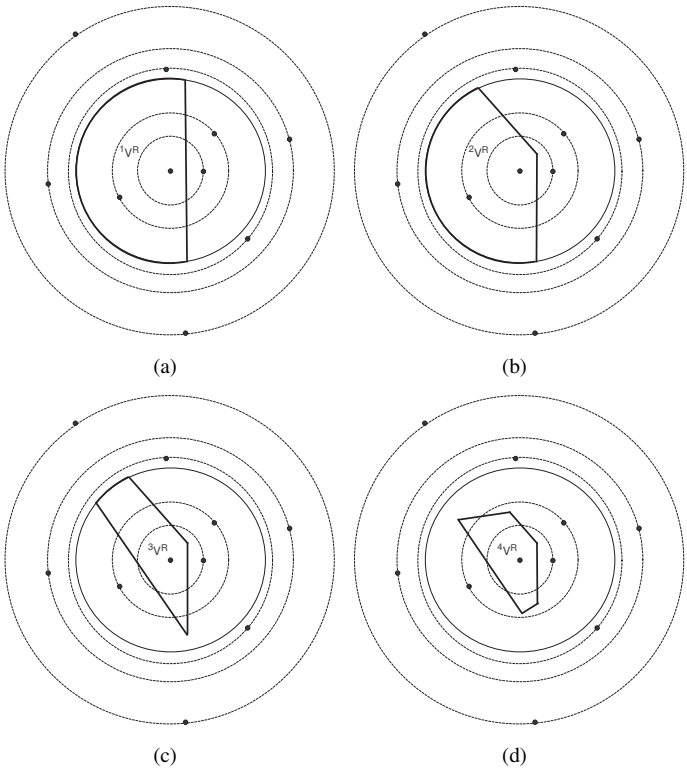
The $i$-th robot has access only to $\bar{B}(p_i, R)$, the closed disc of radius $R$ centered at $p_i$. We denote circle of radius $R$ centered at $p_i$ as ${}^iC^R$ or simply $C^R$. If there are no robots within a distance of $2R$, then $V_i^R = \bar{B}(p_i, R)$. Let $R_i$ be pseudo-communication range. We call $R_i$ as pseudo communication range, as this limit on the range is only used for the purpose of computation, and not the real limitation of the robot. The $i$-th robot computes the Voronoi cell starting with $R_i(1) = r_1$, and expands the communication range incrementally by setting $R_i(k) = r_k$ at $k$-th step, while contracting the range-constrained Voronoi cell. The procedure is discussed formally in the following.

Let $N_k$ be the number of robots on $C_k, k \in I_K$ and $\mathcal{N}_k = N_1 + N_2 + \cdots + N_k$ be the number of robots on or inside $C_k$. Note that in non-degenerate conditions, $N_k \leq 3, \forall k \in I_K$. Let $H(q_0, p)$ be the half plane defined by perpendicular bisector of $q_0 \leftrightarrow p$ containing $q_0$.

When the context is clear, for brevity, we denote $V_i^R$ as $V^R$, and $V^R(\mathcal{P}^{r_k})$ as ${}^kV^R$. The robot starts with $R_i(0) = 0$ and ${}^0V^R = \bar{B}(q_0, R)$. At the $k$-th step,

$$ {}^kV^R =^{k-1} V^R \cap \{ \bigcup_{p_j \in C_k} H(q_0, q_l)\}, k \in I_K \qquad (1)$$

[1]It is easy to show that all such robots are Voronoi neighbors of $p_i$.




(a) (b) (c) (d)

Fig. 3. Illustration of evolution of the range-constrained Voronoi cell as the pseudo communication range (circle shown with dashed line) is increased.

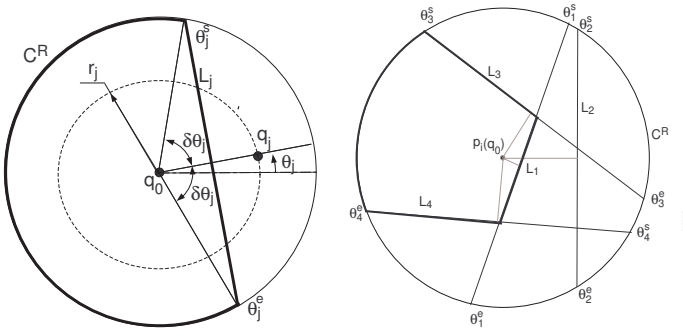
Figure 3 illustrates the evolution of range-constrained Voronoi cell as the pseudo communication range is increased in steps. The computation of the RCVC proceeds in three phases that are described below.

