COMMUNICATION AWARE ROBOTIC SENSOR DEPLOYMENT

by

Mohanad Ajina
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Electrical and Computer Engineering

Committee:

	Dr. Cameron Nowzari, Dissertation Director
	Dr. Abbas Zaidi, Committee Member
	Dr. Damoon Soudbakhsh, Committee Member
	Dr. Feitian Zhang, Committee Member
	Dr. Monson H. Hayes, Department Chair
	Dr. Kenneth Ball, Dean, The Volgenau School of Engineering
Date:	Spring Semester 2020 George Mason University Fairfax, VA

Communication Aware Robotic Sensor Deployment

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Mohanad Ajina
Master of Science
George Mason University, 2016
Bachelor of Science
Southern Illinois University, 2014

Director: Cameron Nowzari, Professor
Department of Electrical and Computer Engineering

Spring Semester 2020
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my beloved family (father, mother, wife, siblings and daughter) for all the encouragement, support and fun time I had.

Acknowledgments

I would like to special thank you Dr. Cameron Nowzari and Prof. Abbas Zaidi who made this possible.

Table of Contents

List of Tables

List of Figures

Abstract

COMMUNICATION AWARE ROBOTIC SENSOR DEPLOYMENT

Mohanad Ajina, PhD

George Mason University, 2020

Dissertation Director: Dr. Cameron Nowzari

Wireless sensor networks (WSNs) have been receiving a lot of interest in the last two decades due to the variety of applications that can benefit from them. As the interest in WSNs grows, addressing the physical constraints of these networks has become extremely important to ensure their reliability and efficiency. It was noted in the US Military challenge in 2016 that the wireless networks were already overcrowded and by 2030 the demands on wireless networks will be 250 times greater. Taking this into account, reducing the amount of messages exchanged (communication) between the sensors in the network while maintaining an adequate level of performance has become a critical task, and this will be the focus of this dissertation.

A trivial solution which can reduce the amount of communication over a wireless network is by using a periodic communication model and specifying a long enough communication period. In spite of this, this solution may not ensure the level of performance desired. For a WSN to do its job the controller must have sufficient information to be updated. This requires finding a communication strategy for the sensors to exchange messages and find the conditions for the controllers to be updated. This dissertation addresses this problem for different deployment approaches.

In this dissertation, we explore various existing and new methods to reduce the amount of communication with sufficient autonomy that allows the sensors to determine when information is needed and what kind of information is needed to perform a given task. The first chapter of this dissertation focuses on the virtual force deployment approach, the second chapter revisits the problem of the Voronoi diagram deployment approach to provide a more practical solution, and the third chapter adapts the tools and concepts from chapter 2 to any type of Voronoi diagram.

Chapter 1: An Event-Triggered Virtual Force Algorithm for Multi-Agent Coverage Control with Obstacles

Abstract

In this work, we propose an event-triggered algorithm based on a virtual force deployment approach to address the multi-agent coverage control problem in the presence of obstacles. Unlike most works that consider this problem, we are mainly interested in reducing the amount of communication and motion required by the agents to reach a configuration that increases the coverage throughout an environment of interest. In particular, most works that consider this problem assume agents are in constant communication with each other. Instead, the event-triggered algorithm we propose allows agents to decide for themselves when communication is necessary while still achieving the primary goal of covering the environment and ensuring collisions are avoided. Several simulations illustrate the result of our algorithm with and without the presence of an obstacle and compares it against a similar algorithm that does not consider event-triggered communication.

1.1 Introduction

The topic of wireless sensor networks (WSN) is receiving close attention in many research areas today with applications like sensors monitoring for temperature, traffic, health status, and many others [4–7]. The number of sensors in the WSN varies depending on the application. The WSN can be built with only a few sensors such as home security WSN [8] or several thousand such as environmental monitoring WSN [9]. As the size of the networks and number of possible applications increase, researchers have been working to improve performance of WSN and to reduce their costs. To address these challenges, efforts have been made to maximize sensor area coverage while reducing the amount of communication between sensors. In this paper, we refer to sensors as agents, and to WSN as a multi-agent system. Our work studies the trade-offs of the multi-agent system performing an area coverage control task when the communication between agents is controlled by an event-triggered law. To achieve this goal, we designed an event-triggered law that allows agents to decide autonomously when updated information about neighbors' locations is needed to complete the task. Our motivation comes from the need to reduce the performance cost by reducing unnecessary communication.

Literature review There are two common strategies that have been utilized to address the area coverage control problem. These strategies are computational geometry based and force based [10]. An example of the computational geometry based strategy is a Voronoi diagram or a Voronoi partition. This approach divides an environment into coverage regions based on the distance between agents; more details can be found in [11]. The authors in [12] successfully applied the Voronoi diagram strategy to achieve optimal deployment for mobile sensing networks. However, they assumed that each agent has access to all the agents' locations at all times. In [13, 14], the assumption was removed, and the Voronoi diagram was implemented based on local information provided by the agent's neighbors. In [15], the Voronoi diagram with the theoretical techniques was used to calculate the best and worst coverage cases. The above presented Voronoi diagram work assumed free environment. The authors in [16] proposed extension to the Voronoi diagram strategy that addressed the issue of area coverage with heterogeneous agents, and they generalized their approach for non-convex environment. In [17], the authors developed a new approach for area coverage in the presence of known obstacles of non-convex polygonal environments. In [18], obstacles are unknown, and the Voronoi diagram strategy is applied to maximize area coverage.

The second common strategy is force based, that allows agents to move based on attacked attractive and/or repulsive forces. An early work of the force based strategy is proposed in [19] that ensures collision avoidance. In [20], the authors used a repulsive force algorithm to deploy agents in a building. In [21], a target involved virtual force algorithm is proposed to locate more agents closer to an area of interest and to keep agents far away from obstacles. In [22], a virtual force algorithm (VFA) is proposed to increase the coverage area after initially placing agents randomly in the environment in the presence of obstacles. However, they assumed that each agent has access to all agent's locations at all times. The authors in [23] eliminate this assumption, and they proposed an improved VFA (IVFA) and exponential VFA (VFA). Both algorithms performed better area coverage task than VFA even when agents have limited communication ranges, but on paper, no results considered the presence of obstacles. Recently, in [1], the obstacle avoidance virtual force algorithm (OAVFA) proposed to maximize the area coverage and to minimize the average moving distance. The OAFVA performed better over IVFA and EVFA with and without the presence of an obstacle. However, all the above work assume continuous or periodic communication and/or continuous or periodic sensing among agents.

This brings us to the other area of relevance to this work which are distributed event- and self-triggered coordination strategies. Both types of triggered laws have been proposed to reduce the amount of communication between agents while maintaining some desired system level properties. The authors in [2] propose a self-triggered algorithm to save communication power for the optimal deployment problem without obstacles. In [24], an event-triggered algorithm was able to reduce the amount of communication between agents when performing the multi-agent rendezvous task. In [25], the proposed event-triggered algorithm saved a significant communication power for the average consensus problem. In [26] and [27], the event-triggered control lowered the number of communication between agents for a leader-following consensus problem, and in [28], for a multi-agent systems consensus problem.

Thus, we are interested in combining the distributed triggering strategies with a force based coverage control strategy in order to improve the performance of the system in terms of reduced communication. More specifically, our work builds on the obstacle avoidance virtual force algorithm (OAVFA) proposed in [1], where the agents are always aware of their neighbors' positions. Instead, we are interested in combining this motion control algorithm with an event-triggered communication strategy to reduce communication while achieving the same level of coverage as the OAVFA that requires perfect information at all times.

Statement of contributions The main contribution of our work is the design of the `event-triggered virtual force algorithm` that reduces communication while agents still complete the primary coverage control task. We first design a `motion control law` that allows agents to determine their control inputs based on a virtual force deployment approach. Second, we design a `decision control law` that allows agents to determine when communication with neighbors is needed to complete the task. The `event-triggered virtual force algorithm` combines both laws to allow agents to deploy in the environment with less communication performed. Our algorithm does not require periodic communication as in [1] while still achieving the same level of coverage. Various simulations illustrate the performance of the `event-triggered virtual force algorithm` with and without the presence of obstacles.

1.2 Problem Statement

Consider a network of n agents moving in a rectangle environment $S \in \mathbb{R}^2$ with some static obstacles $O \subset S$. More specifically, we consider N_o distinct obstacles $o_1, \dots, o_{N_o}$ such that $\cup_{m \in \{1,\dots,N_o\}} o_m = O$. We denote $\partial S \subset S$ to be the of set environment's boundaries, and denote the position of agent $i \in \{1, \dots, n\}$ at discrete time $t \in \mathbb{Z}_{\geq 0}$ to be p_t^i. The collection of all agent positions at time t is then given by $P_t = (p_t^1, \dots, p_t^n) \subset S^n$.

We consider a simple kinematic model with bounded velocity $V_{\max}$,

$$p_{t+1}^i = p_t^i + u_t^i,$$

where $\left\| u_t^i \right\| \leq V_{\max} \Delta t$ is the control input of agent i at time t and Δt is the actual time between two discrete time steps.

We assume that the agents are initially unaware of the positions and number of obstacles in the environment S. Instead, the agents are able to sense obstacles up to a distance R_S away. We also assume the agents are only able to communicate with neighbors that are agents R_C distance away, $R_C = 2R_S$.

The goal of the agents is now to reach a configuration to cover as much of the unoccupied environment $S \setminus O$ as possible, while keeping the total moving distance and communication among agents as low as possible. More specifically, we consider a binary disk sensing model (BSM) due to its simplicity and effectiveness in modeling covered area [1,29,30]. Given the position of an agent p_t^i, we say that an arbitrary point in the environment $c \in S$ is *covered* by agent i at time t if it is within the sensing range, i.e., $\left\|p_t^i - c\right\| \leq R_S$. Formally, we define the indicator function

$$\text{BSM}(p_t^i, c) = \begin{cases} 1, & \text{if } \left\|p_t^i - c\right\| \leq R_S \\ 0, & \text{otherwise.} \end{cases}$$

which returns 1 if the point c is covered by the agent at p_t^i, and 0 otherwise. Then, given the vector of all agent positions P_t at some time t, we define the coverage ratio as the ratio between areas of all points in the unoccupied domain $S \setminus O$ that are covered and the entire area,

$$C_t^{\text{Ratio}} = \frac{\int_{S \setminus O} \max_{i \in \{1,\dots,n\}} \text{BSM}(p_t^i, c) dc}{\text{Area}(S \setminus O)}. \tag{1.1}$$

We calculate the moving distance of agent i between its current and last locations, and the average moving distance at time step t, D_t^{Ave} is averaging of all agents' moving distances. The D_t^{Ave} is defined formally:

$$D_t^{\text{Ave}} = \frac{\sum_{i=1}^{n} \left\|(p_t^i - p_{t-1}^i)\right\|}{n} \tag{1.2}$$

More specifically, our goal now is to maximize C_t^{Ratio} in (1.1) while keeping D_t^{Ave} in (1.2) and the amount of communication required by the agents as small as possible. In particular, our work builds on the work of [1], where an algorithm is proposed to solve this problem but requires constant communication among the agents. Our main goal is to relax this assumption to improve efficiency of the network while still achieving good overall performance in terms of the metrics defined above.

1.3 Event-Triggered Algorithm Design

In this section, we design the `event-triggered virtual force algorithm` that allows agents to decide for themselves when communication with neighbors is necessary to complete the global task. The `event-triggered virtual force algorithm` has two components: a `motion control law` that determines the control input of each agent, and a `decision control law` that decides when communication is required.

1.3.1 Motion control law

We first design the `motion control law` that allows agents to determine how to move in the environment based on a virtual force approach. Each agent is exposed to three type of forces: 1. a neighbor force, $\vec{F}_t^{ij}$, that could be an attractive or a repulsive force, 2. an obstacle force, $\vec{F}_t^{i,o_m}$, that is a repulsive force, 3. a boundary force, $\vec{F}_t^{i,b}$, that is a repulsive force. These forces are heavily effected by the distance between an agent and neighbors, obstacles, and boundaries, respectively. We adopt the following force equations from [1], where the authors only consider deployment over a field of uniform density. Modifying the algorithm to deploy over non-uniform fields will be reserved for future work. The neighbor force $\vec{F}_t^{ij}$ is defined as

$$\vec{F}_t^{ij} = \begin{cases} 0, & \text{if } d_t^{ij} > R_C, \\ K_A(d_t^{ij} - d_{ij}^{th})(\frac{p_t^j - p_t^i}{d_t^{ij}}), & \text{if } R_C \geqslant d_t^{ij} > d_{ij}^{th}, \\ 0, & \text{if } d_t^{ij} = d_{ij}^{th}, \\ K_B(d_{ij}^{th} - d_t^{ij})(\frac{p_t^i - p_t^j}{d_t^{ij}}), & \text{if } d_t^{ij} < d_{ij}^{th}, \end{cases}$$

where $d_{i,j}^{th} = \sqrt{3}R_C/2 = \sqrt{3}R_S$, and K_A and K_B are constants. The $d_t^{i,j}$ is a distance between an agent and a *neighbor*. Agent j is a neighbor of agent i if and only if $\left\| p_t^i - p_t^j \right\| \leqslant R_C$. If agent i has a neighbor within its communication range, the agent communicate with the neighbor to collect its current location. When the neighbor's location is received, the agent i calculates the distance to the neighbor $d_t^{i,j}$.

In addition, the obstacle force $\vec{F}_t^{i,o_m}$ is defined as

$$\vec{F}_t^{i,o_m} = \begin{cases} 0, & \text{if } d_t^{i,o_m} \geqslant d_o^{th}, \\ (K_{r1}(d_o^{th} - d_t^{i,o_m}), \alpha_{i,o_m} + \pi), & \text{if } d_t^{i,o_m} < d_o^{th}, \end{cases}$$

where $d_o^{th} = \sqrt{3}R_S/2$, $m \in \{1, \ldots, N_o\}$, and K_{r1} is a constant. The d_t^{i,o_m} is the shortest distance between an agent and an obstacle that is within agent i' sensing range. If agent i senses an obstacle, o_m, the agent calculates the shortest distance to the obstacle d_t^{i,o_m}.

Furthermore, The boundary force $\vec{F}_t^{i,b}$ is defined as

$$\vec{F}_t^{i,b} = \begin{cases} 0, & \text{if } d_t^{i,b} \geqslant d_b^{th}, \\ (K_{r2}(d_b^{th} - d_t^{i,b}), \alpha_{i,b} + \pi), & \text{if } d_t^{i,b} < d_b^{th}, \end{cases}$$

where $d_b^{th} = \sqrt{3}R_S/2$ and K_{r2} is a constant. The $d_t^{i,b}$ is the perpendicular distance between an agent and a boundary that is within the agent's sensing range. Let $b \in \partial S$ to be a point on the environment's boundary, and the closed segment $[p_t^i, b] \subset S$ to be a line such that $[p_t^i, b] \perp \partial S$. If agent i senses an environment's boundary, the agent calculates the perpendicular distance to the boundary. In a rectangle environment, the $\vec{F}_t^{i,b}$ is the total sum of the four boundaries' forces. Formally:

$$\vec{F}_t^{i,b} = \vec{F}_t^{i,bx+} + \vec{F}_t^{i,bx-} + \vec{F}_t^{i,by+} + \vec{F}_t^{i,by-}$$

Therefore, the Net-Force that attacks an agent at time t, $\vec{F}_t^i$, is the sum of all forces. The $\vec{F}_t^i$ defined as:

$$\vec{F}_t^i = \sum_{j \in \mathcal{N}_t^i} \vec{F}_t^{ij} + \sum_{m=1}^{N_0} \vec{F}_t^{i,o_m} + \vec{F}_t^{i,b}, \tag{1.3}$$

where $\mathcal{N}_t^i \subset P_t$ is the set of agent i's neighbors at time t, and N_0 is the number of obstacles in the environment.

The Net-Force of an agent could be 0 if no neighbor is within the agent's communication range and no obstacle and boundary are within its sensing range. Also, $\vec{F}_t^i$ could be weak if the forces are contradictory or neighbors, obstacles and/or boundaries are far from an agent but within its communication and sensing ranges. On the other hand, $\vec{F}_t^i$ could be strong if an agent has very close neighbors, obstacles and/or boundaries.

More specifically, $\vec{F}_t^i$ is the desired displacement of agent i at position p_t^i. Thus, it simply moves with velocity v_{t}^i towards this point by setting

$$u_t^i = \frac{\vec{F}_t^i}{\left\|\vec{F}_t^i\right\|} v_t^i, \tag{1.4}$$

where its velocity v_t^i is given by

$$v_t^i = \min\left(V_{\max}, \alpha \frac{\left\|\vec{F}_t^i\right\|}{\Delta t}\right), \tag{1.5}$$

where $\alpha \in (0, 1)$.

The `motion control law` in short, at every instant of time, each agent calculates its $\vec{F}_t^i$ and moves in the direction as fast as possible if $\left\|\vec{F}_t^i\right\| > V_{\max}\Delta t$. Otherwise, it moves in $\vec{F}_t^i$ direction at slower speed. The simple `motion control law` is described formally in Algorithm 1

Algorithm 1 : `motion control law`

```
Agent $i \in \{1, \dots, n\}$ performs at all times $t \in \mathbb{Z}_{\geq 0}$:
1: receives positions $p_t^j$ from neighbors $j$ within a distance $R_C$
2: senses boundary $\partial S$ and detects obstacles $o_m$ within a distance $R_S$
3: computes $\vec{F}_t^i$ according to (1.3)
4: sets $v_{\mathrm{t}}^i$ according to (1.5)
5: computes $u_t^i$ according to (1.4)
6: computes $p_{t+1}^i = p_t^i + u_t^i$
7: moves to $p_{t+1}^i$ by $v_{\mathrm{t}}^i$
```

1.3.2 Decision control law

We are now interested in improving the `motion control law` which requires all agents to be in communication each time a control signal is computed by relaxing the need for constant communication. Let p_{event}^i be an intermediate goal that agent i can reach in multiple timesteps and define it as $p^i_{\text{event}} = p^i_t + \vec{F}^i_t$. We aim to allow agents to travel towards p^i_{event} without constantly communicating with neighbors by designing the `decision control law` that combines two event-trigger conditions.

A trivial event-trigger condition, $Condition1$, to be that an agent moves to p^i_{event} in multiple Δt without communicating with others. When the agent reaches p^i_{event}, the agent communicates with neighbors to update its F^i_t, p^i_{event} and u^i_t. However, this condition is problematic in some scenarios such as there could an obstacle blocking the way to p^i_{event} in $2\Delta t$.

To avoid collisions, we introduce an additional mechanism to trigger an event. Let $X_i \in \mathbb{R}^2$ be the union of neighbors, obstacles, and boundaries within agent i's sensing range, and define R_T as the triggering radius. Then, given agent i's current position p^i_t at time t, we let a triggering sensing model (TSM) to be

$$\text{TSM}(p^i_t, X_i) = \begin{cases} 1 & \text{if } \exists x \in X_i \text{ s.t. } \left\| p^i_t - x \right\| \leq R_T \\ 0 & \text{otherwise.} \end{cases} \tag{1.6}$$

which returns 1 if agent i detects an object $x \in X_i$ within this triggering range R_T of p^i_t, and 0 otherwise. This triggering sensing model (TSM) does not guarantee no collisions unless R_T is bounded. In our analysis, the worst case scenario is that when two agents are traveling in the opposite direction by $V_{\max}$. The distance that will guarantee no collision between the agents must be more than $2V_{\max}\Delta t$. Therefore, we lower bounded by $R_T > 2V_{\max}\Delta t$. For agents to move without a collision, $Condition2$, they are required to communicate with neighbors to update their F^i_t, p^i_{event} and u^i_t if they sense an object within their triggering sensing ranges. Note, the smaller the R_T, the less communication performed. Thus, we let R_T as small as possible. The `decision control law` combines both conditions and is described formally in Algorithm 2.

Algorithm 2 : `decision control law`

```
Agent $i \in \{1, \ldots, n\}$ performs at every triggered event:
 1: receives positions $p_t^j$ from neighbors $j$ within a distance $R_C$
 2: senses boundary $\partial S$ and detects obstacles $o_m$ within a distance $R_S$
 3: computes $\vec{F}_t^i$ according to (1.3)
 4: computes $p_{\text{event}}^i = p_t^i + \vec{F}_t^i$
 5: set TSM= 0
 6: while ($\|p_t^i - p_{\text{event}}^i\| \neq 0$ & TSM $\neq 1$) do
 7:     computes $p_{t+1}^i$
 8:     move to $p_{t+1}^i$
 9:     set $p_t^i = p_{t+1}^i$
10:     sense all objects $X_i$ within triggering sensing range
11:     compute TSM according to (1.6)
12: end while
```

1.3.3 The `event-triggered virtual force algorithm`

Here, we synthesize the event-triggered strategy that helps agents to determine at each Δt when updated information is needed to complete the task. Our designed algorithm is a combination of `motion control law` of Section 1.3.1 and `decision control law` of Section 1.3.2 with a procedure to acquire communication when the conditions are met.

> *[Informal description]:* Agent i communicates with neighbors to collect their locations, and senses the surrounding to locate obstacles and boundaries that are within its sensing range. When, the attractive and/or repulsive forces are calculated, the agent computes its $\vec{F}_t^i$. When $\vec{F}_t^i$ is computed, the agent calculates p_{event}^i, sets its $v_{\text{t}}^i = \min\left(V_{\max}, \alpha \left\|\vec{F}_t^i\right\| / \Delta t\right)$, and computes its u_t^i. Then, the agent calculates p_{t+1}^i and moves toward it. When p_{t+1}^i is reached, the agent senses for an object within its triggering sensing range and updates TSM. If the TSM returned 0, the agent calculates a new p_{t+1}^i, moves to p_{t+1}^i, senses for an object at p_{t+1}^i until it reaches the p_{event}^i. If p_{event}^i is reached or the TSM returned 1, the agent communicates with neighbors to update its F_t^i, p_{event}^i and u_t^i.

The `event-triggered virtual force`
`algorithm` is described formally in Algorithm 3.

Algorithm 3 : `event-triggered virtual force algorithm`

```
Agent $i \in \{1, \ldots, n\}$ performs at every triggered event:
 1: receives positions $p_t^j$ from neighbors $j$ within a distance $R_C$
 2: senses boundary $\partial S$ and detects obstacles $o_m$ within a distance $R_S$
 3: computes $\vec{F}_t^i$ according to (1.3)
 4: computes $p_{\text{event}}^i = p_t^i + \vec{F}_t^i$
 5: sets $v_{\text{t}}^i$ according to (1.5)
 6: computes $u_t^i$ according to (1.4)
 7: sets TSM= 0
 8: while ($\|p_t^i - p_{\text{event}}^i\| \neq 0$ & TSM $\neq 1$) do
 9:     if $\|u_t^i\| \leq \|p_t^i - p_{\text{event}}^i\|$ then
10:         computes $p_{t+1}^i = p_t^i + u_t^i$
11:         moves to $p_{t+1}^i$ by $v_{\text{t}}^i$
12:         updates $p_t^i = p_{t+1}^i$
13:         sense all objects $X_i$ within triggering sensing range
14:         compute TSM according to (1.6)
15:     else
16:         sets $p_{t+1}^i = p_{\text{event}}^i$
17:         moves to $p_{t+1}^i$ by $v_{\text{t}}^i$
18:         updates $p_t^i = p_{t+1}^i$
19:     end if
20: end while
```

1.4 Simulation

In this section, we provide simulations of the `event-triggered virtual force algorithm`. All simulations are done in MATLAB with $n = 40$ agents moving in $100m$ by $100m$ square environment and 300 time steps. All simulation are averaged over 100 runs, and only few simulations are shown due too limited paper number. The parameters of the simulations are given in Table 1.1.