*Phase 1: Chord construction:* The boundary of ${}^kV^R$ is made up of line segments corresponding to perpendicular bisectors and arcs on the circle $C^R$. First we look at the intersection of $b_{0j}$, the perpendicular bisector of line joining $q_0$ and $q_j$ ($j$-th node on $Q$, the relative configuration sorted based on radius) with the circle $C^R$. Let the points of intersection of $b_{0j}$ with $C^R$ be $(R, \theta_j^s)$ and $(R, \theta_j^e)$. Since $R$ is known, we need to find only $\theta_j^s$ and $\theta_j^e$, which are given by:

$$\begin{aligned} \theta_j^s &= \mod_{2\pi}(\theta_j + \delta\theta_j) \\ \theta_j^e &= \mod_{2\pi}(\theta_j - \delta\theta_j) \end{aligned} \qquad (2)$$

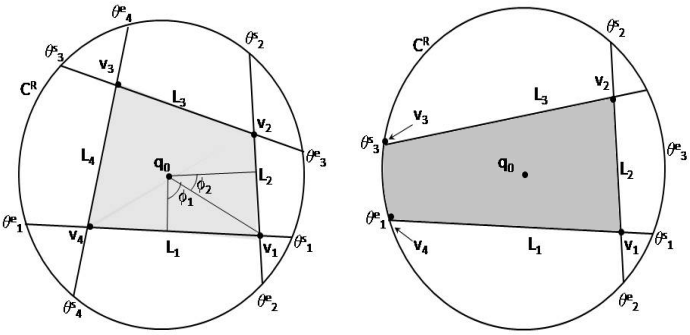
where, $\delta\theta_j = cos^{-1}\left(\frac{r_j}{2R}\right)$. This is illustrated in Figure 4(a). Let $L_j$ denote the segment of $b_{0j}$ that lies between $(R, \theta_j^s)$ and $(R, \theta_j^e)$. Note that $L_j$ forms a chord within $C^R$ corresponding to $q_j$. A segment of $L_j \in \partial V^R$ if and only if $q_j \in N_{LD}(\mathcal{P}, 2R, p_i)$, where $\partial V^R$ represents the boundary of $V^R$. Further note that, if $p_j \in \mathcal{P}_i^{2R} \cap N_D(\mathcal{P}, p_i)$, then $q_j \in N_{LD}(\mathcal{P}, 2R, p_i)$. Starting with $C_1$, the $i$-th robot computing $V_i^R$ constructs the chords corresponding to robots in $Q$ in successive virtual circles and stores them in a set denoted by $\mathcal{L}$. The cumulative set of chords $\mathcal{L}$ at the $k$-th virtual circle $C_k$ is the given by:

$$\mathcal{L} = \mathcal{L} \cup_{j \in \{l | q_l \in C_k\}} L_j.$$




(a) (b)

Fig. 4. a) Intersection of the perpendicular bisector of $(q_0, q_j)$ and the circle $C^R$ gives rise to the chord $L_j$ and the half-plane induced by $q_j$ containing $q_0$. b) The chord $L_2$ is removed by CheckChords method, as it does not contribute to $V^R$.




(a) (b)

Fig. 5. a) $V^R$ can be calculated by finding the vertices corresponding to the intersection points of consecutive chords returned by *checkChords* b) A scenario where two consecutive chords $L_3$ and $L_1$ do not intersect within $C^R = \bar{B}(p_i, R)$.

The *constructChords* method shown in Algorithm 1 is used to constructing chords. It takes $Q$, the set of robots that are in $\mathcal{P}_i^{2R}$ as input and returns the angle-sorted set chords $\mathcal{L}$ corresponding to each $q_j \in Q$. It also constructs a sorted set $\Theta$ containing the polar angles of the endpoints of every chord in $\mathcal{L}$, which is used in the next phase of the construction of the RCVC .

*Phase 2: Removing Redundant Chords:* Let $L_j, L_k \in \mathcal{L}$ denote two chords corresponding to points $q_j$ and $q_k$ respectively. If $L_k$ lies entirely within the half-plane of $L_j$ that does not contain $p_i$, then $L_k \notin \partial V^R$. As illustrated in Figure 4(b), $L_2 (= L_k)$ is entirely within the half-plane of $L_1 (= L_j)$ that does not contain $p_i$. In such a case, $L_k$ is redundant for the construction of $V^R$ and can be omitted from further calculations related to $V^R$ by removing it from $\mathcal{L}$. The *checkChords* method shown in Algorithm 2 is used to remove redundant chords in $\mathcal{L}$. To do this, for every chord $L_j \in \mathcal{L}$, it inspects the set of endpoints of other chords lying in the half-plane of $L_j$ (on the circumference of $C^R$) not containing $p_i$. If the set of other chords' endpoints contains both extremities of any chord $L_m$, then $L_m$ is a redundant chord and it is removed from $\mathcal{L}$. The *checkChords* method removes all such redundant chords and returns a reduced set of chords for the next phase of the RCVC computation.

**constructChords**($Q$)
**Input**: $Q$// $Q$ the relative configuration of robots in $\mathcal{P}_i^{2R}$ in polar coordinates with $p_i$ as reference and sorted based on radii.
**Output**: $\mathcal{L}, \Theta$, // $\mathcal{L}$: set of chords corresponding to nodes in $Q$, $\Theta$: set of endpoints of chords in $\mathcal{L}$ within $C^R$
$\mathcal{L} \leftarrow \emptyset; \Theta \leftarrow \emptyset$;
**foreach** $q_j \in Q$ **do**
  $L_j \leftarrow$ Perp. bisector of line $(q_0, q_j) : q_j \in Q$;
  $(\theta_j^s, \theta_j^e) \leftarrow$ Angles of extremities of chord $L_j$ with $q_0$ within circle $C(q_0, R)$ of radius $R$;
  $\mathcal{L} \leftarrow \mathcal{L} \cup L_j$;
  $\Theta \leftarrow \Theta \cup \{\theta_j^s, \theta_j^e\}$
**end**
Sort $\mathcal{L}$ in ascending order of angle of corresponding nodes;
Sort $\Theta$ in ascending order;
**return** $\mathcal{L}, \Theta$;

**Algorithm 1:** Method for constructing chords for robots (nodes) that are within twice the sensor range of robot $i$.

*Phase 3: Vertex computation and refinement:* The final step in finding the RCVC $V^R$ for robot $p_i$ is to determine the vertex set corresponding to the Voronoi cell of robot $p_i$. Recall that the set of chords returned by *checkChords* is ordered by the polar angles of the chords' corresponding $q_j$ node. Therefore, it seems intuitive that $V^R$ can be computed by finding the intersection point of consecutive chords while cycling through once over the angle-ordered set of chords returned by the *checkChords* method. Figure 5(a) shows a scenario where four chords $L_1$ through $L_4$ are returned by *checkChords* method and the vertices of $V^R$ are computed by finding the intersection points of consecutive pairs of chords. For two consecutive chords, $L_j$ and $L_k$, that intersect at a point, we refer to $L_j$ as the preceding chord and $L_k$ as the succeeding chord. $L_j$ and $L_k$'s intersection point $v_j = (r(v_j), \theta(v_j))$ (in polar coordinates) can be computed as follows:

$$\begin{array}{lcl} \phi_j & = & \text{atan2}(\cos(\theta_k - \theta_j) - \frac{r_k}{r_j}, \sin(\theta_k - \theta_j)) \\ \theta(v_j) & = & \theta_j + \phi_j \\ r(v_j) & = & \frac{r_j}{2\cos(\phi_j)} \end{array} \quad (3)$$

The vertex at the intersection point of two chords (lines) is indexed with a marker called 'line'. The pseudo-code for finding the vertices of $V^R$ is given in the *constructVor* method shown in Algorithm 3.