Table 1.1: Simulation parameter of the `event-triggered virtual force algorithm`

grid size $=100mX100m$	$V_{\max} = 0.5m/s$
$K_A = 0.001$	$K_B = 0.2$
$K_{r1} = 0.8$	$K_{r2} = 0.8$
$R_S = 10m$	$R_C = 20m$
$R_T = 1.1m$	$\alpha = 0.5$

We adopt a communication power model from [31]. Specifically, the total power $\mathcal{P}_i$ used by agent i to communicate, in $dBmW$ power units is defined as:

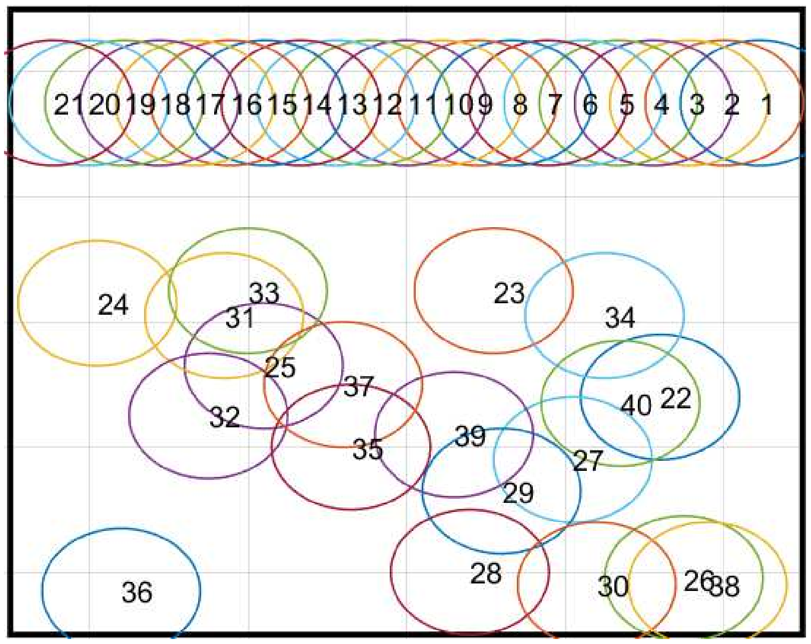
$$\mathcal{P}_i = 10\log_{10}\left[\sum_{j\in\{1,\ldots,n\},i\neq j}^{n} \beta_1 10^{0.1P_{i\rightarrow j}+\beta_2\left\|p^i-p^j\right\|}\right]$$

where β_1 and β_2 are positive real parameters that depend on the characteristics of the wireless medium, and $P_{i\rightarrow j}$ is the power received by agent j of the signal transmitted by agent i. In our simulations, all these values are set to 1.

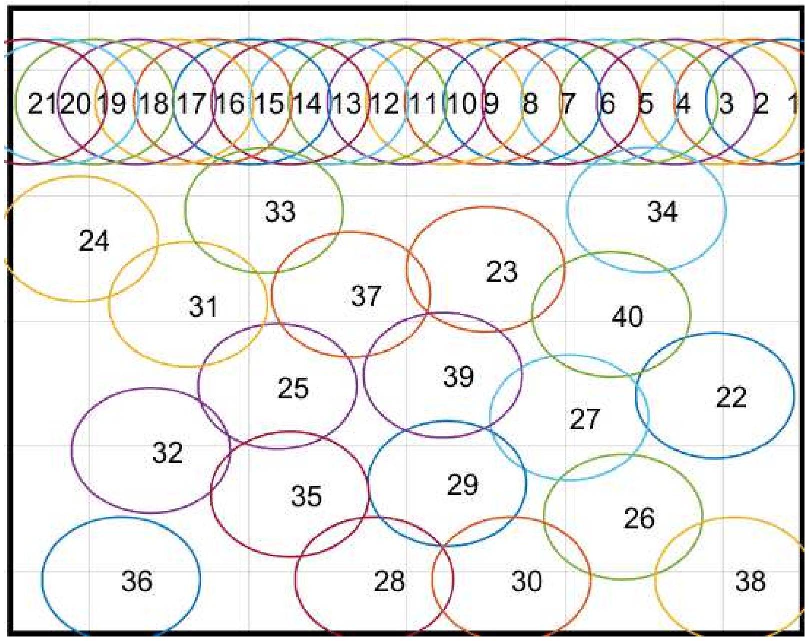
We start by comparing the performance of `event-triggered virtual force algorithm` with OAVFA in [1]. Our analysis on OAVFA showed that the maximum coverage area cannot be always achieved because of a steady state condition. The a steady state condition is that an agent stops updating its control input if it travels less than $0.001m$ for $10\Delta t$. Figure 1.1.(a) and .(b) show agents initialization and the final deployment where OAVFA failed. We removed the steady state condition, and it allows agents to communicate with others and sense the surroundings regardless of the amount they travel to avoid similar scenarios. Figure 1.2 shows that our algorithm maximized coverage area when the OAVFA did not.

The `event-triggered virtual force algorithm` achieves the final deployment as in [1]. Figure 1.3 illustrate the initial and final deployment of the `event-triggered virtual force algorithm` in the presence of an obstacle. Our algorithm achieves maximum coverage area of the environment. This initialization of the agents is based on randomly placing ten agents in each corner of the environment, and note that all further illustrated results assume this initialization.

Figures 1.4.(a) and .(b) compare the C_t^{Ratio} in (1.1) performance of `event-triggered virtual force algorithm` and OAVFA without and with an obstacle. The result shows that both algorithms achieve maximum area coverage, but the OAVFA reaches the maximum slightly faster. Figures 1.5.(a) and .(b) compares the D_t^{Ave} in (1.2) performance



(a)



(b)

Figure 1.1: (a) Initial deployment of 40 agents in the environment when OAVFA fails and (b) final deployment of figure (a) using OAVFA in [1].

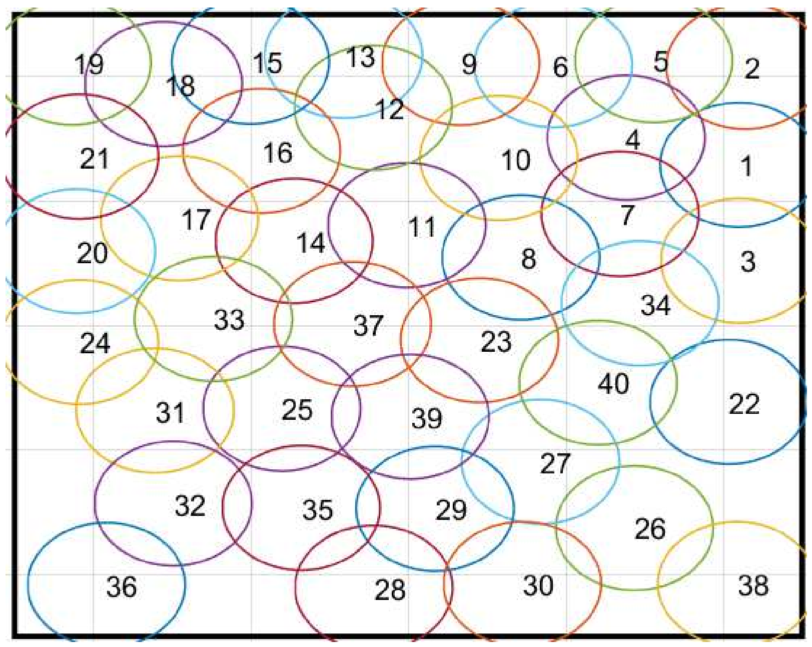
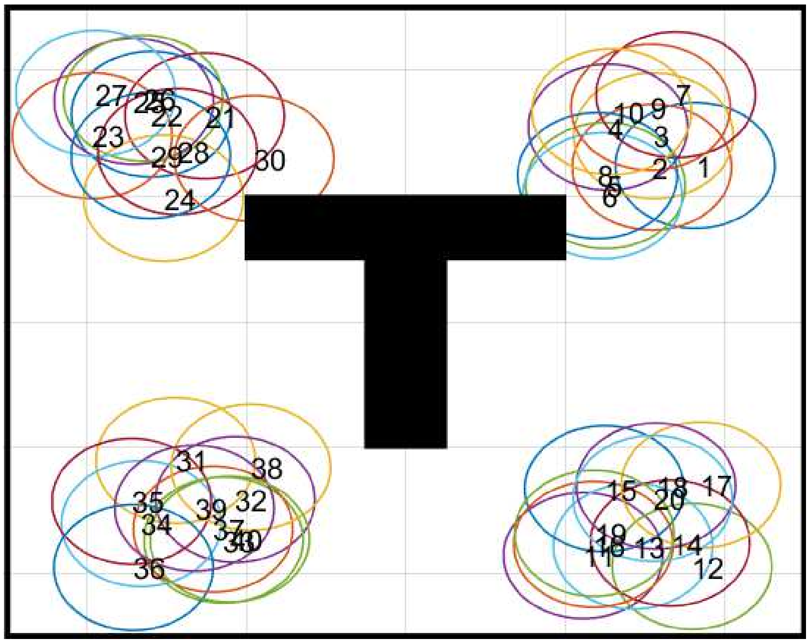


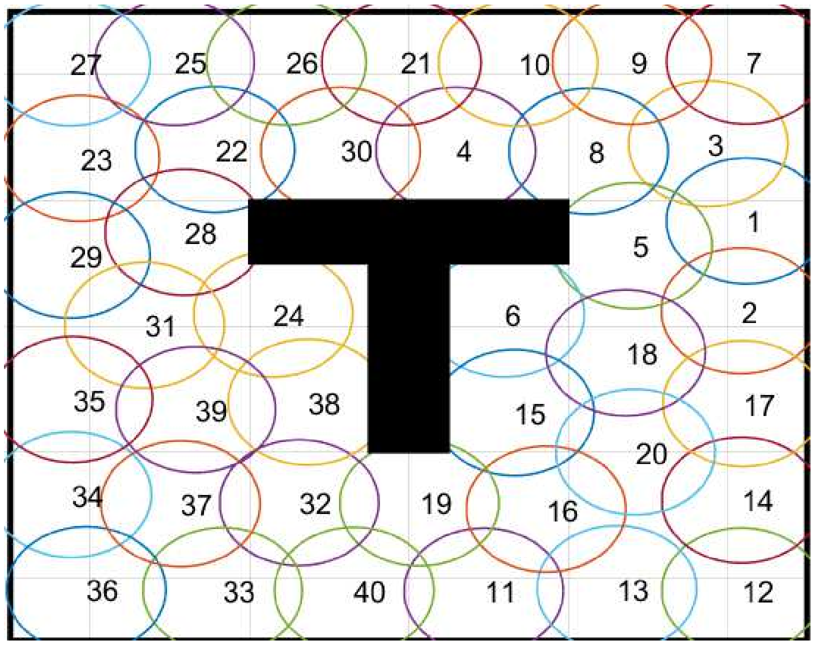
Figure 1.2: Final deployment of Figure 1.1.(a) using the proposed `event- triggered virtual force algorithm.`

without and with an obstacle. The results show that our algorithm has a better performance. In case of no obstacles, our algorithm has slightly better D_t^{Ave} in (1.2), but in case of the presence of an obstacle, our performance minimized D_t^{Ave} in (1.2) more than OAFVA.

Figures 1.6.(a) and .(b) illustrate the average communication power consumed without and with an obstacle. The figures show the contribution of our work. The `event-triggered virtual force algorithm` saved unnecessary communication between agents that resulted in signification reduction of the communication power consumed. The average power saved without obstacles is more than 50%, and the average power saved in the presents of an obstacle is more than 54%.

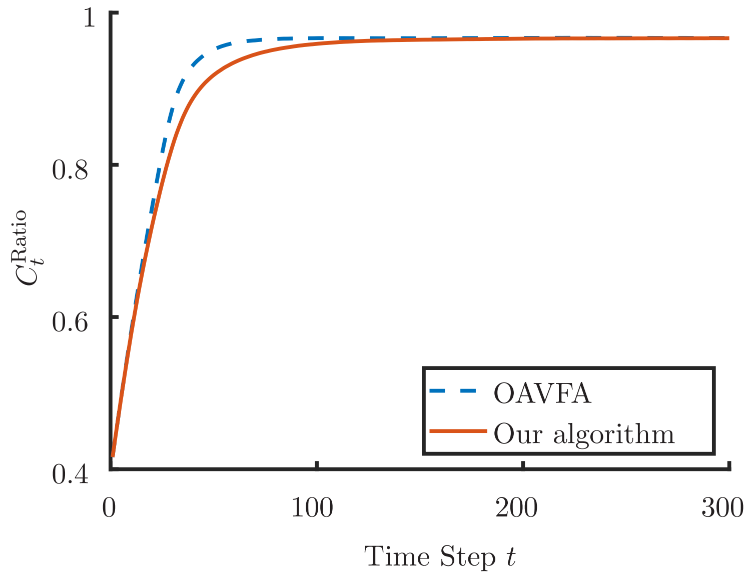


(a)

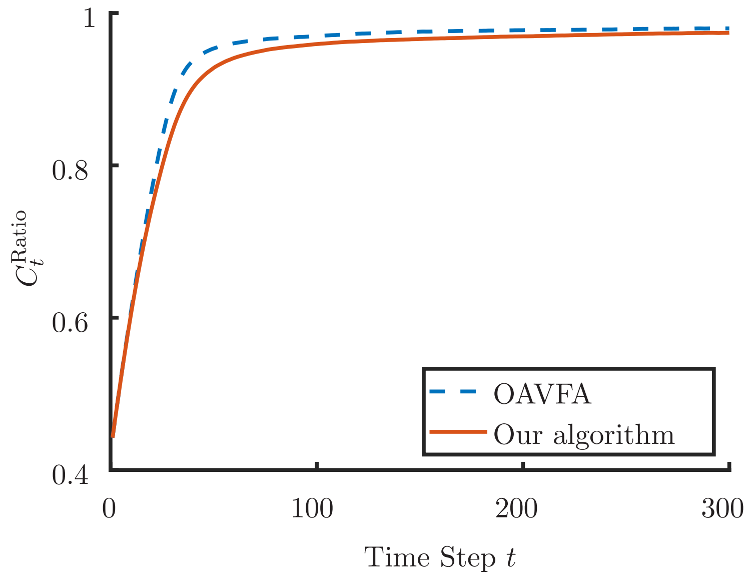


(b)

Figure 1.3: (a) Initial deployment of 40 agents in the environment with an obstacle and (b) final deployment of figure (a) using `event-triggered virtual force algorithm`.

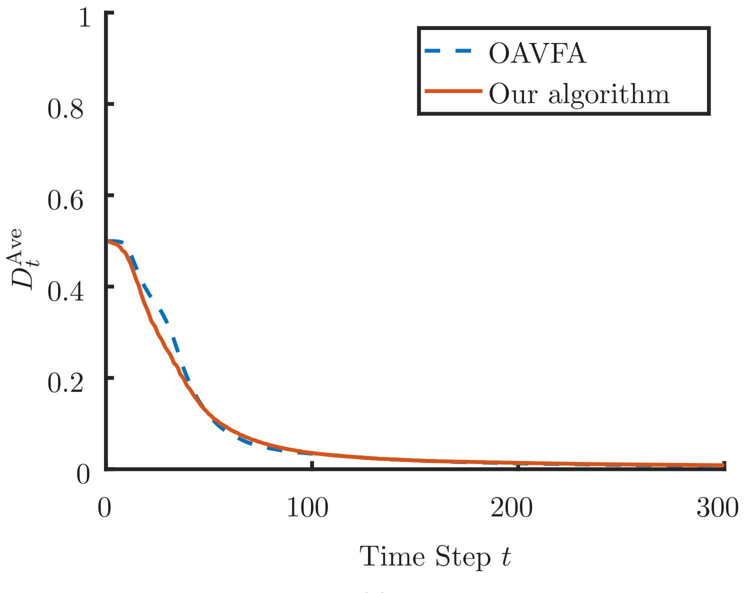


(a)

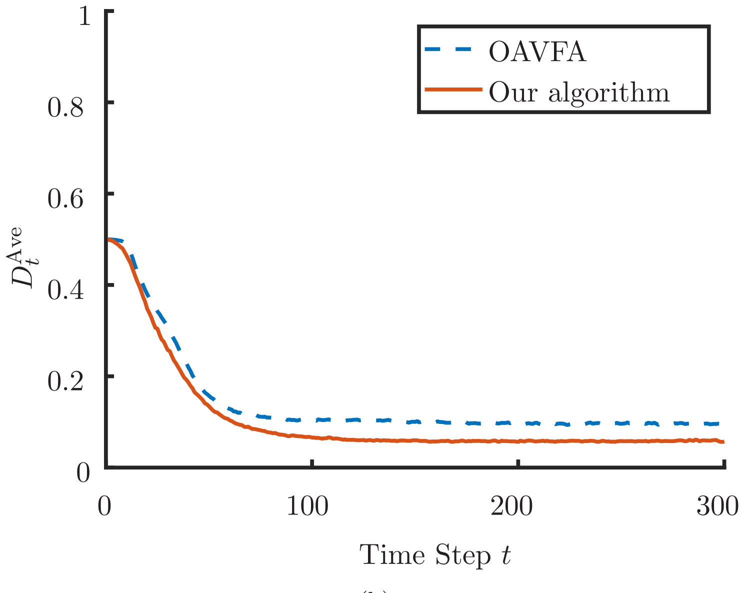


(b)

Figure 1.4: A comparison of the C_t^{Ratio} in (1.1) between `event-triggered virtual force algorithm` and OAVFA in [1] (a) without obstacles and (b) with an obstacle.

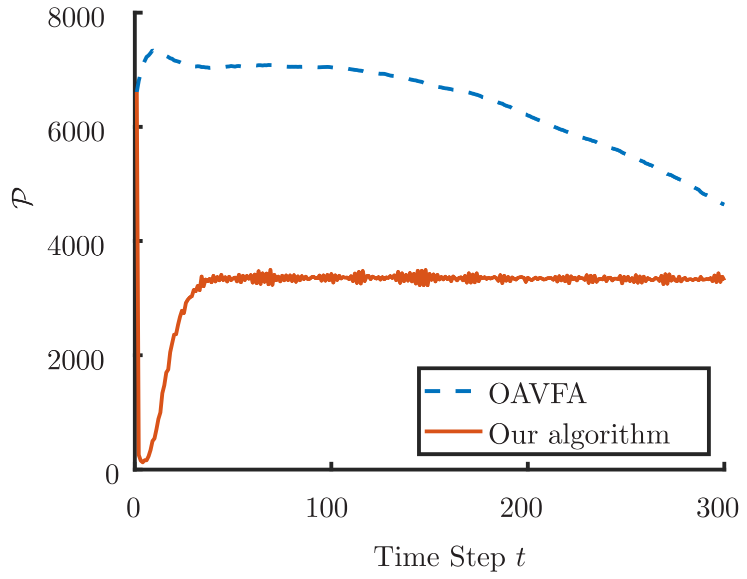


(a)

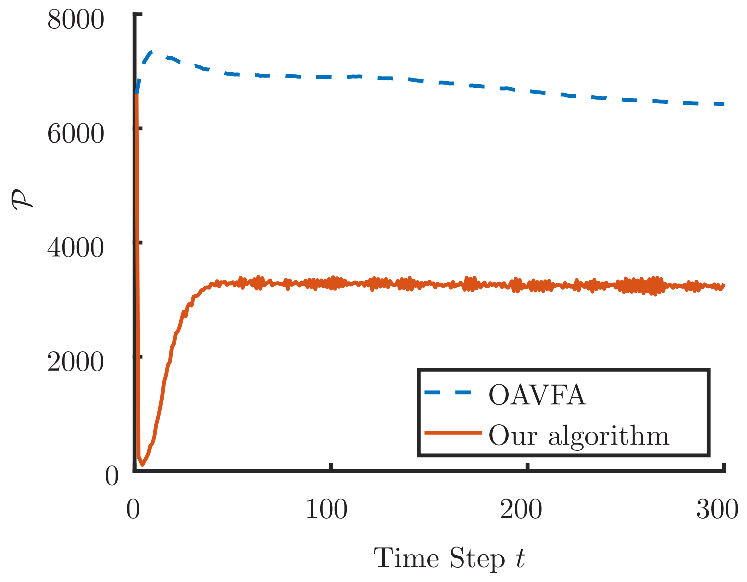


(b)

Figure 1.5: A comparison of the D_t^{Ave} in (1.2) between `event-triggered virtual force algorithm` and OAVFA in [1] (a) without obstacles and (b) with an obstacle.



(a)



(b)

Figure 1.6: A comparison of the the average communication power consumed between `event-triggered virtual force algorithm` and OAVFA in [1] (a) without obstacles and (b) with an obstacle.

1.5 Conclusions

In this paper we considered a multi-agent coverage control problem in the presence of obstacles. To solve the problem, we have proposed the `event-triggered virtual force algorithm` by combining the `motion control law` and the `decision control law` which allows agents to autonomously decide for themselves when communication is required, in addition to how to move. Our algorithm allows agents to travel in the environment without the need to communicate with neighbors at every instant of time. An agent only needs to communicate if it reaches its new location or if it senses an object within the triggering sensing range. The main contribution of this work is reducing the amount of communication between agents while maintaining the desired coverage control performance. Our simulations illustrated that the `event-triggered virtual force algorithm` had reduced more than 50% of communication power compared to the OAVFA in [1] with and without the presence of obstacles, and also achieved maximum area coverage as if agents communicated periodically. Future work will be devoted to include scenarios such as communication delays and package drops with grantees on the level of performance and power saving. In addition, we are interested in developing asynchronous implementation by identifying event trigger condition that ensure the level of performance.

Chapter 2: Asynchronous Distributed Event-Triggered Coordination for Multi-Agent Coverage Control

Abstract

This paper re-visits a multi-agent deployment problem where agents are restricted from requesting information from other agents as well as sending acknowledgments when information is received. These communication constraints relax the assumptions of instantaneous communication and synchronous actions by agents (request and response actions). In this paper, we propose a fully asynchronous communication aware solution to the multi-agent deployment problem that uses an event-triggered broadcasting strategy. Unlike all existing triggered solutions, our event-triggered broadcasting algorithm relies on agents to decide when to broadcast (push) information to others in the network without the need for a response from other agents. In addition, the proposed strategy determines how best to move when up-to-date information is unavailable and cannot be requested. The algorithm is capable of achieving similar levels of performance to that of a continuous or periodic strategy. Our solution is proven to achieve asymptotic convergence and simulation results are provided to demonstrate that the proposed event-triggered broadcasting algorithm can achieve an adequate level of performance under the communication constraints.