However, there are two wrinkles to this method that make finding the RCVC a bit involved. First, two consecutive chords might intersect outside the sensor range $R$ of robot $p_i$ as shown in Figure 5(b). It is easy to detect such a scenario by inspecting the pair of extremities of two consecutive chords - if neither of the extremities of the succeeding chord lie within the half-

plane of the preceding chord not containing $p_i$, the chords do not intersect within $C^R$. For example, in Figure 5(b), neither of $L_1$'s extremities lie in the half-plane of $L_3$ not containing $p_i$. In this case, the vertices of the Voronoi cell generated by these two chords are calculated as the intersection points of the chords with $C^R$ as shown in Figure 5(b). These vertices are indexed with a marker called 'circ' to distinguish them from vertices at which two lines intersect.

The second wrinkle while finding chord intersection points arises from the fact that two chords that are not consecutive might intersect with each other and give rise to a self-intersecting polygon. As shown in Figure 6(b), $L_k, L_j$ and $L_l$ are three consecutive chords. The intersection point of $(L_k, L_j)$ is at $v_1$ and the intersection point of $(L_j, L_l)$ is at $v_2$. However, $v_1$ and $v_2$ give rise to a self-intersecting polygon. In this case, $v_1$ and $v_2$ need to be discarded along with $L_j$ and the intersection of $L_k$ and $L_l$ has to be computed as a valid vertex of $V^R$. To check for the self-intersecting polygon case, a newly formed vertex is checked with the preceding vertex for acceptability. Let $L_k, L_j, L_l$ be three consecutive angle sorted chords. Let $v_1$ be the point of intersection of $L_k$ and $L_j$, and $v_2$, that of $L_j$ and $L_l$. For two polar angles $a, b \in [0, 2\pi]$, define $\Delta(a, b)$ as the following function:

$$\Delta(a,b) = \begin{cases} \mid a-b \mid & \text{if } \mid a-b \mid \leq \pi \\ 2\pi - \mid a-b \mid & \text{if } \mid a-b \mid > \pi \end{cases}$$

Now let $\Delta\theta_1 = \Delta(\theta(v_1), \theta_j^e)$ and $\Delta\theta_2 = \Delta(\theta(v_2), \theta_j^e)$. The vertex $v_2$ is accepted with respect to $v_1$ only if $\Delta\theta_2 > \Delta\theta_1$, as shown in Figure 6(a). If $\Delta\theta_2 < \Delta\theta_1$ the vertex $v_2$ is not accepted, as in Figure 6(b). In this case, the chord $L_j$ is discarded as it does not contribute to $V^R$. As a final note, the chord corresponding to robot $q_1$ that is closest to $p_i$ must be part of $V^R$. Therefore, it makes sense to start checking chords from the chord corresponding to robot $q_1$ as that chord can never get removed while remedying an instance of the self-intersecting polygon problem.

The algorithm for constructing the RCVC of $p_i$ is called *RangeConstrainedVor*. It first calculates the set of robots $Q$ that are in $\mathcal{P}_i^{2R}$. It then calls the *constructChords(), checkChords()* and *constructVor()* methods sequentially to calculate $V_i^R$. Every robot $p_i \in \mathcal{P}$ uses the *RangeConstrainedVor* algorithm independently to calculate its Voronoi cell in a distributed manner.

*Lemma 2:* Let $\mathcal{L}_1$ be the set of chords at the end *checkChords* method.

$$N_{LD}(\mathcal{P}, 2R, p_i) = N_{LD}(\mathcal{P}_i^{2R}, 2R, p_i) \subseteq \{q_j | L_j \in \mathcal{L}_1\}$$

*Proof.* Let $\mathcal{L}$ be the chord set before *checkChords* method, which contains chords corresponding to all $p_j \in \mathcal{P}_i^{2R}$. A chord $L_j$ is removed from $\mathcal{L}$ by the *checkChords* method, only when $\exists L_k \in \mathcal{L}$, s.t. both end points of $L_j$ represented by $\theta_j^s$ and $\theta_j^e$, lie in the range $(\theta_k^e, \theta_k^s)$, which is the portion of half plane corresponding to $L_k$, not containing $p_i$, within $C^R$. Thus, $L_j$ lies outside $V^R$ and hence does not contribute to $V^R$ and hence corresponding node $q_j \notin N_{LD}(\mathcal{P}, 2R, p_i)$. □

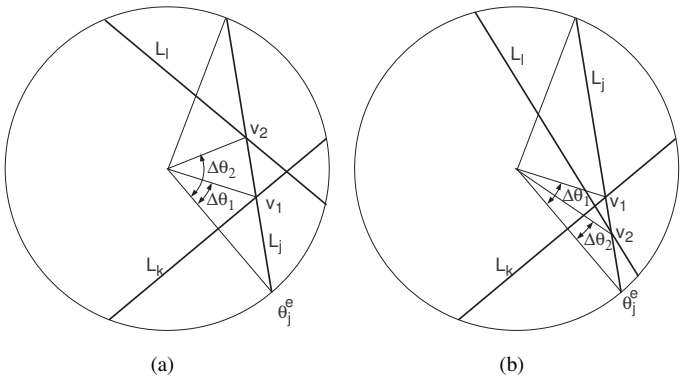



Fig. 6. Illustration of condition for (a) accepting and (b) discarding a vertex formed by intersection of two successive chords in $\mathcal{L}$.

**checkChords**$(\Theta, \mathcal{L})$
**Input**: $\Theta, \mathcal{L}$; // $\mathcal{L}$: angle-sorted list of chords in circle $C^R$, $\Theta$: sorted list of endpoints of chords in $\mathcal{L}$
**Output**: $\mathcal{L}, \Theta$// refined set of chords and chord endpoints
**foreach** $(\theta_j^s, \theta_j^e) \in \Theta$ **do**
  $\Theta_{diff} \leftarrow$ ordered set of endpoints of chords lying between $(\theta_j^e, \theta_j^s)$ in $\Theta$;
  **if** $\exists m$ *s.t.* $\theta_m^s \wedge \theta_m^e \in \Theta_{diff}$ **then**
    $\mathcal{L} \leftarrow \mathcal{L} \setminus \{L_m\}$;
    $\Theta \leftarrow \Theta \setminus \{\theta_m^s, \theta_m^e\}$;
  **end**
**end**
**return** $\mathcal{L}, \Theta$;

**Algorithm 2:** Method used to remove non-intersecting chords which are not contributing to $V^R$.

*Lemma 3:* Let $\mathcal{L}_2$ be the set of chords remaining at the end of *constructVor* method.

$$N_{LD}(p_i, R, \mathcal{P}) = N_{LD}(p_i, R, \mathcal{P}_i^{2R}) = \{q_j | L_j \in \mathcal{L}_2\}$$

*Proof.* $L_j \in \mathcal{L}_1$ is removed from the chord list in *constructVor* method only if the point of intersection of $L_j$ and $L_{j.\text{next()}}$ is on the portion of $L_j$ which is in the half plane corresponding to $L_{j.\text{prev()}}$, the chord preceding $L_j$, not containing $p_i$, thus lying outside $V^R$ (see Figure 6). In such a situation, as illustrated in Figure 6(b), $L_j \cap H(p_0, q_{j.\text{next()}}) \cap H(p_0, p_{j.\text{prev()}}) \cap C^R = \emptyset$, and hence $L_j$ does not contribute to $V^R$. Thus, if $L_j$ is removed from the chord list, it does not contribute to $V^R$, and hence $q_j \notin N_{LD}(p_i, R, \mathcal{P})$.

Now let $q_j \notin N_{LD}(\mathcal{P}, 2R, p_i)$. If $L_j \notin \mathcal{L}_1$, then $L_j \notin \mathcal{L}_2$. Now, if $L_j \in \mathcal{L}_1, \exists L_k \in \mathcal{L}_2$ s.t, chords $L_k$ and $L_j$ intersect. It is clear that if no other chords in $\mathcal{L}_2$ intersect $L_k$, then $q_j \in N_{LD}(\mathcal{P}, 2R, p_i)$, hence, there exists at least one chord, say $L_m$, which intersects $L_k$. Further, if there is no chord $L_m$ s.t. it intersects both $L_k$ and $L_j$, then $q_j \in N_{LD}(p_i, R, \mathcal{P})$. Hence, $\exists L_m$ which intersects both $L_k$ and $L_m$. Now, if the vertex formed by $L_k$ and $L_j$, $v(L_k, L_j) \in H(p_0, p_k) \cap H(p_0, p_m) \cap C^R$, then $q_j \in N_{LD}(\mathcal{P}, 2R, p_i)$. Hence, $v(L_k, L_j) \notin H(p_0, p_k) \cap H(p_0, p_m) \cap C^R$, leading to the condition in *constructVor*, which removes the chord $L_j$