2.1 Introduction

Researchers have gained increasing interest in Wireless Sensor Networks (WSNs) due to the variety of applications that can benefit from their exploitation. A WSN is a collection of sensors that have the ability to communicate with one another through a shared wireless spectrum (SWS), and the use of WSNs have been observed in a variety of indoor and outdoor monitoring, tracking and security applications [4–7]. They can be constructed with a small number of sensors, such as in home security applications [8], or can be constructed with several thousand sensors as in environmental monitoring applications [9]. Sensors can be equipped with locomotion capabilities allowing them to reconfigure based on changes to the environment as well as allowing them to be deployed to areas that would otherwise be deemed infeasible for placement. This paper is interested in optimally deploying mobile sensors with the use of a practical communication model that does not require instantaneous communication and synchronous actions by mobile sensors while performing the deployment task.

As the number of sensors in a WSN increases, the amount of communication between sensors increases as well. This rise in communication comes with a cost where more messages that are exchanged between sensors can create a bottleneck over the SWS. This can increase the frequency of packet drops and can create transmission delays in the network. Current data suggests that the SWS is already overcrowded and by 2030 the demands on SWS applications will be 250 times greater than present day [32].

Previous efforts in [2, 33–35] have been made to reduce the amount of communication between sensors while still maintaining an adequate level of performance. This is achieved using event-, self- or team-trigger strategy to communicate with the requirement of instantaneous communication and synchronous actions by mobile sensors. In these works, when a trigger has occurred, an mobile sensor must request information, and the neighboring mobile sensors must respond to that request immediately. We investigate this problem by relaxing the assumption that mobile sensors must communicate instantaneously and take synchronous actions. In other words, the existing works rely on the fact that the mobile sensor can obtain information when it deems necessary. In contrast, our work builds on mobile sensors pushing (broadcasting) information to others, which means that mobile sensors can no longer request information when they desire. Our solution is a fully asynchronous event-based broadcasting communication strategy that further reduces the number of messages exchanged between mobile sensors. For the remainder of this paper, we more generally refer to mobile sensors as agents.

Literature review Similar to previous works that study the multi-agent deployment problem, we make use of *Voronoi partitions* in order for the agents to use distributed gradient descent laws to converge to the set of locally optimal solutions [12]. Also, the authors in [36] utilized Voronoi partitioning techniques to verify whether or not a set of sensors sufficiently covers the environment. The authors assume that the agents always have access to all other agents' locations at all times. In [13, 14], the assumption of always having access to other agent positions is relaxed whereby deployment to the set of critical points is achieved based only on local information provided by an agents' neighbors. However, all of the above-mentioned works assume that continuous or periodic communication occurs between agents. It is the desire of this paper to relax the requirement of continuous or periodic communication without the need for instantaneous communication and synchronous actions by agents. To do so, we turn to the active research areas involving distributed event-, self- [37] and team-triggered strategies [38].

The principal idea behind event-, self-, and team-triggered strategies is that they provide a means for which agents are capable of only communicating when a designed triggering criterion occurs. The usual consequence of these methods is that agents communicate aperiodically and therefore less frequently than continuous or periodic strategies [2, 24, 25, 28, 38–40]. Consider an agent that continuously or periodically communicates its state to neighboring agents. This may be considered a waste of resources. Instead, a more efficient approach would be for the agent to only communicate its state information when it anticipates something may have changed or when something may have gone wrong.

A brief explanation of the differences between the three mentioned triggering strategies follows. An event-triggered strategy requires an agent to monitor its state until some conditions are met to initiate communication between agents. A self-triggered strategy requires an agent to determine when the next triggering time will occur given its current information. Finally, a team-triggered strategy requires a group of agents to collectively determine when to initiate communication by using an event- and/or self-triggered strategy [33]. Generally speaking, we would like to note that there is no clear advantage of one strategy over the others, and depending on the problem or application, one strategy maybe suitable where the others are not.

All triggering strategies in [2, 24, 25, 28, 38–40] have been shown to reduce the amount of message exchange and/or the amount of communication power consumed between the agents in a number of different multi-agent coordination tasks. This is due to the fact that agents only communicate when there is a need. In [24], an event-triggered algorithm for a multi-agent rendezvous task was developed that reduces the communication between the agents. For multi-agent average consensus tasks, self-triggered algorithms in [25, 28] and an event-triggered broadcasting algorithm in [39] have been shown to reduce the amount of communication and the amount of communication power consumed during the task. In [26,41], the communication power consumed was shown to be reduced for a leader-follower consensus problem using an event-triggered algorithm. The most relevant to this paper is the work involving event-, self- or team-triggered strategies applied to the deployment problem.

In relation to multi-agent deployment, triggering strategies have been shown to reduce the amount of communication as in [38, 40, 42]. More specifically, we are interested in algorithms that use Voronoi partitions to solve the deployment problem. The authors in [2] proposed a self-triggered Voronoi partitioning algorithm, the authors in [34, 35] proposed an event-triggered Voronoi partitioning algorithm, and the authors in [33] proposed a team-triggered Voronoi partitioning algorithm. The major drawback of these algorithms is the fact that when a trigger occurs, the agents must make a request for the information they require and other agents must respond to those requests i.e. instantaneous communication and synchronous actions by agents. We would like to note that the instantaneous communication (immediate requests and immediate responce) can also be viewed in the context of sensing capabilities as well. This is due to the fact that both methods obtain information from neighbors instantaneously when needed. Thus, if the agents have sensing capabilities, the agents can sense the surrounding environment to localize other agents rather than communicating with them via message transmissions. In this work, we propose a one-way communication model that is fully asynchronous where agents decide when to initiate communication by only broadcasting their information to others. This is similar to the communication model proposed in [39] for multi-agent average consensus. The broadcasting communication model can be considered a more practical strategy given that it is an asynchronous model that reduces the amount of communication traffic seen over a SWS by eliminating the need for response messages to occur. Additionally, since agents are restricted from requesting information or sending acknowledgments, they are unable to determine if their information has been sent to all intended recipients. This is in contrast to [2, 43] where agents communicate back-and-forth instantaneously to ensure sufficient information exchange. Also, the works in [34, 35] assume that the agents are aware of their neighbors and can precisely determine the communication range required, which is not a practical assumption. Instead, we will design our solution to allow agents to independently determine the sufficient broadcasting range required to ensure that their information is shared with the intended recipients.

Statement of contribution The main contribution of our work is the design of a fully asynchronous communication `event-triggered broadcasting algorithm` that allows agents to reach the set of locally optimal solutions to the deployment problem. Unlike all existing triggered solutions, our event-triggered broadcasting algorithm relies on agents to decide when to broadcast (push) information to others in the network without the need for a response from other agents. As a result, our communication model relaxed the assumptions of instantaneous communication and synchronous actions by agents (request and response actions). In addition, given the challenges of the communication constrains, we develop a distributed control algorithm such that the agents can determine the sufficient communication range required to reach all intended recipients. Finally, our distributed algorithm is shown to achieve a level of performance that is similar to that of a deployment strategy that uses continuous or periodic communication methods.

2.2 Preliminaries

Let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{Z}_{\geq 0}$ denote the sets of real, non-negative real, and non-negative integer numbers, respectively. Let $|\cdot|$ be the cardinality of a set. Also, we denote the Euclidean distance between two points $p, q \in \mathbb{R}^2$ by $\|p - q\|$.

Let Q be a convex polygon in $\mathbb{R}^2$ with a probability density function $\phi : Q \to \mathbb{R}_{\geq 0}$ that maps the probability of a spatial action or event occurring at point $q \in Q$. The *mass* and *center of mass* of Q with respect to the density function ϕ are

$$M_Q = \int_Q \phi(q) dq \quad \text{and} \quad C_Q = \frac{1}{M_Q} \int_Q q\phi(q) dq,$$

respectively. For a bounded set $Q \subset \mathbb{R}^2$, the *circumcenter*, $cc(Q) \in \mathbb{R}^2$, is the center of the closed ball of a minimum radius contained in Q, and the *circumradius*, $cr(Q) \in \mathbb{R}_{\geq 0}$, is the radius of the closed ball. Then, let the closed ball centered at q with a radius r be $\overline{B}(q, r)$.

2.2.1 Voronoi partition

In this subsection, we briefly present some concepts necessary for the development of the `event-triggered broadcasting algorithm`; further details on *Voronoi partitions* can be found in [11]. For a convex polygon, $Q \subset \mathbb{R}^2$, let $P = \{p_1, \ldots, p_N\}$ denote the locations of N agents in Q and let $\mathcal{I} = \{1, \ldots, N\}$ denote the set of identification numbers corresponding to the N agents with locations P. The set Q can be partitioned into N polygons $\mathcal{V}(P) = \{V_1, \ldots, V_N\}$ such that the union of their disjoint interiors is Q. The Voronoi cell of agent i is formally defined as

$$V_i = \{q \in Q \mid \|q - p_i\| \leqslant \|q - p_j\| \ \forall\, i \neq j\}. \tag{2.1}$$

When all agents are positioned at the centroids of their Voronoi cells, i.e., $p_i = C_{V_i}$, $\forall i \in \mathcal{I}$, the agents' locations $P = (p_1, \ldots, p_N)$ are said to be in a *centroidal Voronoi configuration.*

Furthermore, when the intersection of two Voronoi cells V_i and V_j generated by the points p_i and p_j is non-empty, $V_i \cap V_j \neq \emptyset$, the agents i and j are Voronoi neighbors. The set of Voronoi neighbors for the ith agent is denoted by $\mathcal{N}_i$.

2.2.2 Space partition with uncertain information

In order to compute the Voronoi cells defined by (2.1), each agent requires the exact positions of their neighboring agents. If agents do not continuously communicate and/or sense their surroundings, the exact locations of neighbors may not be available. Then, the agents must rely on inexact information to approximate their Voronoi cells. Here we discuss the concept of space partitioning when an agent's knowledge of its neighbors' positions is uncertain. Two methods for space partitioning under uncertain conditions are utilize, the *guaranteed* and the *dual-guaranteed Voronoi diagrams* [2,38,44,45].

Let $\mathcal{X} = \{X_1, \ldots, X_N\} \subset Q$ be a collection of compact sets containing the true positions of agents with $p_i \in X_i$, $\forall i \in \mathcal{I}$. The set X_i is considered to be a region of uncertainty and represents all the possible points in Q where agent i could potentially be located. If the location of agent i does not contain uncertainty, then the set X_i is simply a singleton with $X_i = \{p_i\}$.

The guaranteed Voronoi diagram, also known as the fuzzy Voronoi diagram [45], is the collection $g\mathcal{V}(\mathcal{X}) = \{gV_1, \ldots, gV_N\}$ of guaranteed Voronoi cells generated by the uncertainty regions of $\mathcal{X}$. The guaranteed Voronoi cell gV_i for agent i is the set of points that are guaranteed to be closer to agent i than any other agents. Formally,

$$gV_i = \{q \in Q \mid \max_{x_i \in X_i} \|q - x_i\| \leq \min_{x_j \in X_j} \|q - x_j\| \ \forall i \neq j\}.$$

Note that in general, the guaranteed Voronoi cell is a subset of the Voronoi cell, i.e. $gV_i \subset V_i \ \forall i \in \mathcal{I}$, which implies that the guaranteed Voronoi diagram is not a partition of Q. Fig. 2.1.(a). provides an example of a guaranteed Voronoi diagram consisting of five agents with uncertainty regions represented by circles.

Complementary to the guaranteed Voronoi diagram, the dual-guaranteed Voronoi diagram, first introduced in [2], is the collection $dg\mathcal{V}(\mathcal{X}) = \{dgV_1, \dots, dgV_N\}$ of dual-guaranteed Voronoi cells generated by the uncertainty regions of $\mathcal{X}$. The dual-guaranteed cell dgV_i for agent i is the collection of points such that any point outside the cell is guaranteed to be closer to all other agents than to agent i. Formally,

$$dgV_i = \{q \in Q \mid \min_{x_i \in X_i} \|q - x_i\| \leq \max_{x_j \in X_j} \|q - x_j\| \ \forall i \neq j\}.$$

Note that in general the Voronoi cell is a subset of the dual-guaranteed Voronoi cell, i.e. $V_i \subset dgV_i \ \forall i \in \mathcal{I}$. Fig. 2.1.(b). shows the dual-guaranteed Voronoi diagram of five agents and their uncertainty regions.

2.2.3 Facility location

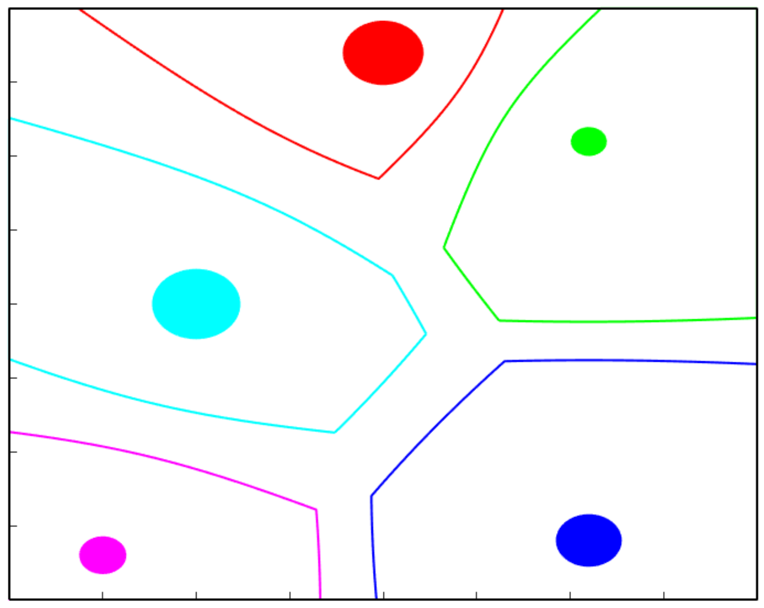
In this subsection, the locational optimization function presented in [12] is discussed. The function that quantifies the sensing performance of an agent located at point p_i to a point of interest $q \in Q$ is given by,

$$f(\|q - p_i\|) = \|q - p_i\|^2 . \tag{2.2}$$

This function measures the sensing quality of an agent based on its distance to a given point q. As the distance from an agent position p_i to the point of interest q decreases, the sensing performance for agent i at point q increases. For a density function $\phi(q) : Q \to \mathbb{R}_{\geq 0}$ that captures the likelihood of an action occurring at q, the total network performance of N agents at fixed positions P is given by,

$$\mathcal{H}(P) = E_\phi \Big[\min_{i \in \mathcal{I}} \|q - p_i\|^2 \Big]. \tag{2.3}$$

We assume $\phi(q)$ is provided to agents prior to deployment. The $\mathcal{H}$ function is beneficial when the agents are the closest to the actions that they are responsible for, and it has


(a)

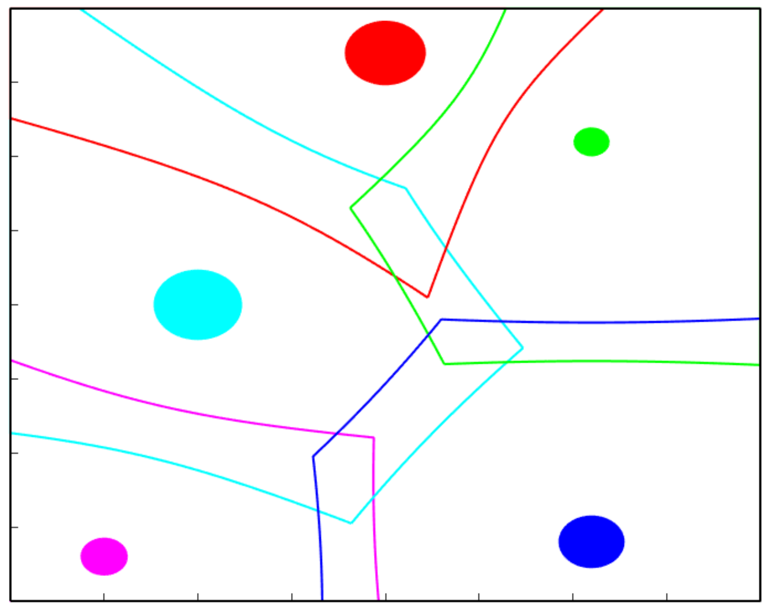

(b)

Figure 2.1: (a) Guaranteed and (b) dual-guaranteed Voronoi diagrams.

been used in a number of applications previously, such as in event detection and resource allocation [46, 47]. Given a Voronoi partition, each agent will be responsible for the points that are closer to itself than to any other agent, and now, the objective function with respect to a Voronoi partition as in [12] is written as

$$\mathcal{H}(P) = \sum_{i=1}^{N} \int_{V_i} \|q - p_i\|^2 \phi(q) dq. \tag{2.4}$$

For a distributed control algorithm where the agents have local or limited information, the agents optimize the objective function by moving toward the centroid of their Voronoi cells. For $P' \in Q$ with $\|p_i' - C_{V_i}\| \leq \|p_i - C_{V_i}\|$ for all $i \in \mathcal{I}$, then

$$\mathcal{H}(P', \mathcal{V}(P)) \leq \mathcal{H}(P, \mathcal{V}(P)). \tag{2.5}$$

In addition, when agents are located at the centroids of their Voronoi cells, $(p_1, \ldots, p_N) = (C_{V_1}, \ldots, C_{V_N})$, the objective function is considered to be in a locally optimized state [12]. In other words, when agents are located at the centroids of their Voronoi cells, the agents cannot improve the objective function by moving in any direction. This is commonly referred to as Nash equilibrium or a local minima result.

2.3 Problem Statement

Let $P = \{p_1, \ldots, p_N\} \in Q^N$ be the location of N agents moving in a convex polygon $Q \subset \mathbb{R}^2$. In this work we consider first-order dynamics for each agent,

$$\dot{p}_i = u_i, \tag{2.6}$$

where u_i is the control input for agent i, which is constrained by $\|u_i\| \leq s_{\max}$ with $s_{\max}$ being the maximum speed for all agents. We assume that all agents have knowledge of the density function $\phi(q)$ before starting the deployment task. In such a case, the density function can be seen as a populated area or as a higher crime area in a city that require more surveillance and monitoring. Similar to [2, 12], the $\phi(q)$ can be fixed during the deployment task. In addition, in case the density function is unknown, there have been many works to proper estimate a density function using, but not limited to, distributed Kriged Kalman filter and neural network as in [48, 49]. The authors in these works consider the scenario where the agents response to a dynamic change in an environment that is modeled by unknown dynamic density function. Also, we assume that agents are capable of maintaining speeds between 0 and $s_{\max}$ for some duration of time. It is worth pointing out that many ground, under water, and unmanned aerial vehicles have this capability.

Our objective is to achieve a locally optimal (local minimum) value of the objective function $\mathcal{H}$ in (2.4) with a fully asynchronous communication model while also taking a more communication-aware approach. More specifically, rather than requiring instantaneous communication and/or synchronous actions by agents, as in many similar triggered deployment algorithms [2, 33–35], we consider a fully asynchronous broadcast model without acknowledgment of or requests for information where the agents must actively choose a broadcast radius to transmit their messages similar to [39]. Additionally, we want to reduce the amount of communications across the SWS by having the agents only broadcast messages when necessary, rather than continuously or periodically. This means the agents will need to determine exactly when to broadcast messages to one another, with what distance, and how to move in the environment based on locally available information.

Formally, let $\{t_\ell^i\}_{\ell \in \mathbb{Z}_{\geq 0}} \subset \mathbb{R}_{\geq 0}$ be the sequence of times (to be determined on-line) at which agent i broadcasts its position p_i to other agents in the network. The radius at which the message is broadcast is $R_\ell^i > 0$, meaning any agent j that is within R_ℓ^i of $p_i(t_\ell^i)$ will receive the message.

Let $p_j^i(t) \in Q$ be the last known position of agent j by agent i at any given time $t \in \mathbb{R}_{\geq 0}$. Note that if agent i has never received information from some agent j, then $p_j^i(t) = \emptyset$. Then, given a sequence of broadcast times $\{t_\ell^j\}_{\ell \in \mathbb{Z}_{\geq 0}}$ and broadcast radii $\{R_\ell^j\}_{\ell \in \mathbb{Z}_{\geq 0}}$ for each agent $j \in \mathcal{I} \setminus \{i\}$, the memory of each agent i is updated according to

$$p_j^i(t) = \begin{cases} p_j(t_\ell^j) & \text{if } p_i(t_\ell^j) \in \overline{B}(p_j(t_\ell^j), R_\ell^j), \\ p_j^i(t_{\ell-1}^j) & \text{otherwise,} \end{cases} \tag{2.7}$$

for $t \in [t_\ell^j, t_{\ell+1}^j)$.

The goal is now to devise a distributed coordination strategy such that the agents converge to a centroidal Voronoi configuration, which locally optimizes $\mathcal{H}$ in (2.4), as in [12]. However, we do so with an asynchronous communication aware model. To this end we are interested in broadcasting messages as minimally as possible, both in frequency and space, by only broadcasting when necessary.

Problem 2.1. Given the dynamics (2.6) and the communication model (2.7), find a distributed communication and control strategy to find a sequence of broadcasting times $\{t_\ell^i\}_{\ell \in \mathbb{Z}_{\geq 0}}$, broadcasting ranges $\{R_\ell^i\}_{\ell \in \mathbb{Z}_{\geq 0}}$, and a control strategy $u_i(t)$ for all $i \in \mathcal{I}$ that drives the agents to their Voronoi centroids in order to locally optimize the objective function $\mathcal{H}$ in (2.4).

2.4 Event-Trigger Algorithm Design

When an agent has knowledge of the exact locations of its Voronoi neighbors, the agent can compute its exact Voronoi cell and the exact location of the cell's centroid. This allows the agent to move directly toward the Voronoi cell centroid. This will monotonically optimize the objective function $\mathcal{H}$ in (2.4) as in [12]. Unfortunately with a fully asynchronous event-triggered broadcasting communication model, the agents do not know when they will receive new information from other agents, nor can they request the information when they require it as is done in [2, 33–35]. Instead, they must rely on the information that they possess at each moment in time in order to determine exactly how to move and when to initiate data transfers to others. Although, it is known from [2] that agents only need information from their Voronoi neighbors to move to their Voronoi centroids, the major challenge now is that the agents do not necessarily know their actual Voronoi neighbors. This is due to the continuously increasing uncertainty that exists with respect to the positions of other agents when not communicating. This is combined with the fact that agents cannot gain knowledge of other agent positions by requesting the information.

2.4.1 Information Required and Memory Structures

Since agents do not exchange information with each other on a continuous basis, each agent must maintain the most current state information they have received from other agents, as well as a method to model the uncertainty that evolves over time. The data structure that allows the agents to compute the uncertainty regions of any other agent using the most recently received information with respect to agent $j \in \mathcal{I} \setminus \{i\}$ is the following. One, the time instance t_j^i when agent i last successfully received information from agent j. Two, the position $p_j^i = p_j(t_j^i)$ of agent j received at time t_j^i, and three, the speed promise $s_j^i = s_j(t_j^i)$ by agent j received at time t_j^i. The notion of the promise is borrowed from [50]. The speed promise s_j^i made to agent i by agent j states that agent j promises not to exceed the speed given by the promise s_j^i. In other words, agent j's dynamics will follow $\|u_j(t)\| \leq s_j^i$, and this will hold for agent j until it broadcasts again. For now, we consider agent j only setting its speed to one of two values $s_j \in \{0, s_{\max}\}$. This can be seen as two modes of operation signaling either 'active' i.e. agent j is moving or 'inactive' i.e. agent j is not moving by holding its current position. This assumption helps simplify the convergence result in Section 2.6 and will be relaxed in Section 2.7 where agents can modify their speed and promises to any $s_j \in [0, s_{\max}]$.

We define $D_j^i = (t_j^i, p_j^i, s_j^i) \in (\mathbb{R}_{\geq 0} \times Q \times \{0, s_{\max}\})$ as the information that agent i last received about any given agent j, and $D_j^i = (\emptyset)$ if agent i has not received information from agent j. This information allows agent i to construct a closed ball that guarantees to contain the jth agents' real location. For any time $t \geq t_j^i$, agent i knows that the jth agent did not move farther than $s_j^i(t-t_j^i)$ away from p_j^i. Given the promise and the dynamics (2.6), the uncertainty region of the jth agent with respect to the information agent i has is formally defined as

$$X_j^i(t) = \overline{B}(p_j^i, s_j^i(t-t_j^i)). \tag{2.8}$$

We denote by $D_i^i = (t_i^i, p_i^i, s_i^i)$ the ith agent's information at its latest broadcast time t_i^i and by $D_i = (t, p_i, s_i)$ the ith agent's current information. The ith agent's full memory at any given time is collected in

$$\mathcal{D}_i = (D_1^i, \ldots, D_N^i) \in (\mathbb{R}_{\geq 0} \times Q \times \{0, s_{\max}\})^N.$$

Additionally, the memory for all agents in the entire network is defined by

$$\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_N\} \in (\mathbb{R}_{\geq 0} \times Q \times \{0, s_{\max}\})^{N^2}. \tag{2.9}$$

With the memory structure defined, we are now ready to begin solving Problem 2.1. We begin by determining exactly which agents in the network should a particular agent broadcast to in order to complete the deployment task. Since it is known from [2] that an agent's knowledge of its Voronoi neighbors is useful in solving a similar self-triggered deployment problem, we determine a method for agents to keep track of their Voronoi neighbors for the event-triggered broadcasting problem given the uncertainty regions. For a given agent i and its memory $\mathcal{D}_i$, we propose the notion of *dual-guaranteed neighbors* next. Then, as long as agent i maintains some type of communication with its dual-guaranteed neighbors $j \in \mathcal{N}_{dg_i}$, we conclude that the agents will have a sufficient amount of information to allow them to move toward their Voronoi centroids effectively locally optimizing the objective function $\mathcal{H}$ in (2.4).

Definition 2.2 (Dual-Guaranteed Neighbors)**.** Given a set of uncertainty regions $\mathcal{X} = (X_1, \ldots, X_N)$ such that $p_j \in X_j$ for all $j \in \mathcal{I}$, a *dual-guaranteed neighbor* of agent i is any agent j that can be made a Voronoi neighbor for at least one configuration of positions $P \subset \mathcal{X}$. Formally, the set of dual-guaranteed neighbors of agent i is

$$\mathcal{N}_{dg_i} = \{j \in \mathcal{I} \mid \exists P \subset \mathcal{X} \text{ s.t. } j \in \mathcal{N}_i\}.$$

The dual-guaranteed neighbors are defined such that $\mathcal{N}_i \subset \mathcal{N}_{dg_i}$ which is formalized in Lemmas 2.3 below.

Lemma 2.3 (Dual-Guaranteed Neighbors)**.** Given a set of uncertainty regions $\mathcal{X} = (X_1, \dots, X_N)$ such that $p_j \in X_j$ for all $j \in \mathcal{I} \setminus \{i\}$, if

$$dgV_i(\mathcal{X}) \cap dgV_j(\mathcal{X}) \neq \emptyset,$$

then $\exists P \subset \mathcal{X}$ such that agent $j \in \mathcal{N}_{dg_i}$ can be a Voronoi neighbor of agent i.

Proof. In appendix A. □

Moreover, an agent i might not have any information about some agents in the network if they have not communicated yet. This is troublesome since the computation of gV_i and dgV_i are based on the availability of the uncertainty regions that guarantee $p_j \in X_j^i$ for all times. An even bigger challenge given the setup of our problem is the possibility that some agents are in communication for some period of time before stopping altogether, since it is no longer necessary. In case that agents do not communicate with all other agents, we need to determine additional conditions to ensure that, with partial information from $\mathcal{D}_i$, $\mathcal{N}_i \subset \mathcal{N}_{dg_i}$ is guaranteed. To address this, we define a map $\pi_{\mathcal{J}} : (\mathbb{R}_{\geq 0} \times Q \times \{0, s_{\max}\})^N \to (\mathbb{R}_{\geq 0} \times Q \times \{0, s_{\max}\})^{|\mathcal{J}|}$ for any $\mathcal{J} \subset \mathcal{I}$ that extracts information corresponding only to agents $j \in \mathcal{J}$ from $\mathcal{D}_i$. Formally,

$$\pi_{\mathcal{J}}(\mathcal{D}_i) = \cup_{j \in \mathcal{J}} \{D_j^i\}.$$

Furthermore, we require $\pi_{\mathcal{J}}(\mathcal{D}_i)$ to be sufficient to guarantee $\mathcal{N}_i \subset \mathcal{N}_{dg_i}$. Intuitively, this means that all $k \notin \mathcal{J}$ agents should be sufficiently far away from agent i that agent k is not a Voronoi neighbor of agent i given it $p_k(t)$. This is formalized in Corollary 2.4 below.

Corollary 2.4 (Condition on Voronoi Neighbors Sets)**.** Given a set of uncertainty regions $\mathcal{X} = (X_i \cup \{X_j^i\}_{j\in\mathcal{J}})$ such that $X_i = \overline{B}(p_i, 0)$ and $p_j \in X_j^i$ for all $j \in \mathcal{J}$, if

$$p_k \notin \overline{B}\left(p_i, 2 \cdot \max_{q \in dgV_i(\mathcal{X})} (\|q - p_i\|)\right) \ \forall k \in \mathcal{I} \setminus \mathcal{J} \setminus \{i\},$$

then $\mathcal{N}_i \subset \mathcal{N}_{dg_i}$ is guaranteed.

Proof. In appendix B. □

It is important to emphasize that for any $j \in \mathcal{N}_{dg_i}$, if agent j is a dual-guaranteed neighbor of agent i, it does not imply that agent i is a dual-guaranteed neighbor of agent j since $\mathcal{D}_i \neq \mathcal{D}_j$. It is worth noting that agents do not need to have knowledge of the number of agents in the network. In order to describe the algorithm in a simpler manner, $\mathcal{D}_i$ contains a place holder for each agent in the network (even if the actual execution of the algorithm will never need to access all of this information), and as further explained, the agents only need information from the dual-guaranteed neighbors. For convenience, we let

$$gV_j^i(\mathcal{J}) = gV_j(\{X_j^i\}_{j\in\mathcal{J}} \cup X_i),$$

$$dgV_j^i(\mathcal{J}) = gV_j(\{X_j^i\}_{j\in\mathcal{J}} \cup X_i).$$

The objects in this subsection are summarized in Table 2.1 for agent $i, j \in \mathcal{I}$.

2.4.2 Motion Control Law

The motion control law defines a method to generate trajectories for the agents that allows them to contribute positively to the deployment task. This is accomplished by having agents move towards the midpoint between the centroids of their guaranteed and dual-guaranteed Voronoi cells. Due to uncertainties, agents must use the information pertaining

Table 2.1: Agent i model definitions

$p_i \in \mathbb{R}^2$	agent i's location
$p_j^i \in \mathbb{R}^2$	agent j broadcasted location to agent i
$s_i \in \{0, s_{\max}\}$	agent i's speed
$s_j^i \in \{0, s_{\max}\}$	agent j promised speed to agent i
$t_j^i \in \mathbb{R}_{\geq 0}$	agent j broadcasting time to agent i
$\mathcal{N}_i \subset \mathcal{I}$	agent i's Voronoi neighbors
$\mathcal{N}_{dg_i} \subset \mathcal{I}$	agent i's dual-guaranteed neighbors
$\mathcal{D}_i \in \mathcal{D}$	agent i's full memory
$D_j^i \in \mathcal{D}_i$	agent j's information in the ith agent's memory
$X_j^i \subset Q$	agent j's uncertainty region given D_j^i
$gV_j^i(\mathcal{J}) \subset Q$	agent j's guaranteed Voronoi cell with respect to $\pi_{\mathcal{J}}(\mathcal{D}_i) \cup D_i$ information
$dgV_j^i(\mathcal{J}) \subset Q$	agent j's dual-guaranteed Voronoi cell with respect to $\pi_{\mathcal{J}}(\mathcal{D}_i) \cup D_i$ information

to their dual-guaranteed neighbors, which is guaranteed to contain all their Voronoi neighbors. Since $\mathcal{N}_i \subset \mathcal{N}_{dg_i}$ by Lemma 2.3, the agents guarantee computing the guaranteed and dual-guaranteed Voronoi cells such that $gV_i^i(\mathcal{N}_{dg_i}) \subset V_i \subset dgV_i^i(\mathcal{N}_{dg_i})$. Now let us informally describe the idea behind it here.

At each time-step, agent i uses partial information, its dual-guaranteed neighbors' information, to determine if it can move in a manner to optimize $\mathcal{H}$ in (2.4). If so, it computes the centroids of the guaranteed and dual-guaranteed Voronoi cell and then moves toward the midpoint between them. Otherwise, it does not move and waits until it receives sufficient information to initiate the continuation of motion.

In general, it is preferable that agents move toward their Voronoi centroids directly. However, this requires perfect information at all times. Instead, we establish a convex set $\mathcal{C}_i(\mathcal{N}_{dg_i})$ that is guaranteed to contain the true centroid C_{V_i} based on the information $\mathcal{N}_{dg_i}$ available to agent i. This set can then be used not only to determine how to move but also exactly when updated information is needed. This is formalized next in Proposition 2.5.

Proposition 2.5. Given $D^i_j = (t^i_j, p^i_j, s^i_j)$ for all $j \in \mathcal{N}_{dg_i}$, let

$$\mathcal{C}_i(\mathcal{N}_{dg_i}) = \overline{B}(C_{gV^i_i(\mathcal{N}_{dg_i})}, \text{bnd}_i) \cap \overline{B}(C_{dgV^i_i(\mathcal{N}_{dg_i})}, \text{bnd}_i), \tag{2.10}$$

where

$$\text{bnd}_i = 2cr_{dgV^i_i(\mathcal{N}_{dg_i})}\left(1 - \frac{M_{gV^i_i(\mathcal{N}_{dg_i})}}{M_{dgV^i_i(\mathcal{N}_{dg_i})}}\right),$$

then $C_{V_i} \in \mathcal{C}_i(\mathcal{N}_{dg_i})$.

Proposition 2.5 says that although based on uncertain information the exact location of C_{V_i} cannot be determined, its distance from the centroids of the guaranteed $C_{gV^i_i(\mathcal{N}_{dg_i})}$ and dual guaranteed $C_{dgV^i_i(\mathcal{N}_{dg_i})}$ can be upper-bounded by the same quantity bnd_i. The set $\mathcal{C}_i$ is then just the intersection of the two balls centered at $C_{gV^i_i(\mathcal{N}_{dg_i})}$ and $C_{dgV^i_i(\mathcal{N}_{dg_i})}$ with radii bnd_i. It is then easy to see that the set $\mathcal{C}_i$ is convex, and thus while agent i is outside of this set, it can guarantee to be getting closer to the true centroid C_{V_i} by simplify moving towards $\mathcal{C}_i$. With that being said, we define m_i as the closest point on $\mathcal{C}_i(\mathcal{N}_{dg_i})$ to agent i. Formally,

$$m_i = \underset{q \in \mathcal{C}_i(\mathcal{N}_{dg_i})}{\arg\min} \|q - p_i\|, \tag{2.11}$$

and as long as

$$p_i \neq m_i, \tag{2.12}$$

the agent has good information to move. Then, our motion control becomes

$$u_i = s_i \frac{m_i - p_i}{\|m_i - p_i\|}, \; \forall\, i \in \mathcal{I}, \tag{2.13}$$

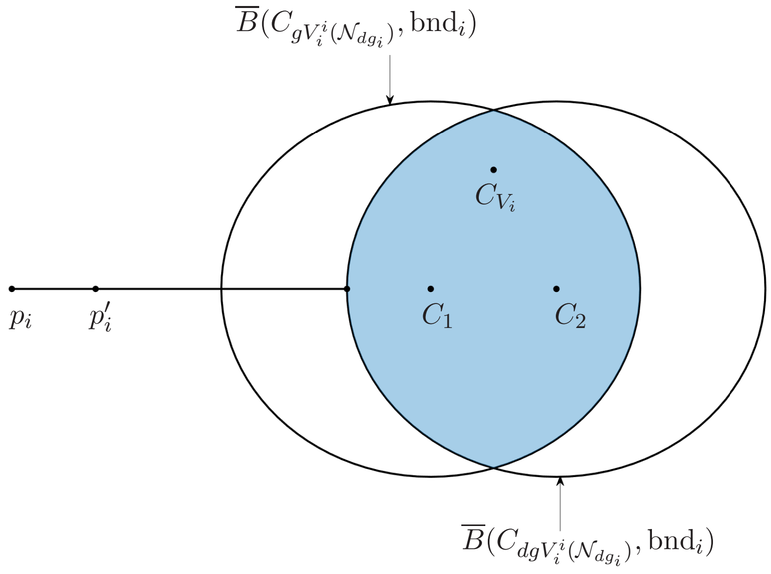

$\overline{B}(C_{gV_i^i(\mathcal{N}_{dg_i})}, \text{bnd}_i)$
C_{V_i}
p_i
p_i'
C_1
C_2
$\overline{B}(C_{dgV_i^i(\mathcal{N}_{dg_i})}, \text{bnd}_i)$

Figure 2.2: Example of a point p_i' that an agent can move to given $C_1 = C_{gV_i^i(\mathcal{N}_{dg_i})}$, $C_2 = C_{dgV_i^i(\mathcal{N}_{dg_i})}$ and bnd_i.

where

$$s_i = \begin{cases} s_{\max} & \text{if condition (2.12) is true,} \\ 0 & \text{otherwise .} \end{cases}$$

Fig.2.2 show an example of a point an agent can move to given the uncertainties, and Algorithm 4 formalizes the Motion control law.

Algorithm 4: Motion Control Law

```
At any time $t > 0$, agent $i \in \mathcal{I}$ performs:
1: sets $D = \pi_{\mathcal{N}_{dg_i}}(\mathcal{D}_i) \cup D_i$
2: computes $\mathcal{X}(D)$ as in (2.8)
3: computes $L = gV_i(\mathcal{X})$ and $C_L$
4: computes $U = dgV_i(\mathcal{X})$ and $C_U$
5: computes $m_i$ as in (2.11)
6: computes $u_i$ as in (2.13)
```

2.4.3 Decision Control Law

Equipped with a motion control law, we now need to design a communication protocol that provides sufficient information to the motion control law to properly do its job. This is nontrivial because the agents do not have control over when they will receive updated information from others, but instead can only choose when they send information. Thus, rather than designing the decision control law for agent i based on when *it* needs information, our decision control law would ideally be designing in terms of when its neighbors need information. However, since agents in general may not know exactly what their neighbors need (due to distributed information), we instead propose a law that broadcasts messages as minimally as possible while still ensuring the entire network converges to the desired set of states.

Starting with the case when agent i increases its speed from 0 to $s_{\max}$ without broadcasting, the neighbors $j \in \mathcal{N}_{dg_i}$ will not be able to capture the correct uncertainty region of agent i. This can lead to failure of achieving the deployment task if the case is not handled appropriately. To handle this situation, the dual-guaranteed neighbors must be informed about the new change in speed as soon as it increases. This makes it possible for other agents to appropriately manage the uncertainty that they possess for agent i's location by modifying the rate of change at which the uncertainty evolves. Therefore, agents shall broadcast their information when their speed increases i.e. change from 0 to $s_{\max}$.

For the case when agent i changes its speed to 0 without broadcasting, the uncertainty about its location held by other agents will increase. As a result, its uncertainty will eventually become larger than necessary. Consequently, the dual-guaranteed neighbors will no longer have enough reliable information to continue to move. To prevent this scenario, the ith agent's dual-guaranteed neighbors need to know when the agent's speed has been set to 0. This allows the neighbors to halt the expansion of the uncertainty region for agent i. Therefore, agents shall broadcast their state information when they change their speeds to 0.

For the case when agent i realizes that it has a new dual-guaranteed neighbor $j \in \mathcal{N}_{dg_i}$, both agents i and j must know about each other because they can be Voronoi neighbors. Thus, agent i must broadcast as soon as it gets information from the new dual-guaranteed neighbor to ensure agent j has its information. Furthermore, we would like to note that for a deployment problem, as the agents move away from each other, the set $\mathcal{N}_{dg_i}$ gets smaller, and if this case happens, it only occurs finite times during the deployment task.

Algorithm 5 formalizes the decision control law.

Algorithm 5: Decision Control Law

```
At any time $t > 0$, agent $i \in \mathcal{I}$ performs:
1: if $s_i > s_i^i$ then
2:     broadcasts $D_i$
3: else if $s_i = 0$ then
4:     broadcasts $D_i$
5: else if agent $i$ has a new dual-guaranteed neighbor then
6:     broadcasts $D_i$
7: end if
```

2.4.4 Broadcasting Range

Equipped with a motion control law and a method for determining exactly when broadcasting new information is necessary, we are interested in determining the minimum broadcasting range required. Since $\mathcal{N}_i \subset \mathcal{N}_{dg_i}$ by Lemma 2.3, we would like to broadcast to all agents $j \in \mathcal{N}_{dg_i}$. This guarantees to send the information to all agents $j \in \mathcal{N}_i$.

Thus, we aim to find the minimum radius for an agent to broadcast in order to guarantee that all $j \in \mathcal{N}_{dg_i}$ are reached. This can be achieved by finding the distance from an agent i at p_i to its farthest dual-guaranteed neighbor. By Corollary 2.4, any agent j s.t.

$$p_j \notin \overline{B}\left(p_i, 2 \cdot \max_{q \in dgV_i^i(\mathcal{N}_{dg_i})} (\|q - p_i\|)\right)$$

is guaranteed not to be a dual-guaranteed neighbor. Therefore, the agent can broadcast with a distance of $(2 \cdot \max_{q \in dgV_i^i(\mathcal{N}_{dg_i})} (\|q - p_i\|))$, which is sufficient to transmit information to all $j \in \mathcal{N}_{dg_i}$. By Lemma 2.3, this guarantees that all the Voronoi neighbors of agent i receive agent i's information while agent i only uses the information provided by the data in $dgV_i^i(\mathcal{N}_{dg_i})$ to achieve this result. The computation of the broadcasting range by agent i is formally defined as

$$R_\ell^i(\mathcal{N}_{dg_i}) = 2 \cdot \max_{q \in dgV_i^i(\mathcal{N}_{dg_i})} (\|q - p_i(t_\ell^i)\|). \tag{2.14}$$

2.5 Event Triggered Broadcasting Algorithm

In this section, we combine the motion control law, decision control law and broadcasting range assignment to synthesize the fully asynchronous communication `event-triggered broadcasting algorithm`. The goal is for the agents to converge to to the set of centroidal Voronoi configurations in order to locally optimize $\mathcal{H}(P)$ given the communication model presented thus far.

Given the nature of our problem, agent i and j may communicate for some duration of time and then stop when it is no longer necessary. However, the sets X_i^j and X_j^i will continue to expand even if communication is discontinued that effect the motion of both agents. To address this issue, we introduce a new set of neighbors called the potential neighbor set $\mathcal{P}_i$ of agent i that exclude the dual-guaranteed neighbors of the ith agents which are guaranteed not to be Voronoi neighbors due to discontinuation of communication. In addition, the potential neighbor set satisfies $\mathcal{N}_i \subset \mathcal{P}_i \subset \mathcal{N}_{dg_i}$. Now, let us introduce the mechanism to update the potential neighbor set when an agent receives new information as following:

$$\mathcal{P}_i^+ = \{\{k\}_{k \in \mathcal{P}_i'} \mid dgV_i(\mathcal{X}) \cap dgV_k(\mathcal{X}) \neq \emptyset\}, \tag{2.15}$$

where $\mathcal{X} = X_i \cup \{X_j^i\}_{j \in \mathcal{P}_i}$ and $\mathcal{P}_i' = (\mathcal{P}_i \cup j)$ if $j \notin \mathcal{P}_i$ and $\mathcal{X} = X_i^i \cup \{X_j^i\}_{j \in \mathcal{P}_i}$ and $\mathcal{P}_i' = \mathcal{P}_i$ otherwise. Note, if an agent does not receive new information, $\mathcal{P}_i$ does not change.

Lemma 2.6 (Potential Neighbors Properties)**.** Under the decision control law, broadcasting range assignment and updating potential neighbors mechanism in (2.15), $\mathcal{N}_i \subset \mathcal{P}_i$ is guaranteed at every event-time t_ℓ^i.

Proof. In appendix D. $\square$

Proposition 2.7. Under the `event-triggered broadcasting algorithm`, $gV_i^i(\mathcal{P}_i) \subset V_i \subset dgV_i^i(\mathcal{P}_i)$ is guaranteed at all times.

Proof. In appendix E. $\square$

Thus far, by Proposition 2.7, the algorithm ensures that the requirement for the motion control law is satisfied, and the analysis of the asymptotic convergence properties is provided in the following section.

Now, we proceed with an informal description of the proposed algorithm. Note that it is assumed that each agent knows their potential neighbors at time $t = 0$ and consequently $\mathcal{N}_i \subset \mathcal{P}_i$. Also, we assume that the agents are provided with the density function $\phi(q)$. Let us start with when the agent's speed is 0. This implies agent i cannot contribute positively to the task. Therefore, it waits until it has a sufficient amount of information such that condition (2.12) holds where the uncertainties of the potential neighbors are computed using (2.8). When agent i receives enough information to move, it sets its speed to $s_{\max}$ and broadcasts its state information at a distance $R_\ell^i(\mathcal{P}_i)$ from its current position p_i.

Furthermore, when agent i is able to contribute positively i.e. $s_i = s_{\max}$, it expands the uncertainties of its potential neighbors by the maximum rate of change, as in (2.16), and hold on the new received information until it broadcasts again. Then, the agent follows the motion control law (2.13) until condition (2.12) is not satisfied. We would like to note that an agent holds on the new information when it is in motion to allow it-self to follow the motion control law without the need to information from any agent $j \notin \mathcal{P}_i$ as explained in the proof of Proposition2.7. Then, when condition (2.12) becomes invalid, the agent computes the uncertainties of its potential neighbors using (2.8) and checks for the condition (2.12). If it is valid, the agent broadcasts its state information to a distance of $R_\ell^i(\mathcal{P}_i)$ to ensure $\mathcal{N}_i \subset \mathcal{P}_i$ and repeats this until condition (2.12) is no longer valid. Next, the agent sets its speed to zero and broadcasts its state information to a distance of $R_\ell^i(\mathcal{P}_i)$.