**ConstructVor**($\mathcal{L}, \Theta$)
**Input**: $\mathcal{L}, \Theta$; $\mathcal{L}$: refined set of chords (angle-sorted) returned by *checkChords*, $\Theta$: angle-sorted extremities of chords in $\mathcal{L}$
**Output**: $V^R$;// ordered set of vertices of RCVC
$V^R \leftarrow \emptyset$; // Initialize $V^R$ as $\bar{B}(q_0, R)$.
**if** $\mathcal{L} \neq \emptyset$ **then**
  $\Delta\theta_1 \leftarrow 0$; $\Delta\theta_2 \leftarrow 0$; $j \leftarrow 1$;
  **while** $(j \neq 1)$ *or* $(|V^R| == 0)$ **do**
    **if** $\theta^e_{j.next()}$ *lies between* $\theta^e_j$ *and* $\theta^s_j$ **then**
      // consecutive chords intersect
      // next() gives the index of the next chord in $\mathcal{L}$ with wrap around
      $v \leftarrow$ findIntersection$(L_j, L_{j.\text{next}()})$;
      **if** $|(V^R)| \geq 2$ **then**
        $\Delta\theta_1 = \Delta(\theta(v.\text{prev}()) - \theta^e_j)$;
        // $v$.prev is the previous vertex in $V^R$
        $\Delta\theta_2 = \Delta(\theta(v) - \theta^e_j)$;
        **if** $\Delta\theta_2 > \Delta\theta_1$ **then**
          // not self-intersecting polygon, calculate vertex
          $V^R \leftarrow V^R \cup (v, \text{'line'})$;
          $j \leftarrow j.\text{next}()$;
        **end**
        **else**
          // self-intersecting polygon case // remove $L_j, \theta^e_j, \theta^s_j$
          $\mathcal{L} \leftarrow \mathcal{L} \setminus \{L_j\}$; $\Theta \leftarrow \Theta \setminus \{\theta^e_j, \theta^s_j\}$;
        **end**
      **end**
    **end**
    **else**
      // chords do not intersect each other
      $v_1 = (R, \theta^s_j)$;
      $v_2 = (R, \theta^e_{j.\text{next}()})$;
      $V^R \leftarrow V^R \cup \{(v_1, \text{'circ'}), (v_2, \text{'circ'})\}$;
      $j \leftarrow j.\text{next}()$;
    **end**
  **end**
**end**
**return** $V^R$;

**Algorithm 3:** Algorithm used to calculate the vertices of $V^R$.

(see Figure 6(b)). That is, if $q_j \notin N_{LD}(\mathcal{P}, 2R, p_i)$, then the corresponding chord is removed.

Thus, a chord $L_j \in \mathcal{L}_2$, if and only if $q_j \in N_{LD}(\mathcal{P}, 2R, p_i)$. Thus, $\{q_j | L_j \in \mathcal{L}_2\} = N_{LD}(\mathcal{P}, 2R, p_i)$. □

*Theorem 1:* The region created by the *RangeConstrainedVor* algorithm is $V^R$, the range-constrained Voronoi cell.

*Proof.* Follows from Lemmas 2 and 3. □

*Theorem 2:* The algorithm **R**angeConstrainedVor terminates in finite time.

*Proof.* The proof is trivial as there are finite number of operations for each virtual circle, and there are a finite number of virtual circles. □

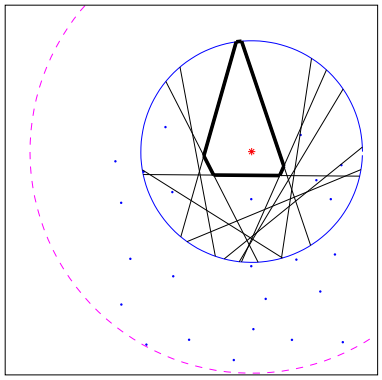


Fig. 7. Range constrained Voronoi cell of point shown with '*', with a sensor range of 75 units, computed using the proposed RangeConstrainedVor algorithm. The positions of other robots are shown with dots.