Then, the agent waits until it gets sufficient information such that condition (2.12) is valid.

$$\overline{X_j^i}(t) = \overline{B}\left(p_j^i, \left(s_{\max}(t - t_i^i) + s_j^i(t_i^i - t_j^i)\right)\right). \tag{2.16}$$

Algorithm 6 formally describes the event-triggered broadcasting control law.

Algorithm 6: `event-triggered broadcasting algorithm`

```
Initialization at time $t = 0$, agent $i \in \mathcal{I}$ performs:
 1: sets $D = \pi_{\mathcal{P}_i}(\mathcal{D}_i) \cup D_i$
At any time $t > 0$, agent $i \in \mathcal{I}$ performs:
 1: updates $\mathcal{P}_i$ using (2.15)
 2: if $s_i = 0$ then
 3:     update $D = \pi_{\mathcal{P}_i}(\mathcal{D}_i) \cup D_i$
 4:     computes $\mathcal{X}(D)$ as in (2.8)
 5: else
 6:     computes $\overline{\mathcal{X}}(D)$ as in (2.16)
 7: end if
 8: if $s_i = 0$ and condition (2.12) is valid then
 9:     sets $s_i = s_{\max}$
10:     broadcasts $D_i$ using (2.14) $R_\ell^i(\mathcal{J}_i)$ distance away
11: else if $s_i \neq 0$ and condition (2.12) is invalid then
12:     update $D = \pi_{\mathcal{P}_i}(\mathcal{D}_i) \cup D_i$
13:     computes $\mathcal{X}(D)$ as in (2.8)
14:     if condition (2.12) is valid then
15:         broadcasts $D_i$ using (2.14) $R_\ell^i(\mathcal{J}_i)$ distance away
16:     else
17:         sets $s_i = 0$
18:         broadcasts $D_i$ using (2.14) $R_\ell^i(\mathcal{J}_i)$ distance away
19:     end if
20: end if
21: if $t \neq t_i^i$ and agent $i$ has new potential neighbor then
22:     broadcasts $D_i$ using (2.14) $R_\ell^i(\mathcal{J}_i)$ distance away
23: end if
24: compute $u_i$ as in (2.13)
```

2.6 Convergence Analysis of the Event-Triggered Broadcasting Algorithm

In this section, we analyze the asymptotic convergence properties of the `event-triggered broadcasting algorithm`. Recall that our objective is to drive the agents to their Voronoi centroids because if the agents converges to the set of centroidal Voronoi configurations, the agents are locally optimized $\mathcal{H}(P)$ [12], which also means reaching a local optimal with respect to $\mathcal{H}(P)$, where

$$\mathcal{H}(P) = \sum_{i=1}^{N} \int_{V_i} \|q - p_i\|^2 \phi(q)dq. \tag{2.17}$$

Proposition 2.8. The agents' location evolving under the `event-triggered broadcasting algorithm` from any initial location configuration in Q^N converges to the set of centroidal Voronoi configurations

Proof. We know from [12] that

$$\dot{\mathcal{H}}(P) = \sum_{i=1}^{N} (2M_{V_i}(p_i - C_{V_i})\dot{p_i}), \tag{2.18}$$

and now we want to show that $\dot{\mathcal{H}} \leq 0$ under `event-triggered broadcasting algorithm`

$$\dot{\mathcal{H}}(P) = \sum_{i=1}^{N} \left(2M_{V_i}(p_i - C_{V_i}) \frac{-s_i}{\|p_i - m_i\|}(p_i - m_i) \right),$$

$$= \sum_{i=1}^{N} \left(\frac{-2M_{V_i}s_i}{\|p_i - m_i\|}(p_i - C_{V_i}) \cdot (p_i - m_i) \right).$$

Under the motion control law if $p_i \neq m_i$, the $(p_i - C_{V_i}) \cdot (p_i - m_i) \geq 0$. This implies $\dot{\mathcal{H}}(P) < 0$. In fact, $\dot{\mathcal{H}}(P)$ is strictly negative if there is at least one agent moving toward its Voronoi centroid [12]. In case all agents are stationary such that $s_i = 0 \ \forall \ i \in \mathcal{I}$, $\dot{\mathcal{H}}(P) = 0$. In addition, if the agents are stationary and stay stationary, it means that the agents know the exact location of their neighbors since $s_j = 0 \ \forall \ j \in \mathcal{P}_i$, and it means that $\text{bnd}_i = 0$, $\overline{B}(C_{gV_i^i(\mathcal{N}_{dg_i})}, \text{bnd}_i) = C_{gV_i^i(\mathcal{N}_{dg_i})}$, and $\overline{B}(C_{dgV_i^i(\mathcal{N}_{dg_i})}, \text{bnd}_i) = C_{dgV_i^i(\mathcal{N}_{dg_i})}$ $\forall \ i \in \mathcal{I}$. If condition 2.12 is invalid for all agents, due to $\|p_i - m_i\| = 0$, this implies $\forall i \in \mathcal{I}, \ p_i = C_{V_i} = m_i = C_{gV_i^i(\mathcal{N}_{dg_i})} = C_{dgV_i^i(\mathcal{N}_{dg_i})}$. Since $p_i = C_{V_i} \ \forall \ i \in \mathcal{I}$, the agents converged to the set of centroidal Voronoi configurations. This conclude the proof. $\square$

2.7 Varying Speed Extensions

In this section, we consider an extension to the `event-triggered broadcasting algorithm` where agents can adjust their speeds to values other than zero or maximum. In other words we consider the case where agent i may take on a speed $s_i \in [0, s_{\max}]$. The main advantage of adjusting the speed of agents is to reduce the amount of communication between agents when an agent approaches its cell centroid while not effecting the asymptotic convergence. As the agents get closer to their Voronoi centroids, the agents start to communicate more frequently when they are moving at $s_{\max}$. Rather than allowing the agents to move with $s_{\max}$ all the time, an alternative approach is for agents to determine the appropriate speed to travel as they become close to their Voronoi centroids. For a variable speed adjustment approach, an agent's speed at any given instance in time can be described by the following,

$$s_i = \beta_i s_{\max}, \tag{2.19}$$

where $0 < \beta_i \leq 1$ is determined online by the agents. Each agent updates β_i based on a design parameter Δ_{TB}, where Δ_{TB} represents a target time duration for agents to attempt to maintain movement prior to switching states and broadcasting information. The parameter Δ_{TB} is chosen prior to the start of the deployment task and the value chosen can depend on the anticipated communication traffic that will occur over the wireless network. If agent i finds itself in a scenario where it stays in a moving state for less time than Δ_{TB}, then this would imply that agent i is moving with a greater speed than desired. Thus, agent i will decrease it's speed by decreasing β_i. On the other hand, if the agent stayed in moving state greater than Δ_{TB}, this would imply that agent i is moving with a speed that is slower than desired. Thus, agent i will increase its speed by increasing β_i. It is important to note that when agent i needs to increase the value of β_i, it must broadcast the new speed to its neighbors so that they can correctly capture the uncertainty region associated with agent i as explained in the first case in Section 2.4.3.

In addition, by using the maximum speed $s_{\max}$ to capture the uncertainty of the neighboring agents in (2.16), a triggered event may occur faster than desired. Instead, the agents assume their neighbors are moving with a constant speed $s_{\mathcal{P}_i}$ when they are in motion. Let $s_{\mathcal{P}_i}$ be the maximum speed of the ith agent and its potential neighbors such that

$$s_{\mathcal{P}_i} = \max\{s_i, \max_{j \in \mathcal{P}_i}(s^i_j)\}, \tag{2.20}$$

and let us rewrite (2.16) in terms of $s_{\mathcal{P}_i}$ as following:

$$\overline{X^i_j} = \overline{B}\left(p^i_j, \left(s_{\mathcal{P}_i}(t - t^i_i) + s^i_j(t^i_i - t^i_j)\right)\right). \tag{2.21}$$

Remark 2.9. The asymptotic convergence properties of the `event-triggered broadcasting algorithm` holds for any speed policy since the speed only effects the convergence time and nothing else.

Algorithm 7 summarizes the event-triggered broadcasting control algorithm with variable speed.

2.8 Simulation

In this section, we provide simulation results for the `event-triggered broadcasting algorithm`. The simulations were developed using `MATLAB 2019a`. For all simulations, a time-step of $\Delta t = 1/60s$ was chosen as if agents operating frequency is 60Hz and all simulations were performed with eight agents $N = 8$ in a $40m \times 40m$ square environment. Agents were initialized with the locations,

$$P = \big\{(11.8, 36.3), (1.1, 6.0), (11.7, 20.1), (15.3, 5.5), \\ (11.6, 1.0), (7.5, 9.1), (17.0, 15.3), (13.5, 6.3)\big\}.$$

Algorithm 7: `event-triggered broadcasting algorithm` with variable speed

```
Initialization at time $t = 0$, agent $i \in \mathcal{I}$ performs:
 1: sets $D = \pi_{\mathcal{P}_i}(\mathcal{D}_i) \cup D_i$
 2: sets $\beta_i = 1$
At any time $t > 0$, agent $i \in \mathcal{I}$ performs:
 1: updates $\mathcal{P}_i$ using (2.15)
 2: if $s_i = 0$ then
 3:     update $D = \pi_{\mathcal{P}_i}(\mathcal{D}_i) \cup D_i$
 4:     computes $\mathcal{X}(D)$ as in (2.8)
 5: else
 6:     computes $\overline{\mathcal{X}}(D)$ as in (2.21)
 7: end if
 8: if $s_i = 0$ and condition (2.12) is valid then
 9:     sets $s_i = \beta_i s_{\max}$
10:     broadcasts $D_i$ using (2.14) $R^i_\ell(\mathcal{J}_i)$ distance away
11: else if $s_i \neq 0$ and condition (2.12) is invalid then
12:     update $D = \pi_{\mathcal{P}_i}(\mathcal{D}_i) \cup D_i$
13:     computes $\mathcal{X}(D)$ as in (2.8)
14:     if $t - t^i_i < \Delta_{TB}$ then
15:         sets $\beta_i = \beta_i/2$
16:     end if
17:     if condition (2.12) is valid then
18:         sets $s_i = \beta_i s_{\max}$
19:         broadcasts $D_i$ using (2.14) $R^i_\ell(\mathcal{J}_i)$ distance away
20:     else
21:         sets $s_i = 0$
22:         broadcasts $D_i$ using (2.14) $R^i_\ell(\mathcal{J}_i)$ distance away
23:         waits for a time duration $\tau_d$
24:     end if
25: end if
26: if $\big(t \neq t^i_i\big)$ and $\big($agent $i$ has new potential neighbor or $s_i > s^i_i\big)$ then
27:     broadcasts $D_i$ using (2.14) $R^i_\ell(\mathcal{J}_i)$ distance away
28: end if
29: compute $u_i$ as in (2.13)
30: if $t - t^i_i > \Delta_{TB}$ and $s_i \neq 0$ then
31:     sets $\beta_i = \min(2\beta_i, 1)$
32:     sets $s_i = \beta_i s_{\max}$
33: end if
```

In addition, the initial value of the speed adjustment parameter β_i and the maximum speed $s_{\max}$ were set to $\beta_i = 1$ and $s_{\max} = 0.1m/s$ for all agents, respectively. The density function $\phi(q)$, is provided to all agents and was chosen to be $\phi(q) = e^{-\|x-q_1\|/100} + e^{-\|x-q_2\|/100}$ where $q_1 = (20, 30)$ and $q_2 = (30, 10)$.

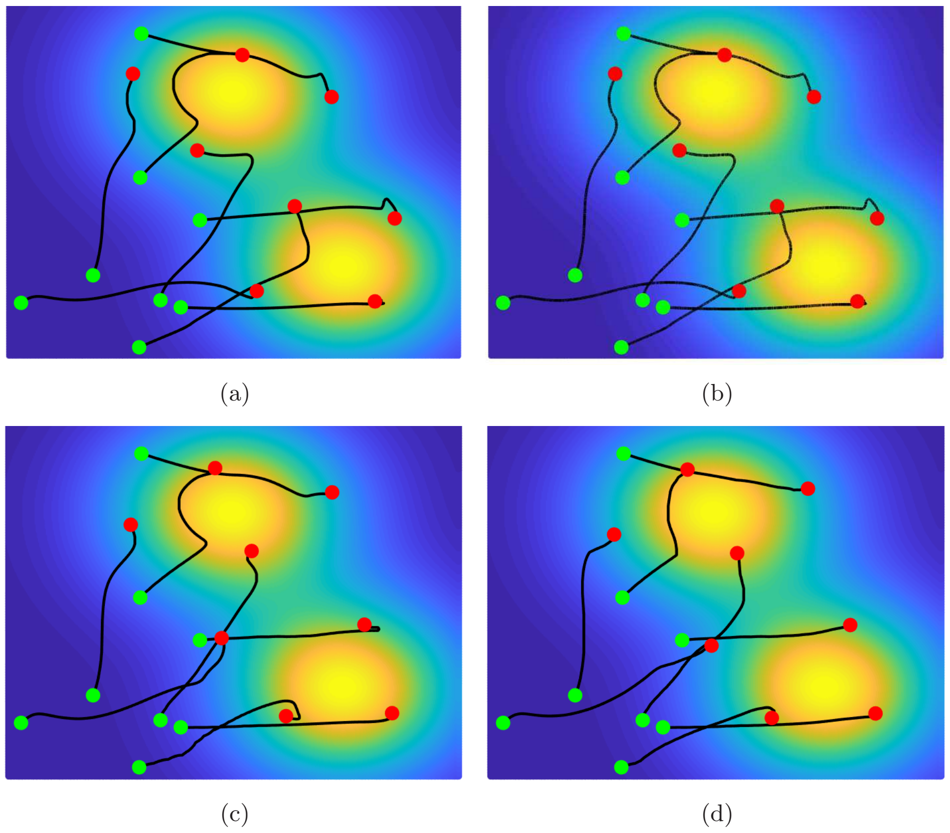


Figure 2.3: Network trajectories of (a) periodic broadcasting algorithm, (b) self-trigger algorithm in [2], (c) `event-triggered broadcasting algorithm` constant $s_{\max}$ speed, and (d) `event-triggered broadcasting algorithm` with variable speed and $\Delta_{TB} = 45/60s$. The green and red dots correspond to the initial and final agent positions, respectively.

2.8.1 Simulation results

The simulation results presented here demonstrate the effectiveness of the `event-triggered broadcasting algorithm` when compared to a periodic broadcasting algorithm where the agents broadcast at every time-step and compared to the self-triggered algorithm in [2]. According to Table 4, in [2] the self-triggered algorithm may require the agents to communicate more than once in the same time instance to ensure the sufficiency of the information. Instead for the self-triggered algorithm, we will assume the agents know their exact Voronoi neighbors at all times and only communicate once when a trigger is occurred. Fig. 2.3

shows the initial and final locations of the agents with their trajectories under periodic broadcasting algorithm, self-triggered algorithm in [2] and our algorithms.

Let us start with Fig. 2.4. This figure shows the algorithms comparison for the convergence of the objective function. It can be seen in Fig. 4 that all algorithms including our algorithm with constant speed $s_{\max}$, with variable speed and $\Delta_{TB} = 45/60s$, the self-trigger algorithm, and the periodic broadcasting algorithm reach similar objective function values. This implies that the reduced communication by our algorithm does not affect the convergence to the set of centroidal Voronoi configurations. In other words, our `event-triggered broadcasting algorithm` preforms as good as the periodic broadcasting algorithm with respect to $\mathcal{H}$.

Additionally, Fig. 2.5 shows the algorithms comparison for the amount of communication between agents. It is clear that our algorithm with constant speed of $s_{\max}$ and our algorithm with variable speed with $\Delta_{TB} = 45/60s$ both significantly reduced the amount of communication between the agents. It is clear that the self-triggered algorithm performs worse than the periodic broadcasting algorithm due to the fact that the self-triggered algorithm requires agents to both request and respond. The amount of communication is greater than the periodic broadcasting algorithm at the $300s$ mark. We would like to note that a request for information is counted as a single communication and each response is counted as a single communication as well. For example, if an agent sent a request and 5 agents respond, this becomes a total of 6 communicated messages. Now the advantage of relaxing the assumption of instantaneous communication and synchronous actions by agents is shown clearly in the comparison between the self-trigger algorithm and our `event-triggered broadcasting algorithm`. To quantify our results, it is noted that for our algorithm with constant speed of $s_{\max}$, the amount of communication between agents is reduced by 63.0% and 78.2% when compared to the periodic broadcasting and self-triggered algorithms, respectively. For our algorithm with variable speed and with $\Delta_{TB} = 45/60s$, the amount of communication between agents is reduced by 97.3% and 98.4% when compared to the periodic broadcasting and self-triggered algorithms, respectively.

Last but not least, our algorithm with constant $s_{\max}$ speed (around the 230s mark) will require periodic communication since condition (2.12) becomes invalid after a time-step for all agents. In addition, our algorithm with variable speed can be seen to address the issue of eventual constant communication as shown in Fig. 2.6 and was able to further reduce the amount of communication as shown in Fig. 2.5. It is notable that in order to see the effectiveness of the `event-triggered broadcasting algorithm` with variable speed, the target time duration Δ_{TB} must be greater than or equal to twice the time-step, $\Delta_{TB} \geq 2\Delta t$. In other words, when $\Delta_{TB} \geq 2\Delta t$, the agents find the appropriate speeds such that they can move at least for $2\Delta t$ without broadcasting. Also as Δ_{TB} increases, the amount of communication is further reduced with a small delay in convergence speed as is seen in Fig. 2.5 and Fig. 2.4 for $\Delta_{TB} = 45/60s$.

In summary, these figures illustrate how the `event-triggered broadcasting algorithm` is able to achieve similar convergence performance to both the periodic broadcasting and self-triggered algorithm while also requiring much less communication between agents. As the figures show, there is a trade-off between the convergence speed and the amount of communication that occurs between agents. With slightly slower convergence speed, our algorithm significantly reduced the amount of communication compared to the periodic broadcasting and the self-triggered cases. The reader may note that given the communication range assignment for the sefl-triggered algorithm in [2], the algorithm will require much more communication than showing in Fig. 2.5.

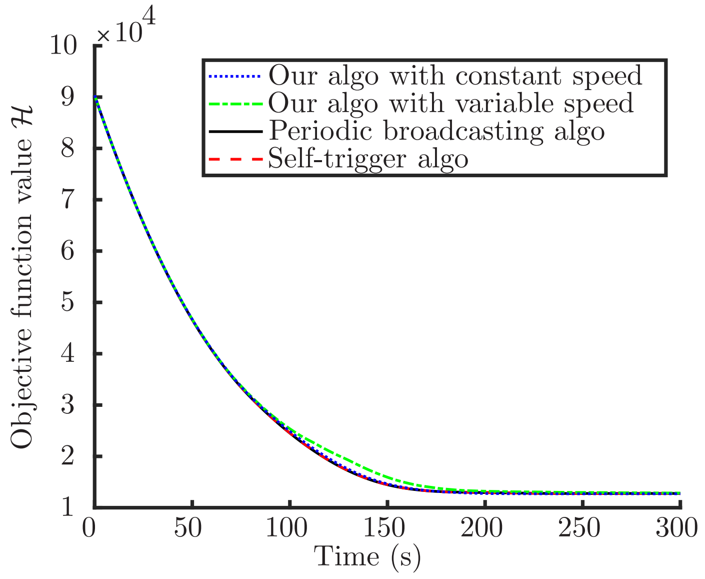


Figure 2.4: A comparison of the objective function value.

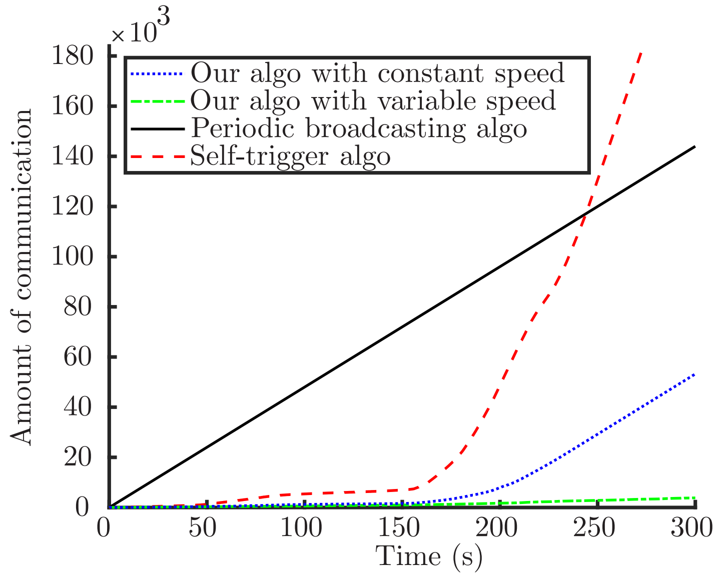


Figure 2.5: A comparison of the amount of communications between the agents.

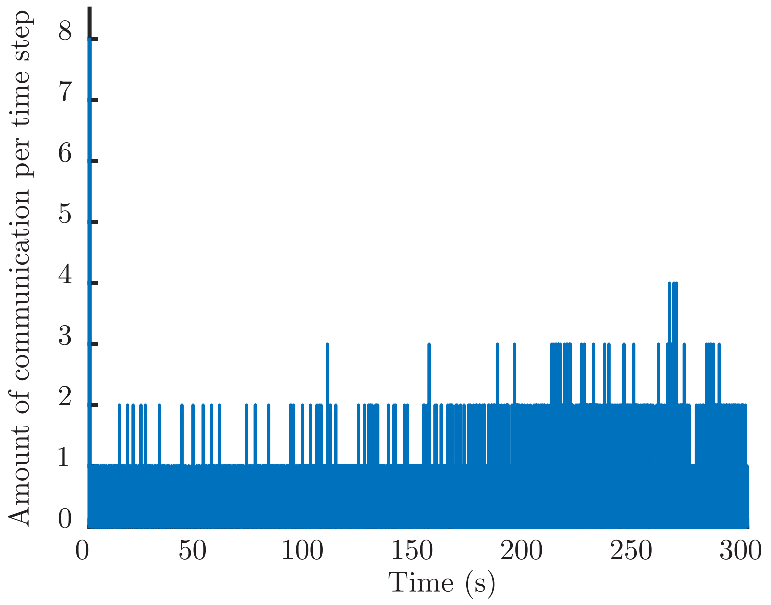


Figure 2.6: The amount of communications per time step under the `event-triggered broadcasting algorithm` with variable speed and $\Delta_{TB} = 45/60$.

2.9 Conclusion and Future Work

This paper proposed a distributed `event-triggered broadcasting algorithm` implemented with a more practical and fully asynchronous broadcasting communication model. The broadcasting communication model relaxes the assumptions of instantaneous communication and synchronous actions by agents. In other words, this means that the communication model does not allow agents to request information from other agents nor acknowledge the reception of messages. Instead, agents must strictly broadcast their information when they decide that it is appropriate. In addition, the agents are capable of determining, prior to transmission, the sufficient broadcasting range to share their information with their potential neighbors i.e. all Voronoi neighbors. Through analysis, the proposed algorithm was shown to provide guaranteed asymptotic convergence. Also, the algorithm was shown to significantly reduce the amount of communication between agents when compared with both a periodic broadcasting strategy and a self-triggered request-response strategy. Future work will focus on a modification of the broadcast deployment problem where the possibility of packet drops may occur. We aim to use the possible and potential neighbors and the communication range assignment method presented here to allow agents to not only determine when to communicate, but also how to select a channel to communicate over in order to minimize packet loss.

Chapter 3: Voronoi Partitioning with Uncertainty: Theory and Applications

Abstract

Voronoi tessellations or Voronoi diagrams have been used in a variety of applications for decades now, and in many applications, it is assumed that the locations of the generator points' are precisely known. This assumption might be acceptable for some applications, but for sure not for all. Our goal is to establish generalized definitions and concepts of when the locations of the cells' generator points are imprecise for N-dimensional Voronoi diagrams, which will be applicable to all types such as, but not limited to, multiplicatively weighted Voronoi and power diagrams. Our definitions and concepts will be useful to define the lower- and upper-bound of the true Voronoi cell, and also determine what information is required to compute the bounds, and establish a guaranteed set that contains the true Voronoi centroid given the bounds. Finally, we illustrate, with a case study, how our work can be adapted to improve existing works using Voronoi diagrams with imperfect generator point locations.

3.1 Introduction

A few hundred years ago, in 1644, the first Voronoi like diagram was observed in literature by the work of René Descartes to show the disposition of matter in the solar system [11]. Almost two centuries later, the mathematician Gustav Lejeune Dirichlet formally introduced the Dirichlet tessellation [51], which is another name for the Voronoi tessellation, for second and third dimensional space. Then, a few decades later, the mathematician Georgy Voronoi introduced the Voronoi tessellation for higher order dimensional space [52], and since then, it has been widely known as the Voronoi diagram or Voronoi partition [11]. In addition, during the 19th century, the Voronoi tessellation was re-discovered several times and was used in a variety of disciplines by different scientists, which established new names. These names are presented in the Table 3.3 and Table 3.4 in Section 3.2, and we hope that we have included all the names.

A Voronoi diagram is a collection of Voronoi cells, where each Voronoi cell is generated by a given *Voronoi generation distance function* with respect to the generator points. From the early 80s, the Voronoi diagram has been applied and observed in a vast number of different disciplines all the way from Anthropology to Zoology [11, 53]. In Anthropology, the Voronoi diagram is used to describe different cultures' influence on a region [54]. In Zoology, the Voronoi diagram is used to model and analyze the territories of animals [55].

A common question among researchers is often how to deal with uncertainty. Unfortunately, given the extremely wide applicability of Voronoi partitions, there are many similar results and methods that appear in different contexts with different terminology, although they are ultimately the exact same thing. For example, when there is uncertainty in the locations of the generator points a common desire is to determine the subsets or supersets of the true Voronoi cells. Table 3.1 presents the names of the Voronoi cell subsets used across different disciplines, and Table 3.2 presents the names of the Voronoi cell supersets used across different disciplines. In this paper, we will use the term guaranteed Voronoi cell for the subset of the Voronoi cell and use the term possible Voronoi cell for the superset of the Voronoi cell.

The two tables focus on the type of Voronoi diagram used and the general class of problem solved, and each cited paper might consider different assumptions and/or constraints. As the reader may notice that the problems solved with the guaranteed and possible cells are few. However, this does not mean that it is not useful to solve other problems. In fact, in the field of bioinformatics or system biology, one computational geometry approach to protein structure is implemented using a Voronoi diagram [82–85]. In this approach, it is assumed that the location of the generator points is precise. According to the protein data

Table 3.1: Guaranteed Voronoi diagrams summary

Type of Voronoi Diagram	Subset Name	Problem Solved?	Citation
Ordinary Voronoi	Fuzzy Voronoi cell	Efficient cell computation	[45, 56]
Ordinary Voronoi	Guaranteed Voronoi cell	Efficient cell computation	[44]
Ordinary Voronoi	Guaranteed Voronoi cell	Facility static location	[2,57–60]
Ordinary Voronoi	Guaranteed Voronoi cell	Maximum area coverage	[61, 62]
Additively weighted Voronoi	Guaranteed Voronoi cell	Maximum area coverage	[63–68]
Multiplicatively weighted Voronoi	Guaranteed Voronoi cell	Facility static location	[3]
Multiplicatively weighted Voronoi	Guaranteed Voronoi cell	Maximum area coverage	[65, 69]
Power diagram	Guaranteed Voronoi cell	Facility static location	[70–72]
Power diagram	Guaranteed Voronoi cell	Maximum area coverage	[69, 73]

bank, the generator points are bounded inside a sphere, where the Voronoi uncertainties are referred to as ”resolutions”. We believe that considering the uncertainties in the protein computational geometry may establish new structures.

Statement of Contributions: In this paper, we formalize and compile various scattered notions and tools for Voronoi partitioning with uncertain information about the locations of the generators points. By doing so, we extend various ideas developed in isolated areas not only to different application areas, but also to generalized Voronoi diagrams. More specifically, we introduce the notion of a guaranteed Voronoi diagram, possible Voronoi diagram, possible neighbors and guaranteed Voronoi Centroid sets in N-dimensional space. We also, present a novel notion that is the guaranteed neighbor. Finally, we show how the results of the paper can be used across various application domains through the use of a case study and examples.

Table 3.2: Possible Voronoi diagrams summary

Type of Voronoi Diagram	Superset Name	Problem Solved?	Citation
Ordinary Voronoi	Dual-guaranteed Voronoi cell	Facility static location	[2,59]
Ordinary Voronoi	Nonzero Voronoi cell	K-Nearest-neighbor query	[74]
Ordinary Voronoi	Possible Voronoi cell	Efficient cell computation	[75,76]
Ordinary Voronoi	Possible Voronoi cell	K-Nearest-neighbor query in multi-dimensional space	[77]
Ordinary Voronoi	Uncertain Voronoi cell	Efficient cell computation	[78]
Ordinary Voronoi	Uncertain Voronoi cell	K-Nearest-neighbor query	[79,80]
Compoundly weighted Voronoi	Weighted imprecise Voronoi cell	K-Nearest-neighbor query and Efficient cell computation	[81]

3.2 Preliminaries

In this section, we start by presenting the names of the Voronoi diagrams, and we hope to have obtained them all in Table 3.3. Also, in Table 3.4, we present other known names of specific types of Voronoi diagram which will be introduced formally in Section 3.2.2.

3.2.1 Notation

Let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$ and $\mathbb{Z}_{\geq 0}$ denote the sets of real, non-negative real and non-negative integer numbers, respectively, and $|\cdot|$ and $\|\cdot\|$ denote the cardinality of a set and the Euclidean distance of a vector, respectively. We let Q to be the space of the Voronoi diagram in $\mathbb{R}^n$. Also, we let $\phi : Q \to \mathbb{R}_{\geq 0}$ to be a probability density function that maps the probability of an action taking place at each point in Q. The *mass* and *center of mass* of Q are defined with respect to ϕ as

$$M_Q = \int_Q \phi(q)dq \quad \text{and} \quad C_Q = \frac{1}{M_Q}\int_Q q\phi(q)dq,$$

respectively. Also, we let a closed set centered at q with a radius r be $\overline{B}(q,r)$.

Table 3.3: Other known names of the Voronoi diagram

Voronoi's Name	Citation
Voronoi diagram	[11]
Voronoi partition	
Voronoi honeycomb	[86]
Voronoi foam	[87]
Voronoi medial axis	[88]
Thiessen polygons	[89, 90]
Wigner-Seitz cells	[90, 91]
Capillary domains	[92, 93]
Voronoi decomposition	[94]
Wirkungsbereich which means (domain of action, field of activity, area of influence)	[11]
Domain of an atom	
Area potentially available	
Plant polygons	

3.2.2 Spatial Partitioning without Uncertainties

Here, we review all the spatial partitioning techniques we are interested in with perfect information. Let $Q \subset \mathbb{R}^N$ be a space that we wish to partition based on the generator points locations $P = \{p_1, \dots, p_N\} \in Q^N$, with identification numbers $\mathcal{I} = \{1, \dots, N\}$, and weights $W = \{w_1, \dots, w_N\} \in (\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \times \mathbb{R})^N$. Based on this data, we present the Voronoi generation assignment rule, Voronoi generation distance function or node function similar to [11, 97, 98] as

$$f(q, p, w_i) = \frac{1}{\alpha_i} \|q - p\|^{\beta_i} - \gamma_i. \tag{3.1}$$

Based on the different parameters α_i, β_i, and γ_i, different types of Voronoi diagrams can be constructed. Table 3.5 presents the well known types of Voronoi diagram.

Definition 3.1 (Voronoi cell)**.** Given the set of generator points P and weights W, the

Table 3.4: Other names of the weighted Voronoi diagram

Voronoi's Type Name	Another Voronoi Name	Citation
Additively weighted power distance Voronoi diagram	Power diagram and Radical tessellation and Laguerre diagram	[11]
	Dirichlet cell complex	[95]
	Sectional Dirichlet tessellation	[96]
Additively weighted Voronoi diagram	Hyperbolic Dirichlet tessellation	[11]
Multiplicatively weighted Voronoi diagram	Circular Dirichlet tessellation and Apollonius model	[11]

Voronoi cell for site i is given by

$$V_i(P, W) = \{q \in Q \mid f(q, p_i, w_i) \leq f(q, p_j, w_j) \; \forall \; j \in \mathcal{I} \setminus \{i\}\},$$

Now, we present a Voronoi diagram similar to [11].

Definition 3.2 (Voronoi diagram)**.** The Voronoi diagram is a collection of Voronoi cells such as

$$\mathcal{V}(P, W) = \bigcup_{i \in \mathcal{I}} V_i(P, W), \tag{3.2}$$

In addition, according to [99], the agents at p_i and p_j are called Voronoi neighbors if their cell intersections satisfy $V_i \cap V_j \neq \emptyset$. Also, it is known that the Voronoi neighbors' information of generator point i is sufficient to compute the ith Voronoi cell. In this paper, we define the neighbor identification number set for agent i as $\mathcal{N}_i$.

Table 3.5: Types of the Voronoi diagrams for different weight constraints

α_i	β_i	γ_i	Type of Diagram
1	1	0	Ordinary Voronoi diagram
1	1	$\in \mathbb{R}$	Additively weighted Voronoi diagram
> 0	1	0	Multiplicatively weighted Voronoi diagram
> 0	1	$\in \mathbb{R}$	Compoundly weighted Voronoi diagram
1	> 1	$\in \mathbb{R}$	Additively weighted power distance Voronoi diagram

3.3 Problem Statement

We are now interested in techniques for dealing with uncertainty about a generator points or agents location. As before, let $P = \{p_1, \ldots, p_N\} \in Q^N$, $\mathcal{I} = \{1, \ldots, N\}$ and $W = \{w_1, \ldots, w_N\} \in (\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \times \mathbb{R})^N$ be the sets of N agent's locations, identification numbers and weights, respectively. Also, let $\mathcal{N} = \{\mathcal{N}_1, \ldots, \mathcal{N}_N\} \subset \mathcal{I}^N$ be the neighboring agents sets of each agent. However, the true positions P are no longer known, instead, let $\hat{P} = \{\hat{p_1}, \ldots, \hat{p_N}\} \in Q^N$ be the estimated generator locations satisfying $\|p_i - \hat{p_i}\| \leq r_i \ \forall \ i \in \mathcal{I}$, and let the set of errors be $R = \{r_1, \ldots, r_N\} \in \mathbb{R}^N_{\geq 0}$. Given the estimated positions $\hat{P}$ and errors R, we define an uncertainty region for each agent i as $X_i = \overline{B}(\hat{p}_i, r_i)$ that is a closed set centered at $\hat{p_i}$ with radius r_i. In case $Q \subset \mathbb{R}^2$, the closed set is a closed ball and in case $Q \subset \mathbb{R}^3$, the closed set is a closed sphere. Now, we define the set of all uncertainty regions $\mathcal{X} = \{X_1, \ldots, X_N\} \subset Q^N$.

The first objective of this work is to define the biggest set of points in Q, that is guaranteed to be within the true Voronoi cell given the uncertainties and define the smallest set of points that is guaranteed to contain the true Voronoi cell given the uncertainties. In other words, we aim to determine the lower- and upper-bound of the true Voronoi cell given the uncertainties presented in $\mathcal{X}$. Let the lower-bound set be the guaranteed Voronoi cell gV_i, and let the upper-bound set be the possible Voronoi cell pV_i. Our goal is to determine the guaranteed and possible cells such as

$$gV_i(\mathcal{X}, W) \subset V_i(P, W) \subset pV_i(\mathcal{X}, W), \tag{3.3}$$

The second objective of this work is to formally define the set of guaranteed neighbors $g\mathcal{N}_i$ and the set of possible neighbors $p\mathcal{N}_i$ for all $i \in \mathcal{I}$ given the uncertainty sets. The guaranteed neighbors of agent i are the neighboring agents that are guaranteed to be Voronoi neighbors regardless of the uncertainties, and the possible neighbors of agent i are the neighboring agents that can be Voronoi neighbors due to the uncertainties. These neighboring agents are known when the uncertainties are singletons. For instance, in an ordinary Voronoi diagram, these points are called Voronoi neighboring points. For this problem, determining these points is challenging. Our goal is to determine the guaranteed and possible neighbors such as

$$g\mathcal{N}_i \subset \mathcal{N}_i \subset p\mathcal{N}_i, \tag{3.4}$$

Additionally, we are looking to establish a condition on $p\mathcal{N}_i$ so that any agent $j \notin p\mathcal{N}_i$ will be sufficiently far away from point i so that the computation of $gV_i(\mathcal{X}, W)$ and $pV_i(\mathcal{X}, W)$ will not be impacted.

The last objective of this work, given the guaranteed and possible Voronoi cells, is to establish a guaranteed set for the centroid of the true Voronoi cell. Let the possible centroid set be $\mathcal{C}_{V_i}$ so that the true Voronoi centroid C_{V_i} must be contained in $\mathcal{C}_{V_i}$. Formally,

$$C_{V_i(P,W)} \in \mathcal{C}_{V_i(\mathcal{X},W)} \tag{3.5}$$

Problem 3.3. Given the sets $\hat{P}$ and R, we determine a lower- and upper-bound on the

Voronoi cell so that $gV_i(\mathcal{X}, W) \subset V_i(P, W) \subset pV_i(\mathcal{X}, W)$; and we determine the set of guaranteed and possible neighbors so that $g\mathcal{N}_i \subset \mathcal{N}_i \subset p\mathcal{N}_i$; we also determine the guaranteed set for the centroid of the Voronoi cell so that $C_{V_i(P,W)} \in \mathcal{C}_{V_i(\mathcal{X},W)}$.

3.4 Space Partition with Uncertainties

In this section, we aim to establish the generalized guaranteed and generalized possible Voronoi cells and diagrams given the uncertainty set $\mathcal{X}$. To do so, we start by presenting the bisector (boundary) between two agents estimated locations $\hat{p}_i$ and $\hat{p}_j$. The guaranteed Voronoi bisector between two agents estimated locations $\hat{p}_i$ and $\hat{p}_j$ is defined as

$$gV_{i,j} = \{q \in Q \mid \max_{x_i \in X_i} f(q, x_i, w_i) \leq \min_{x_j \in X_j} f_j(q, x_j, w_j)\},$$

and the possible Voronoi bisector between between two agents estimated locations $\hat{p}_i$ and $\hat{p}_j$ is defined as

$$pV_{i,j} = gV_{j,i} = \{q \in Q \mid \min_{x_i \in X_i} f(q, x_i, w_i) \leq \max_{x_j \in X_j} f_j(q, x_j, w_j)\}.$$

Now, we can formally define $gV_i(\mathcal{X}, W)$ and $pV_i(\mathcal{X}, W)$ given the uncertainties of all agents.

Definition 3.4 (Generalized guaranteed Voronoi cell)**.** The guaranteed Voronoi cell is formally defined as

$$gV_i(\mathcal{X}, W) = \bigcap_{i \in \mathcal{I}} gV_{i,j}. \tag{3.6}$$

Definition 3.5 (Generalized possible Voronoi cell)**.** The possible Voronoi cell is formally defined as

$$pV_i(\mathcal{X}, W) = \bigcap_{i \in \mathcal{I}} pV_{i,j}. \tag{3.7}$$

The definitions of the Guaranteed and Possible Voronoi cells proposed above serve all types of Voronoi diagrams in N-dimensional space which allow us to establish the generalized notion and tools for all diagrams. Also, to the best of our knowledge, none of the cited papers have proposed a generalized definition for these cells.

By Definition 3.4 and 3.5, the guaranteed and possible Voronoi diagrams are defined as the following.

Definition 3.6 (Guaranteed Voronoi diagram)**.** The guaranteed Voronoi diagram is the collection of all guaranteed Voronoi cells such as

$$g\mathcal{V}(\mathcal{X}, W) = \bigcup_{i \in \mathcal{I}} gV_i(\mathcal{X}, W). \tag{3.8}$$

Definition 3.7 (Possible Voronoi diagram)**.** The possible Voronoi diagram is the collection of all possible Voronoi cells such as

$$p\mathcal{V}(\mathcal{X}, W) = \bigcup_{i \in \mathcal{I}} pV_i(\mathcal{X}, W). \tag{3.9}$$

Every point *inside* the guaranteed cell $gV_i(\mathcal{X}, W)$ is closer to agent i given the Voronoi generation distance function in (3.1) than the other agents in $\mathcal{I} \setminus \{i\}$. Because of this, as can be seen in Figure3.1.(a), there are points that are not assigned to any agents (neutral points) in Q due to the uncertainties presented in set $\mathcal{X}$, which conclude that these points are not guaranteed to be closer to any agent than the others. Also, it is obvious from the figure that $g\mathcal{V}(\mathcal{X}, W)$ is not a partition of Q. Moreover, given the nature of the guaranteed cells, the bigger the uncertainty regions, the smaller the guaranteed Voronoi cell, and the guaranteed Voronoi cell can be an empty set if $X_i \cap X_j \neq \emptyset$ for any $j \in \mathcal{I} \setminus \{i\}$.

On the other hand, every point *outside* the possible cell $pV_i(\mathcal{X}, W)$ is guaranteed to be closer to one of the other agents in $\mathcal{I} \setminus \{i\}$ than i, given the Voronoi generation distance function in (3.1). In addition, as it can be seen in Figure 3.1.(b), the union of the collection of $pV_i(\mathcal{X}, W)\ \forall\ i \in \mathcal{I}$ is Q but it does not necessarily have a disjointed interior. With that being said, the points in set $[pV_i(\mathcal{X}, W) \setminus gV_i(\mathcal{X}, W)]\ \forall\ i \in \mathcal{I}$ are the neutral points, and for agent i, each point in the set $[pV_i(\mathcal{X}, W) \setminus gV_i(\mathcal{X}, W)]$ can be closer to itself than the others for a configuration of the agents' positions in their respective uncertainty regions. In addition, given the nature of the possible cells, the bigger the uncertainty regions, the bigger the possible Voronoi cell.

Remark 3.8. Interestingly, [66,74] noted that the guaranteed and possible ordinary Voronoi bisector between point i and j can be computed as an additively weighted Voronoi bisector where $\gamma_i = \pm r_i$.

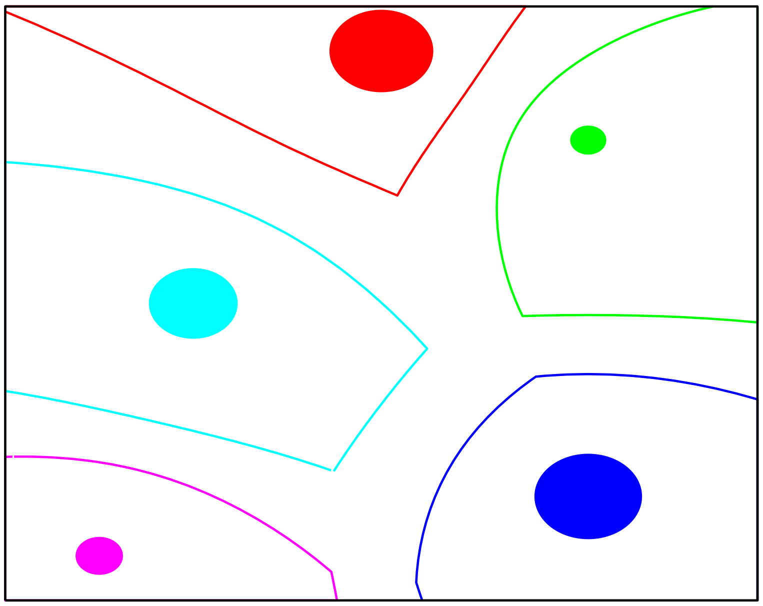
3.5 Guaranteed and Possible Neighbors

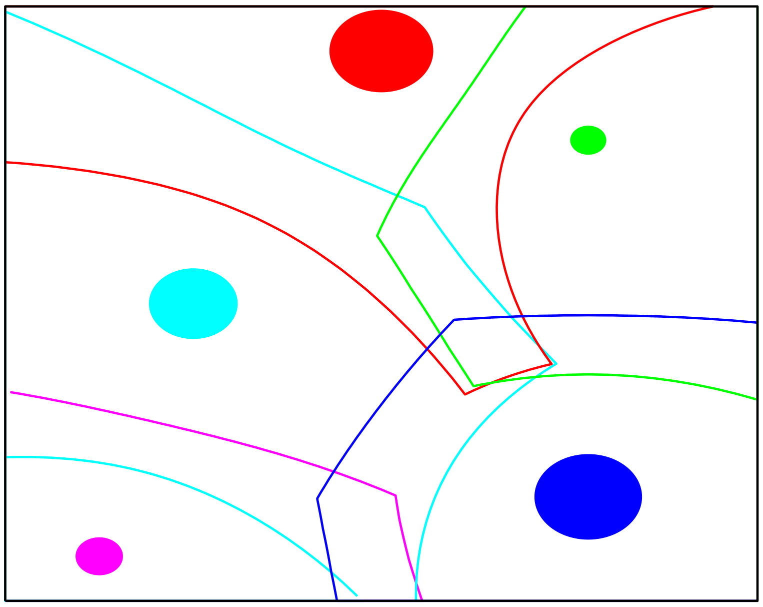
In this section, we aim to formally define the set of guaranteed neighbors $g\mathcal{N}_i$ and possible neighbors $p\mathcal{N}_i$ for the ith agent given the uncertainties presented by $\mathcal{X}$. Let us start by intuitively defining the sets. The set $g\mathcal{N}_i$ contains the identification numbers of all agents that are Voronoi neighbors for any configuration of $p_j \in X_j\ \forall\ j \in \mathcal{I}$. In other words, we are looking to determine all agents that are always affecting guaranteed and possible Voronoi cells. On the other hand, the set $p\mathcal{N}_i$ contains the identification number of all agents that can be Voronoi neighbors for at least one configuration of $p_j \in X_j\ \forall\ j \in \mathcal{I}$. In other words, we are looking to determine any agent that may shrink or expand the guaranteed and possible Voronoi cells. We define the guaranteed and possible neighbors as the following.