## V. Computational Complexity

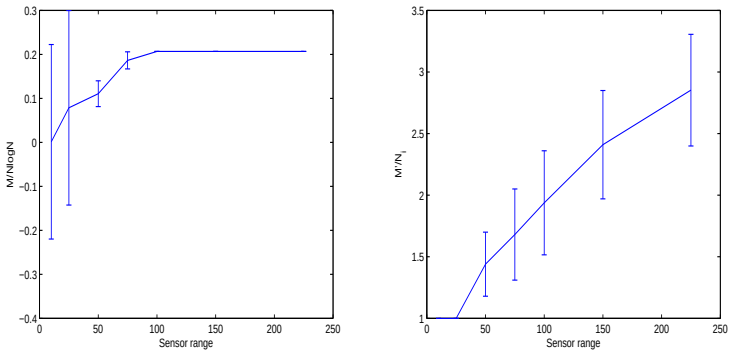
Computational complexity of the any Voronoi partition computation is $\Omega(N \log N)$, where $N$ is the number of nodes (robots) [11]. Thus, the lower bound of computing $V^R_i$ by computing Voronoi partition is $\Omega(N \log N)$. In contrast, we compute $V^R_i$ directly without computing Voronoi partition. It is easy to show that the complexity of computing $V^R_i$ using the *RangeConstrainedVor* algorithm presented in this work is $O(M)$, where $M = |\mathcal{P}^{2R}_i| \leq N$. Also, lower bound on construction of $V^R_i$ using any algorithm is $\Omega(N_i)$, provided the $N_{LD}(p_i, R, \mathcal{P})$ is known. Complexity of *constructVor* is $O(M')$, where $M' = |\mathcal{L}_1|$, number of chords remaining after the *chckChords* method, and $M' \leq M \leq N$.

## VI. Results and Discussions

The proposed *RangeConstrainedVor* algorithm was implemented using C++. Figure 7 shows a range constrained Voronoi cell with $R = 75$ units. The Voronoi cell is marked with darker border, while the chords remaining after *checkChords* method are shown with thin lines. Part of the circle of radius $2R$ centered at $p_i$ (marked with '*') is shown in dashed lines.

We performed simple experiments to compare average complexity with standard Voronoi partition algorithms ($O(N \log N)$ [11] and the lower bound of constructing $V^R_i$ by any algorithm, that is $\Omega(N_i)$. We considered 25 randomly distributed robots and increased the sensor range $R$ from 10 to 225. For each $R$, 25 constrained Voronoi cells corresponding to location of 25 robots are computed.

The quantity $M(p_i)/(N \log N)$, where $M(p_i) = |\mathcal{P}^{2R}_i|$ represents the ratio of computational complexity of proposed *RangeConstrainedVor* algorithm w.r.t. the conventional algorithms computing Voronoi partition (represented by complexity $O(N \log N)$). Figure VI shows the variation of $M(p_i)/(N \log N)$, averaged over 25 nodes, with sensor range. It can be observed that at low sensor range, the computational complexity is much lower, which increases with the sensor range ($M$ increases with increase in $R$). The average value of $M(p_i)/(N \log N)$ reaches a maximum value of about 0.2




(a) (b)

Fig. 8. a) Computational complexity of *RangeConstrainedVor* algorithm to compute the range constrained Voronoi cell in comparison to $N \log N$, the computational complexity of conventional algorithms computing the Voronoi partition. b) Performance of the *constructVor* method compared with the lower bound of $\Omega(N_i)$ of construction $V^R$, where $N_i = |N_{LD}(\mathcal{P}, 2R, p_i)|$.

at higher sensor range, indicating, even at large sensor range, when it is possible that $V_i^R = V_i$, the *RangeConstrainedVor* algorithm is substantially efficient computationally compared to algorithms computing the entire Voronoi partition. Further, the maximum value of $M(p_i)/(N \log N)$ is less than 0.3.