Definition 3.9 (Guaranteed neighbor)**.** Given uncertainty set $\mathcal{X} = \{X_1, \ldots, X_N\}$ so that $p_j \in X_j$ for all $j \in \mathcal{I}$, any two agents that are Voronoi neighbors for all configuration of $p_j \in X_j\ \forall\ j \in \mathcal{I}$ are guaranteed neighbors.

The definition is formally presented by the following lemma.

Lemma 3.10 (Guaranteed neighbors)**.** Given uncertainties set $\mathcal{X}$ so that $p_j \in X_j$ for all





(b)

Figure 3.1: (a) Guaranteed and (b) possible multiplicatively weighted Voronoi digrams in $\mathbb{R}^2$.

$j \in \mathcal{I}$, if

$$\left((pV_i(\mathcal{X}, W) \cap pV_j(\mathcal{X}, W)) \setminus \bigcup_{k \in \mathcal{I} \setminus \{i,j\}} pV_k(\mathcal{X}, W) \right) \bigcup \left(gV_i(\mathcal{X}, W) \cup gV_j(\mathcal{X}, W) \right)$$

is simply connected, then agent j is a guaranteed Voronoi neighbor of agent i.

Proof. In Appendix F $\square$

Definition 3.11 (Possible neighbor)**.** Given an uncertainty set $\mathcal{X} = \{X_1, \dots, X_N\}$ so that $p_j \in X_j$ for all $j \in \mathcal{I}$, any two agents that can be Voronoi neighbors for at least one configuration of $p_j \in X_j \; \forall \; j \in \mathcal{I}$ are possible neighbors.

The definition is formally presented by the following lemma.

Lemma 3.12 (possible neighbors)**.** Given uncertainties set $\mathcal{X}$ so that $p_j \in X_j$ for all $j \in \mathcal{I}$, if

$$pV_i(\mathcal{X}, W) \cap pV_j(\mathcal{X}, W) \neq \emptyset,$$

then agent j is a possible neighbor of agent i.

Proof. In Appendix G $\square$

Now, we can redefine the guaranteed and possible Voronoi cells given $p\mathcal{N}_i$ since any $j \notin p\mathcal{N}_i$ will not impact the structure of $gV_i(\mathcal{X}, W)$ and $pV_i(\mathcal{X}, W)$ as

$$gV_i^{p\mathcal{N}_i} = \bigcap_{i \in p\mathcal{N}_i} gV_{i,j}, \tag{3.10a}$$

$$pV_i^{p\mathcal{N}_i} = \bigcap_{i \in p\mathcal{N}_i} pV_{i,j}, \tag{3.10b}$$

Furthermore, given Lemma 3.12, we aim to establish a condition for the possible neighbors such that any agent outside a closed set centered at $\hat{p}_i$ cannot be a possible neighbor to agent i.

Corollary 3.13 (Condition on possible neighbors)**.** Given the uncertainties $X_j\ \forall\ j \in \{i, \underline{p\mathcal{N}_i}\}$ where $\underline{p\mathcal{N}_i} \subset p\mathcal{N}_i$, if

$$p_k \notin \overline{B}(\hat{p}_i, N_i),$$

then $k \notin p\mathcal{N}_i$ is guaranteed, where

$$N_i = \max_{j \in \underline{p\mathcal{N}_i}} \left(\max_{x_i \in X_i} \left\| q' - x_i \right\| + \sqrt[\beta_j]{\alpha_j \left(\frac{1}{\alpha_i} \min_{x_i \in X_i} \|q' - x_i\|^{\beta_i} - \gamma_i + \gamma_j \right)} \right), \tag{3.11}$$

for $q' = \max_{\substack{x_i \in X_i, \\ q \in pV_i^{p\mathcal{N}_i}}} (\|q - x_i\|)$.

Proof. In Appendix H $\square$

Remark 3.14. If agent j is not a possible neighbor of agent i, it cannot be a Voronoi neighbor of the agent i due that the fact that $pV_i^{p\mathcal{N}_i} \cap pV_j^{p\mathcal{N}_i} = \emptyset \Rightarrow V_i(P, W) \cap V_j(P, W) = \emptyset$.

3.6 Possible Voronoi Centroid Set

In this section, we are looking to provide a generalized form for the Possible Voronoi Centroid set. Let us start by defining the set.

Definition 3.15 (Possible Voronoi centroid set)**.** Given the uncertainty set $\mathcal{X} = \{X_1, \ldots, X_N\}$ so that $p_j \in X_j$ for all $j \in \mathcal{I}$, the possible Voronoi centroid set for agent i is all possible Voronoi centroids for any configuration of $p_j \in X_j\ \forall\ j \in \mathcal{I}$.

This is not trivial to accomplish, but we can establish an upper-bound to guarantee that the true Voronoi centroid is within the set. Our earlier work presented in [59] formalized the Possible Voronoi Centroid set for ordinary Voronoi diagrams. In fact, the result can be used in this work.

Proposition 3.16. [59, Proposition 4.4] Let $L \subset V \subset U$. Then, for any density function ϕ,

$$C_V \in \mathcal{C}_V = \overline{B}(C_L, \mathcal{B}) \cap \overline{B}(C_U, \mathcal{B}),$$

where $\mathcal{B} = 2cr_U \left(1 - \frac{M_L}{M_U}\right)$ and cr_U is the radius of the smallest closed ball that contains the set U.

Let $L = gV_i^{p\mathcal{N}_i}$ and $U = pV_i^{p\mathcal{N}_i}$ in Proposition 3.16. The Possible Voronoi Centroid set is properly defined. Figure 3.2 demonstrates the proposition in $\mathbb{R}^2$ space. Finally, from Proposition 3.16, we establish the following bound

$$\|p - C_V\| \leq \arg\max_{q \in \mathcal{C}_V} \|p - q\|.$$

3.7 Case study

In this section, our goal is to demonstrate the use of the developed concepts in this paper in an established work. For the case study, we adapt our concepts to re-do the problem in [3]. The authors used the guaranteed multiplicatively weighted Voronoi diagram to solve the problem of deploying N mobile sensors with different health conditions and communication delays. In this problem, the communication delays generate fixed uncertainty regions regarding the actual positions of the sensors. Given a maximum velocity $v_{\max}$ and the communication delays τ_i, each mobile sensor is represented by a closed ball $X_i = \overline{B}(\hat{p}_i, r_i)$ centered at its last known location $\hat{p}_i$ with a radius $r_i = v_{\max}\tau_i$. Additionally, the goal is to minimize the cost function

$$\mathcal{H}(P, W) = \sum_{i=1}^{N} \int_{V_i} F_i(q, \alpha_i)\ \phi(q)\ dq,$$

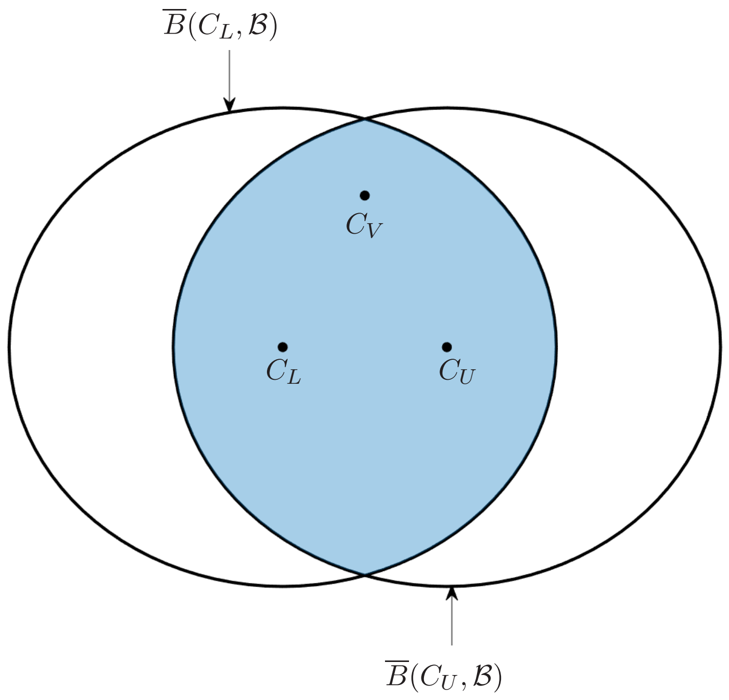

$\overline{B}(C_L, \mathcal{B})$
C_V
C_L
C_U
$\overline{B}(C_U, \mathcal{B})$

Figure 3.2: Example of possible Voronoi centroid set colored in blue in $\mathbb{R}^2$ space.

where $F_i(q, \alpha_i) = \|p_i - q\|^2$ is a measure of the performance function.

Given the performance function, the health condition is represented by $\alpha_i \in w_i$ for all sensors where $\beta_i = 1,\ \gamma_i = 0$. The the cost function using the multiplicatively weighted Voronoi diagram becomes

$$\mathcal{H}(P, W) = \sum_{i=1}^{N} \int_{V_i^m} \frac{1}{\alpha_i} \|p_i - q\|^2 \ \phi(q)\ dq,$$

where V_i^m is the multiplicatively weighted Voronoi (MWV) cell. The authors drove the mobile sensor to the centroid of the guaranteed MWV cell since the Voronoi cell is unknown. Also, we would like to note that, even though the authors required the sensors to find their Voronoi neighbors, they did not provide a method to determine or establish them, which is lacking in the original work.

Before we adapt our work to resolve the problem, we would like to note that this is a special case where the mobile sensor knows its exact location, $X_i = \{p_i\}$, and a closed form equation for the guaranteed and possible MWV bisectors are provided in Appendix I. We start by letting the possible neighbor of agent i set be $p\mathcal{N}_i^m$. The possible neighbors are sensors that sensor i needs information from to compute the guaranteed and possible MWV cells denoted by gV_i^m and pV_i^m respectively. In addition, the condition established in Corollary 3.13 helps determining a communication range on-line for a distributed algorithm, which is lacking in the original work. Knowing that the all the possible neighbors must be in a closed ball centered at $\hat{p}_i$ with a radius of N_i allows us to set the communication range to N_i. The N_i in (3.11) can be upper-bounded and simplified for the MWV diagram as

$$N_i^m \leq \max_{q \in pV_i^m} \left(\|q - p_i\|\right) \left(\max_{j \in p\mathcal{N}_i^m} \left(\frac{\alpha_j}{\alpha_i} \right) + 1 \right). \tag{3.12}$$

Furthermore, it is known from [97] that a local minima of $\mathcal{H}$ can be reached by driving the agents to the centroid of MWV cell. Therefore, we developed a motion control law that drives the agents as close as possible to the centroid of MWV. The motion control drives the sensor to the centroid of $\mathcal{C}_{V_i(\mathcal{X},W)}$. Formally,

$$u_i = v_{\max} \frac{p_i - m_i}{\|p_i - m_i\|}, \tag{3.13}$$

where $m_i = \left(C_{gV_i^m} + C_{pV_i^m}\right)/2$.

Figure 3.4 compare the mobile sensors' trajectories without communication delays, with communication delays as in [3], and with communication delays and motion control law u_i in (3.13), and Figure 3.3 compare the objective function $\mathcal{H}$ values of the three trajectories. As it can be seen from Figure 3.3, both solutions achieve similar cost values. Our algorithm slightly reduces the cost function due to the fact that the agents move as close as possible to the centroids of the true Voronoi cells. Finally, without the possible neighbors, the mobile sensors would not know their neighbors and their communication range; this makes our algorithm fully decentralized and distributed.

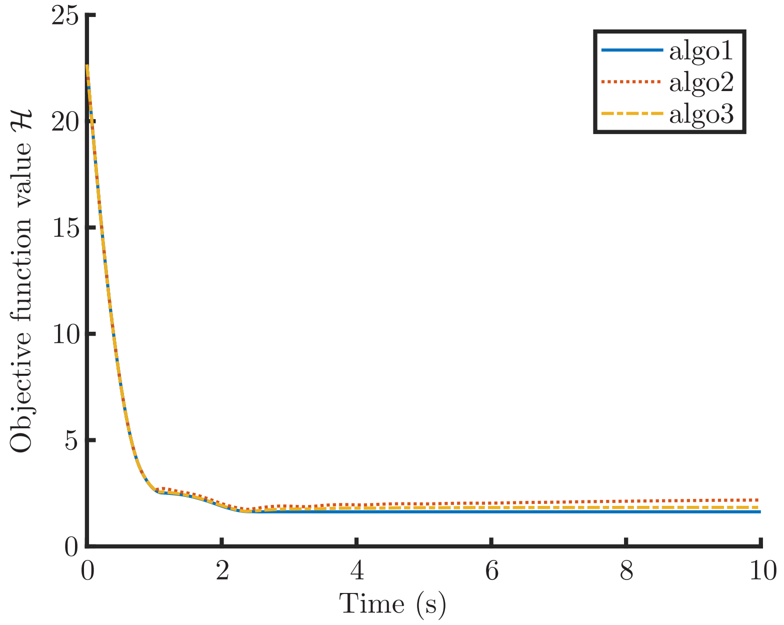
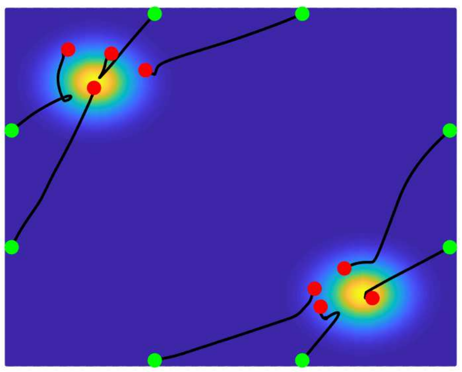
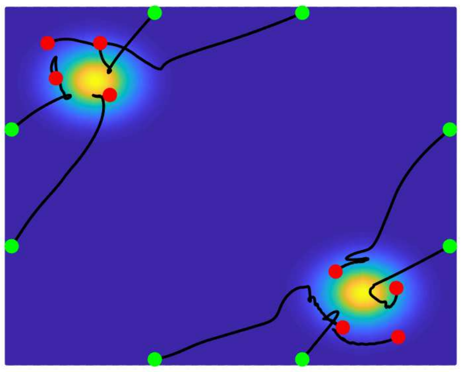


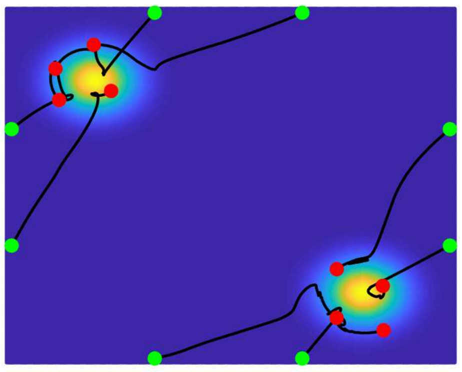
Figure 3.3: A comparison of the objective function value between algo1: without communication delays, algo2: with communication delays as in [3] and algo3: with communication delays and motion control law u_i in (3.13).



(a)



(b)



(c)

Figure 3.4: Mobile sensors trajectories of (a) without communication delays, (b) with communication delays as in [3], and (c) with communication delays and motion control law u_i in (3.13). The green and red dots correspond to the initial and final agent positions, respectively.

3.8 Extended Example

In this section, our goal is to show how the developed concepts in this paper can be useful for established work. The first example relates to the preservation of privacy,and the second example relates to the computational geometry of a protein.

3.8.1 Example 1

In this subsection, we will demonstrate how possible Voronoi cells improve on the solution provided by [100] for the privacy preserving problem. The problem presented in the cited paper has three components: 1) a rider who requests a ride; 2) a driver who responds to a request; 3) a service provider (SP) that manages the requests and responses.

The problem addressed and solution proposed are briefly presented as follows. The SP has continued access to the drivers' locations and computes the Voronoi diagram using the drivers' locations as generator points. When a rider makes a request for a ride, it will appear in one of the drivers Voronoi cells. If the rider requests the closest driver, it is clear that the rider's location is within the requested driver's Voronoi cell (no privacy preserving). Instead, the rider specifies a privacy preference so that it requests a driver within a S region. Therefore, the goal is to match the rider with a driver inside the S region that minimizes the waiting time for a specified weight factor w on the ride matching accuracy.

We would like to note two observations on the proposed algorithm. 1) In November 2014, Uber opened an internal investigation regarding an employee who was tracking a rider without permission [101]. This raises privacy concerns for drivers sharing their precise locations with a SP when they do not have a rider. 2) If we consider a case where the drives are moving, the Voronoi diagram will continually change, which will impact the solution proposed.

These two observations can be addressed by constructing closed balls (uncertainties) that guarantee to contain the drivers' locations. Then, the solution will use possible Voronoi cells instead of Voronoi cells. In addition, we would like to note that the work in [74] may be useful to compute the probability of the nearest drivers with uncertainties about a rider's location.

3.8.2 Example 2

In this subsection, we will show how guaranteed and possible neighbors may be useful for the computational geometry of protein structures. According to the authors in [82–85], the relation between agents (particles) is established based on whether they are Voronoi neighbors or not, and it is assumed that they know the exact locations of the generator points used to create the diagram. As noted earlier, according to the protein data bank, the generator points are bounded inside a sphere where the Voronoi uncertainties are referred to as "resolutions", thus, given the uncertainties of the generator points, new relations between the agents can be discovered.

From our work, the following can be accomplished; The guaranteed neighbor set of agent i will guarantee that there is always a relation between agent i and its guaranteed neighbors, and the possible neighbor set of agent i may establish a possible relation between two agents that had not been considered before. The new relations between the agents may result in new protein structures to cure diseases or alleviate symptoms.

3.9 Conclusion and Future Work

This work has generalized the definitions and concepts related to subsets and supersets of Voronoi cells when the locations of cells' generator points are imprecise. The guaranteed (subset) and possible (superset) cells proposed in this paper are formally defined for generalized Voronoi diagrams (any type of Voronoi diagram) in N-dimensional space. Also, we introduced the guaranteed and possible neighbors whose information is required to compute guaranteed and possible cells. Moreover, we have developed a concept of possible Voronoi centroid sets that guarantees to contain the true Voronoi centroid for any configuration of generator points inside their respective uncertainty regions. Finally, our future work will be an extension of this work for higher order Voronoi diagrams.

Appendix A: Proof of Lemma 2.3

Proof. For convenience, let $dgV_{i,j} = (dgV_i(\mathcal{X}) \cap dgV_j(\mathcal{X}))$ and let $\mathcal{I}_{dg}$ be the set of agents such that $dgV_{i,j} \cap dgV_k \neq \emptyset$. Now, if $dgV_{i,j} \neq \emptyset$, we know

$$\overline{dgV_{i,j}} = dgV_{i,j} \setminus \cup_{k \in \mathcal{I}_{dg}} gV_k(\mathcal{X}) \neq \emptyset$$

since $gV_i(\mathcal{X}) \subset dgV_i(\mathcal{X})\ \forall i \in \mathcal{I}_{dg}$. The point in $\overline{dgV_{i,j}}$ are not guaranteed to be closer to any agent in $\mathcal{I}_{dg}$ than the others, and they are guaranteed to be closer these agents than agents $k \in \mathcal{I} \setminus \mathcal{I}_{dg}$ since it is outside all the others' dual-guaranteed cells. Therefore, $\exists P \subset \mathcal{X}$ such that agent i and j share at least one point in $\overline{dgV_{i,j}}$ such that $V_i \cap V_j = \{q\} \mid q \in \overline{dgV_{i,j}}$. □

Appendix B: Proof of Corollary 2.4

Proof. Let us consider the worst case for $dgV_i(\mathcal{X}) \cap dgV_j(\mathcal{X}) \neq \emptyset$ to be true that is

$$dgV_i(\mathcal{X}) \cap dgV_j(\mathcal{X}) = \{q'\} \mid q' = \max_{q \in dgV_i(\mathcal{X})} (\|q - p_i\|).$$

Given the definition of the dual-guaranteed Voronoi cell, this implies that $\min \|q' - x_i\| = \max \|q' - x_j\|\ \ \forall x_i \in X_i^i,\ x_j \in X_j^i$. Since agent i knows its exact location, we can rewrite the previous equation as

$$\left\|q' - p_i\right\| = \max_{x_j \in X_j^i} \left\|q' - x_j\right\|.$$

By rearranging the previous equation, we get

$$\max_{x_j \in X_j^i} \|p_i - x_j\| \leq 2 \left\|q' - p_i\right\|.$$

Thus, any agent j s.t. $p_j \notin \overline{B}(p_i, 2\|q' - p_i\|)$ is guaranteed not to be a dual-guaranteed neighbor since the dual-guaranteed Voronoi cells will not intersect and is guaranteed not to be a Voronoi neighbor by Lemma 2.3. □

Appendix C: Proof of Proposition 2.5

Proof. To prove the claim we must show that both

$$\left\| C_{gV_i^i(\mathcal{N}_{dg_i})} - C_{V_i} \right\| \leq \mathrm{bnd}_i \tag{C.1}$$

and

$$\left\| C_{dgV_i^i(\mathcal{N}_{dg_i})} - C_{V_i} \right\| \leq \mathrm{bnd}_i \tag{C.2}$$

hold. By [2, Proposition 5.2], we know that for any sets $L \subset V \subset U$,

$$\|C_V - C_L\| \leq 2cr_U \left(1 - \frac{M_L}{M_U}\right).$$

Since $gV_i^i(\mathcal{N}_{dg_i}) \subset V_i \subset dgV_i^i(\mathcal{N}_{dg_i})$, the first condition (C.1) follows immediately with $L = gV_i^i, V = V_i$, and $U = dgV_i^i$. To show (C.2), let $L = V_i$, and $V = U = dgV_i^i$, then

$$\begin{aligned}\left\| C_{dgV_i^i(\mathcal{N}_{dg_i})} - C_{V_i} \right\| &\leq 2cr_{dgV_i^i} \left(1 - \frac{M_{V_i}}{M_{dgV_i^i}}\right) \\ &\leq 2cr_{dgV_i^i} \left(1 - \frac{M_{gV_i^i}}{M_{dgV_i^i}}\right) = \mathrm{bnd}_i,\end{aligned}$$

which concludes the proof. □

Appendix D: Proof of Lemma 2.6

Proof. We start by proving $j \in \mathcal{P}_i \Leftrightarrow i \in \mathcal{P}_j$ at all times. For $j \in \mathcal{P}_i \Leftrightarrow i \in \mathcal{P}_j$ to holds, agent i and j to must have the same last broadcasted information from their common potential neighbors and their-self. The reason begin that any agent that is not a common neighbor cannot be closer to any point in $dgV_i(\mathcal{X}) \cap dgV_j(\mathcal{X})$ than agent i, j and their common neighbors. Under the broadcasting range assignment, each common potential neighbor's information is guaranteed to reach agent i and j. In addition, by (2.15), if $j \in \mathcal{P}_i$, agent i uses to the same information agent j has about it-self X_i^i and agent j will do the same as well. In case $j \notin \mathcal{P}_i$, agent i uses its prefect information since it will broadcast to agent j by the decision control law in Section 2.4.3 if they becomes new neighbor, and when agent j receives agent i's information, agent j will have agent i's prefect information. Therefore, agent i and j will always have the same information required to determine if they are or they are not potential neighbor that guarantees $j \in \mathcal{P}_i \Leftrightarrow i \in \mathcal{P}_j$ at all times.