The quantity $M'/N_i$, where $M' = |\mathcal{L}_1|$ and $N_i = |N_{LD}(p_i, R, \mathcal{P})|$, provides a comparison of the computational complexity of the *constructVor* method with the lower bound $\Omega(N_i)$ (when $N_{LD}(p_i, R, \mathcal{P})$ is known, and $V^R$ needs to be computed). Figure VI shows the variation of $M'/N_i$, averaged over 25 nodes, with the sensor range $R$. The maximum average value is below 3, indicating that on an average, *constructVor* has to handle at most three times more number of chord intersection than $N_i$. The maximum value of $M'/N_i$ is below 3.5.

## VII. Conclusions

We proposed a truly distributed algorithm to compute the range-constrained Voronoi cell for multi-robot/multi-agent system and sensor-network applications. Each robot uses a relative configuration of other robots within its sensor range, in polar coordinate system. The pseudo-sensor range is incremented in steps while the range-constrained Voronoi cell gets contracted, finally to obtain the desired region. It was shown that, if the communication range is at least twice as large as the sensor range, the proposed algorithm successfully computes the desired range-constrained Voronoi cell. The theoretical results are validated with the help of experiments to show that for different values of sensor ranges, our proposed algorithm incurs a time complexity that is significantly lower than that of the existing full Voronoi cell computation algorithm. The maximum number of steps required by our algorithm is also shown to be within a constant times the lower bound given by the number of neighbors of each node.

## References

[1] J. Cortés, S. Martínez, and F. Bullo, "Spatially-distributed coverage optimization and control with limited-range interactions," *ESAIM: Control, Optimization and Calculus of Variations*, vol. 11, no. 4, pp. 691–719, 2005.

[2] J. Cortés, S. Martinez, T. Karata, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[3] J. Cortés, "Coverage optimization and spatial load balancing by robotic sensor networks," p. 749 754, 2010.

[4] A.A. Khalek, L. Al-kanj, Z. Dawy, and G. Turkiyyah, "Site placement and site selection algorithms for umts radio planning with quality constraints," in *Proc of IEEE 17th conf on telecommunication*, 2010, pp. 375–381.

[5] A. Kwok and S. Martnez, "Deployment algorithms for a power-constrained mobile sensor network," *International Journal of Robust Nonlinear Control*, no. 7, p. 725842, 2010.

[6] H.-J. Lee, Y.-H. Kim, Y.-H. Han, and C.Y. Park, "Centroid-based movement assisted sensor deployment schemes in wireless sensor networks," in *proc of IEEE Vehicular Technology Conference*, 2009, pp. 1–5.

[7] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Distributed algorithms for environment partitioning in mobile robotic networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1834 –1848, aug. 2011.

[8] K.R. Guruprasad and D. Ghose, "Automated multi-agent search using centroidal voronoi configuration," *IEEE Tans on Automation Science and Engineering*, vol. 8, no. 2, pp. 420–4234, 2011.

[9] ——, "Performance of a class of multi-robot deploy and search strategies based on centroidal voronoi configurations," *International Journal of Systems Science*.

[10] K.R. Guruprasad, Z. Wilson, and P. Dasgupta, "Complete coverage of an initially unknown environment by multiple robots using voronoi partition," in *Proc of 2nd International Conference on Advances in Control and Optimization of Dynamical Systems*, Frebruary 2012.

[11] F. Aurenhammer, "Voronoi diagrams - a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

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

[13] B. Harrington and Y. Huang, "In-network surface simplification for sensor fields," in *Proc of the 13th Annual ACM International Workshop on GIS*, 2005.

[14] B.A. Bash and P.J. Desnoyers, "Exact distributed voronoi cell computation in sensor networks," in *Proc of the Sixth IEEE/ACM Conference On Information Processing in Sensor Networks*, 2007, pp. 236–243.

[15] Y. N. Rodríguez, H. Xiao, K. Islam, and W. Alsailih, "A distributed algorithm for computing voronoi diagram in the unit disk graph model," in *Proc of 20th Canadian Conference in Computational Geometry*, 2008.

[16] M. Cao and C. Hadjicostis, "Distributed algorithms for voronoi diagrams and application in ad-hoc networks," unpublished.