Now, we want to show $\mathcal{N}_i \subset \mathcal{P}_i$ is guaranteed at event-times. By Corollary 1, If agent i broadcasts a distance $R_\ell^i(\mathcal{P}_i)$ away, then all its potential neighbors will receive this broadcast. Since $j \in \mathcal{P}_i \Leftrightarrow i \in \mathcal{P}_j$ and by the decision control law in Section 2.4.3, any agent $k \notin \mathcal{P}_i$ will broadcast its information as soon as it gets agent i information if they become potential neighbors. Thus, when agent i broadcast, it will receive the new potential neighbors' information immediately. In case the ith agent did not receive any information when it broadcast, it implies that the agent does not have any new neighbor. Therefore, $\mathcal{N}_i \subset \mathcal{P}_i$ is guaranteed at event-times. $\square$

Appendix E: Proof of Proposition 2.7

Proof. In this proof, we want to guarantee that

$$gV_i^i(\mathcal{P}_i) \subset V_i \subset dgV_i^i(\mathcal{P}_i) \tag{E.1}$$

at all times under the the `event-triggered broadcasting algorithm`. Let us start by saying that as the potential neighbors' uncertainties increase, the guaranteed Voronoi cell shrinks and the dual-guaranteed Voronoi cell expands, and when the agent computes the uncertainties using (2.16), the guaranteed and dual-guaranteed Voronoi cells change faster than when the uncertainties computed by (2.8). By [2, Lemma 4.1 and 4.2], the ith agent' cells given the agent's and $\pi_{\mathcal{P}_i}(\mathcal{D}_i)$ information satisfies $gV_i^i(\overline{\mathcal{X}}) \subset gV_i^i(\mathcal{X}) \subset V_i \subset dgV_i^i(\mathcal{X}) \subset dgV_i^i(\overline{\mathcal{X}})$, where $\overline{\mathcal{X}} = \{X_j^i\}_{j\in\mathcal{P}_i}$ computed using (2.16) and $\mathcal{X} = \{X_j^i\}_{j\in\mathcal{P}_i}$ computed using (2.8). Let us start by proving (E.1) is guaranteed at every event-time. By Lemma 2.6, at every event-time, $\mathcal{N}_i \subset \mathcal{P}_i$ is guaranteed, and as a result (E.1) is guaranteed as well since the ith agent has all Voronoi neighbors' information.

Now, we will prove that (E.1) is guaranteed between event-times. Ideally, the ith agent can move and compute the uncertainties using (2.8) until condition (2.12) is invalid. However, this requires $\mathcal{N}_i \subset \mathcal{P}_i$ to be true at all times. This is challenging to ensure because agent i will not know about a new potential neighbor until it broadcast. Instead, we let the agents compute the uncertainties using (2.16) when they are in motion. [2, Lemma 4.1] state that if $\mathcal{N}_i \subset \mathcal{P}_i$ it satisfied and the agent expands the uncertainties using (2.16), $gV_i^i(\mathcal{P}_i) \subset V_i$ is guaranteed without using any additional information. In addition, [2, Lemma 4.2 and 4.3] state that by expanding $\{X_j^i\}_{j\in\mathcal{P}_i}$ using (2.16), the dual-guaranteed Voronoi cell cannot be bigger given any agent $k \in \mathcal{I}$ perfect information. In fact [2, Lemma 4.2 and 4.3] guarantee $V_i \subset dgV_i^i(\mathcal{P}_i)$ at all times even if $\mathcal{N}_i \not\subset \mathcal{P}_i$. Since $\mathcal{N}_i \subset \mathcal{P}_i$ is guaranteed at event-time, by Lemma 2.6, under the `event-triggered broadcasting algorithm`, (E.1) is guaranteed while the agents are in motion.

Furthermore, when the ith agent is waiting for new/updated information, the agent will not affect any agent $k \in \mathcal{I}$ because it is not moving. Also, since (E.1) is guaranteed while the other agents are moving, the moving agents will not affect agent i. Therefore, agent i cannot affect or be affected by any other agent. Thus, the agent expands the uncertainties as necessary using (2.8), and the `event-triggered broadcasting algorithm` guaranteed (E.1) while the agents are waiting.

Since (E.1) is guaranteed at event-times and between event-times, (E.1) is guaranteed at all-times. $\square$

Appendix F: Proof of Lemma 3.10

Proof. Given the definition of possible Voronoi cell, any point inside the set

$$\left((pV_i(\mathcal{X}, W) \cap pV_j(\mathcal{X}, W)) \setminus \bigcup_{k \in \mathcal{I} \setminus \{i,j\}} pV_k(\mathcal{X}, W) \right)$$

is guaranteed to be closer to agent i and j only since they are outside all the other possible Voronoi cells. In addition, if the set connects the the guaranteed Voronoi cells of agent i and j, agents i and j must share at least one point in Q that ensure them to be Voronoi neighbors. □

Appendix G: Proof of Lemma 3.12

Proof. Given the definition of possible Voronoi cell, any point inside $pV_i(\mathcal{X}, W)$ can be a point in $V_i(P, W)$ for at least one configuration of $p_j \in X_j \ \forall \ j \in \mathcal{I}$. With that being said, if $pV_i(\mathcal{X}, W) \cap pV_j(\mathcal{X}, W) \neq \emptyset$, there is a $q \in Q$ such that $q \in V_i(P, W)$ and $q \in V_j(P, W)$ for at least one configuration of $p_j \in X_j \ \forall \ j \in \mathcal{I}$. Thus, the agents i and j can be Voronoi neighbors for at least one configuration of their Voronoi cell will intersect. □

Appendix H: Proof of Corollary 3.13

Proof. Let us consider the worst case for $pV_i(\mathcal{X}, W) \cap pV_j(\mathcal{X}, W) \neq \emptyset$ to be true that is

$$pV_i(\mathcal{X}, W) \cap pV_j(\mathcal{X}, W) = \{q'\} \ s.t.$$

$$q' = \max_{\substack{x_i \in X_i, \\ q \in pV_i(\mathcal{X}, W)}} \left(\|q - x_i\| \right).$$

Given the definition of the possible Voronoi cell in (3.6), this implies that

$$\frac{1}{\alpha_i} \min_{x_i \in X_i} \left\| q' - x_i \right\|^{\beta_i} - \gamma_i = \frac{1}{\alpha_j} \max_{x_j \in X_j} \left\| q' - x_j \right\|^{\beta_j} - \gamma_j.$$

By rearranging the previous equation, we get

$$\max_{x_j \in X_j} \left\| q' - x_j \right\| = \sqrt[\beta_j]{\alpha_j \left(\frac{1}{\alpha_i} \min_{x_i \in X_i} \| q' - x_i \|^{\beta_i} - \gamma_i + \gamma_j \right)}.$$

Let

$$N_i = \max_{j \in p\mathcal{N}_i} \left(\max_{x_i \in X_i} \left\| q' - x_i \right\| + \sqrt[\beta_j]{\alpha_j \left(\frac{1}{\alpha_i} \min_{x_i \in X_i} \| q' - x_i \|^{\beta_i} - \gamma_i + \gamma_j \right)} \right),$$

where $q' = \max_{\substack{x_i \in X_i, \\ q \in pV_i(\mathcal{X}, W)}} (\| q - x_i \|)$, Thus, any agent k s.t. $p_k \notin \overline{B}(\hat{p_i}, N_i)$ is guaranteed not to be a possible neighbor since the possible Voronoi cells will not intersect. □

Appendix I: Closed Form Equation for the guaranteed and Possible MWV Bisector

The closed form expression for the bisectors are a long expression, and they can be simplified if the agents at $\hat{p_i}$ and $\hat{p_j}$ are on the x-axis. Assume that the agents are on the x-axis is very useful because it allows us to use the symmetric property which does not require a horizontal and vertical transverse axis equations. Moreover, this assumption can be interpreted as shifting the points and then rotating them, and when the bisector is determined, a rotation back then shifting will provide the exact bisector. Now, we will present the steps to define the bisectors $gV_{i,j}^m$ and $pV_{i,j}^m$.

Let us start with the shifting step. The agent associated with the higher weights gets shifted to $[r_j, 0]$ coordination, and the other point must be shifted by the same amount. Then, the agent associated with the lower weights gets rotated such as it leis on the right to the other point. Now, let

$$q_y^1 = \Bigg(\Big(- (\hat{p_j}_x + r_j - x_1)(r_j - \hat{p_j}_x + x_1)(\alpha_i^4 q_x^2 - 2\alpha_i^4 q_x x_1 + \alpha_i^4 x_1^2 + \alpha_i^2\alpha_j^2 \hat{p_j}_x^2$$

$$- 2\alpha_i^2\alpha_j^2\hat{p_j}_x x_1 - 2\alpha_i^2\alpha_j^2 q_x^2 + 2\alpha_i^2\alpha_j^2 q_x \hat{p}_{ix} + 2\alpha_i^2\alpha_j^2 q_x x_1 - \alpha_i^2\alpha_j^2 r_j^2$$

$$- \alpha_i^2\alpha_j^2\hat{p}_{ix}^2 + \alpha_j^4 q_x^2 - 2\alpha_j^4 q_x \hat{p}_{ix} + \alpha_j^4 \hat{p}_{ix}^2)\Big)^{1/2} - \alpha_i^2 \hat{p_j}_x^2 + \alpha_i^2 r_j^2$$

$$- \alpha_i^2 x_1^2 + 2\alpha_i^2 \hat{p_j}_x x_1\Bigg)\Bigg/\Bigg((\alpha_i^2 - \alpha_j^2)\Big(- \hat{p_j}_x^2 + 2\hat{p_j}_x x_1 + r_j^2 - x_1^2\Big)^{1/2}\Bigg),$$

and

$$q_y^2 = -\Bigg(\Big(- (\hat{p_j}_x + rj - x_2)(rj - \hat{p_j}_x + x_2)(\alpha_i^4 q_x^2 - 2\alpha_i^4 q_x x_2 + \alpha_i^4 x_2^2$$

$$- \alpha_i^2\alpha_j^2\hat{p}_{ix}^2 + 2\alpha_i^2\alpha_j^2\hat{p}_{ix} q_x + \alpha_i^2\alpha_j^2\hat{p_j}_x^2 - 2\alpha_i^2\alpha_j^2\hat{p_j}_x x_2 - 2\alpha_i^2\alpha_j^2 q_x^2$$

$$+ 2\alpha_i^2\alpha_j^2 q_x x_2 - \alpha_i^2\alpha_j^2 rj^2 + \alpha_j^4\hat{p}_{ix}^2 - 2\alpha_j^4\hat{p}_{ix} q_x + \alpha_j^4 q_x^2)\Big)^{1/2} + \alpha_i^2\hat{p_j}_x^2$$

$$- \alpha_i^2 rj^2 + \alpha_i^2 x_2^2 - 2\alpha_i^2\hat{p_j}_x x_2\Bigg)\Bigg/\Bigg((\alpha_i^2 - \alpha_j^2)\Big(- \hat{p_j}_x^2 + 2\hat{p_j}_x x_2 + rj^2 - x_2^2\Big)^{1/2}\Bigg),$$

where

$$\begin{aligned} x_1 = \Bigg(& \alpha_j^2 \hat{p}_{j_x}^3 - 2\alpha_j^2 \hat{p}_{j_x}^2 q_x + \alpha_i^2 q_x r_j^2 - \alpha_j^2 \hat{p}_{j_x} \hat{p}_{i_x}^2 + \alpha_j \hat{p}_{j_x} r_j \Big(- \alpha_i^2 \hat{p}_{j_x}^2 \\ & + 2q_x \alpha_i^2 \hat{p}_{j_x} + \alpha_i^2 r_j^2 + \alpha_i^2 \hat{p}_{i_x}^2 - 2q_x \alpha_i^2 \hat{p}_{i_x} + \alpha_j^2 \hat{p}_{j_x}^2 - 2q_x \alpha_j^2 \hat{p}_{j_x} - \alpha_j^2 \hat{p}_{i_x}^2 \\ & + 2q_x \alpha_j^2 \hat{p}_{i_x} \Big)^{1/2} - \alpha_j q_x r_j \Big(- \alpha_i^2 \hat{p}_{j_x}^2 + 2q_x \alpha_i^2 \hat{p}_{j_x} + \alpha_i^2 r_j^2 + \alpha_i^2 \hat{p}_{i_x}^2 - 2q_x \alpha_i^2 \hat{p}_{i_x} \\ & + \alpha_j^2 \hat{p}_{j_x}^2 - 2q_x \alpha_j^2 \hat{p}_{j_x} - \alpha_j^2 \hat{p}_{i_x}^2 + 2q_x \alpha_j^2 \hat{p}_{i_x} \big)^{1/2} + 2\alpha_j^2 \hat{p}_{j_x} q_x \hat{p}_{i_x} \Bigg) \Bigg/ \Bigg(\alpha_i^2 r_j^2 \\ & + \alpha_j^2 \hat{p}_{j_x}^2 - 2q_x \alpha_j^2 \hat{p}_{j_x} - \alpha_j^2 \hat{p}_{i_x}^2 + 2q_x \alpha_j^2 \hat{p}_{i_x} \Bigg), \end{aligned}$$

and

$$\begin{aligned} x_2 = \Bigg(& \alpha_j^2 \hat{p}_{j_x}^3 - \alpha_j^2 \hat{p}_{i_x}^2 \hat{p}_{j_x} - 2\alpha_j^2 \hat{p}_{j_x}^2 q_x + \alpha_i^2 q_x r_j^2 - \alpha_j \hat{p}_{j_x} r_j \Big(\alpha_i^2 \hat{p}_{i_x}^2 \\ & - 2q_x \alpha_i^2 \hat{p}_{i_x} - \alpha_i^2 \hat{p}_{j_x}^2 + 2q_x \alpha_i^2 \hat{p}_{j_x} + \alpha_i^2 r_j^2 - \alpha_j^2 \hat{p}_{i_x}^2 + 2q_x \alpha_j^2 \hat{p}_{i_x} + \alpha_j^2 \hat{p}_{j_x}^2 \\ & - 2q_x \alpha_j^2 \hat{p}_{j_x} \Big)^{1/2} + \alpha_j q_x r_j \Big(\alpha_i^2 \hat{p}_{i_x}^2 - 2q_x \alpha_i^2 \hat{p}_{i_x} - \alpha_i^2 \hat{p}_{j_x}^2 + 2q_x \alpha_i^2 \hat{p}_{j_x} + \alpha_i^2 r_j^2 \\ & - \alpha_j^2 \hat{p}_{i_x}^2 + 2q_x \alpha_j^2 \hat{p}_{i_x} + \alpha_j^2 \hat{p}_{j_x}^2 - 2q_x \alpha_j^2 \hat{p}_{j_x} \Big)^{1/2} + 2\alpha_j^2 \hat{p}_{i_x} \hat{p}_{j_x} q_x \Bigg) \Bigg/ \Bigg(\alpha_i^2 r_j^2 \\ & - \alpha_j^2 \hat{p}_{i_x}^2 + 2q_x \alpha_j^2 \hat{p}_{i_x} + \alpha_j^2 \hat{p}_{j_x}^2 - 2q_x \alpha_j^2 \hat{p}_{j_x} \Bigg) \end{aligned}$$

are evaluated on

$$\min(gV_1^m, gV_2^m) < q_x < \max(gV_1^m, gV_2^m),$$

where

$$gV_1^m = (\alpha_i(\hat{p_j}_x + r_j) - \alpha_j \hat{p}_{ix})/(\alpha_i - \alpha_j)$$

$$gV_2^m = (\alpha_i(\hat{p_j}_x - r_j) + \alpha_j \hat{p}_{ix})/(\alpha_i + \alpha_j)$$

and where

$$q_y^1(\min(gV_1^m, gV_2^m)) = q_y^2(\min(gV_1^m, gV_2^m)) = 0$$

$$q_y^1(\max(gV_1^m, gV_2^m)) = q_y^2(\max(gV_1^m, gV_2^m)) = 0$$

Note that $\hat{p}_{ix}$ and $\hat{p_j}_x$ are the x-coordinate of $\hat{p_i}$ and $\hat{p_j}$.

Now, we provide the closed form equations when $\hat{p_i}$ and $\hat{p_j}$ on the x-axis as

$$gV_{i,j}^m = \begin{cases} Q \setminus \{\pm q_y^1\} & \text{if } w_i > w_j \\ \{\pm q_y^2\} & \text{if } w_i < w_j \end{cases}$$

and

$$pV_{i,j}^m = \begin{cases} Q \setminus \{\pm q_y^2\} & \text{if } w_i > w_j \\ \{\pm q_y^1\} & \text{if } w_i < w_j \end{cases}$$

In case, $w_i = w_j$, the multiplicatively guaranteed and possible Voronoi cells are computed as ordinary guaranteed and possible Voronoi cells respectively.

References

[1] M. Rout and R. Roy, "Dynamic deployment of randomly deployed mobile sensor nodes in the presence of obstacles," *Ad Hoc Networks*, vol. 46, pp. 12 – 22, 2016.

[2] C. Nowzari and J. Cortés, "Self-triggered coordination of robotic networks for optimal deployment," *Automatica*, vol. 48, no. 6, pp. 1077 – 1087, 2012.

[3] F. Sharifi, Y. Zhang, and A. G. Aghdam, "A distributed deployment strategy for multi-agent systems subject to health degradation and communication delays," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 623–633, 2014.

[4] R. Zhang, D. Yuan, and Y. Wang, "A health monitoring system for wireless sensor networks," in *2007 2nd IEEE Conference on Industrial Electronics and Applications*, Shangri-La Hotel, Harbin, May 2007, pp. 1648–1652.

[5] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393 – 422, 2002.

[6] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[7] S.-H. Yang, *Wireless Sensor Networks: Principles, Design and Applications*. London, UK: Springer-Verlag, 2014.

[8] N. Noury, T. Herve, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, and T. Porcheron, "Monitoring behavior in home using a smart fall sensor and position sensors," in *1st Annual International IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology. Proceedings (Cat. No.00EX451)*, Palais des Congrès, Lyon, France, Oct 2000, pp. 607–610.

[9] J. K. Hart and K. Martinez, "Environmental sensor networks: A revolution in the earth system science?" *Earth-Science Reviews*, vol. 78, no. 3, pp. 177 – 191, 2006.

[10] N. A. A. Aziz, K. A. Aziz, and W. Z. W. Ismail, "Coverage strategies for wireless sensor networks," *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, vol. 3, no. 2, pp. 171 – 176, 2009.

[11] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams, Second Edition*. New York, NY, USA: John Wiley & Sons, May 2008, vol. 501.

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

[13] A. Boukerche and X. Fei, "A voronoi approach for coverage protocols in wireless sensor networks," in *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, Washington, DC, USA, Nov 2007, pp. 5190–5194.

[14] G. Hasegawa, S. Takemori, Y. Taniguchi, and H. Nakano, "Determining coverage area using voronoi diagram based on local information for wireless mesh networks," in *2012 Ninth International Conference on Information Technology - New Generations*, Las Vegas, Nevada, USA, April 2012, pp. 71–76.

[15] S. Megerian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Worst and best-case coverage in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 1, pp. 84–92, 2005.

[16] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *2008 47th IEEE Conference on Decision and Control*, Cancun, Mexico, Dec 2008, pp. 3947–3952.

[17] A. Breitenmoser, M. Schwager, J. C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *2010 IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, USA, May 2010, pp. 4982–4989.

[18] H. F. Parapari, F. Abdollahi, and M. B. Menhaj, "Coverage control in non-convex environment considering unknown non-convex obstacles," in *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, Oct 2014, pp. 119–124.

[19] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[20] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*, Fukuoka, Japan, June 2002, pp. 299–308.

[21] S. Li, C. Xu, W. Pan, and Y. Pan, "Sensor deployment optimization for detecting maneuvering targets," in *2005 7th International Conference on Information Fusion*, vol. 2, July 2005, pp. 1629–1635.

[22] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 1, pp. 61–91, 2004.

[23] J. Chen, S. Li, and Y. Sun, "Novel deployment schemes for mobile sensor networks," *Sensors*, vol. 7, no. 11, pp. 2907–2919, 2007.

[24] Y. Fuan, G. Feng, Y. Wang, and C. Song, "Distributed event-triggered control of multi-agent systems with combinatorial measurements," *Automatica*, vol. 49, no. 2, pp. 671–675, 2013.

[25] C. Nowzari and J. Cortés, "Distributed event-triggered coordination for average consensus on weight-balanced digraphs," *Automatica*, vol. 68, pp. 237 – 244, 2016.

[26] X. Liu, J. Sun, L. Dou, and J. Chen, "Leader-following consensus for discrete-time multi-agent systems with parameter uncertainties based on the event-triggered strategy," *Journal of Systems Science and Complexity*, vol. 30, no. 1, pp. 30–45, 2017.

[27] M. Zhao, C. Peng, W. He, and Y. Song, "Event-triggered communication for leader-following consensus of second-order multiagent systems," *IEEE Transactions on Cybernetics*, vol. 48, no. 6, pp. 1888–1897, 2018.

[28] X. Pan, Z. Liu, and Z. Chen, "Distributed optimization over weight-balanced digraphs with event-triggered communication," in *Proceedings of 2016 Chinese Intelligent Systems Conference*, Xiamen, China, September 2016, pp. 489–504.

[29] X. Shen, J. Chen, and Y. Sun, "Grid scan: A simple and effective approach for coverage issue in wireless sensor networks," in *2006 IEEE International Conference on Communications*, vol. 8, Istanbul, Turkey, June 2006, pp. 3480–3484.

[30] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 2, San Francisco, California, USA, March 2003, pp. 1293–1303.

[31] S. Firouzabadi and N. Martins, "Jointly optimal power allocation and constrained node placement in wireless networks of agents," *University of Maryland*, Technical Report. 2008.

[32] S. Captain. (2016) The U.S. military offers $2 million for wireless devices that can share the airewaves. [Online]. Available: https://www.fastcompany.com/3059991/the-us-military-offers-2-million-for-wireless-devices-that-can-share-the-airwaves

[33] C. Nowzari and J. Cortés, "Self-triggered and team-triggered control of networked cyber-physical systems," in *Event-Based Control and Signal Processing*, M. Miskowicz, Ed. CRC Press, 2015, pp. 203–2019.

[34] Z. Zheng, X. Zhang, and L. Jiao, "Optimized deployment of sensor networks based on event-triggered mechanism," in *2018 37th Chinese Control Conference (CCC)*, Wuhan, China, July 2018, pp. 7304–7309.

[35] N. Hayashi, Y. Muranishi, and S. Takai, "Distributed event-triggered control for voronoi coverage," in *2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, Krakow, Poland, June 2015, pp. 1–4.

[36] A. M.-C. So and Y. Ye, "On solving coverage problems in a wireless sensor network using voronoi diagrams," in *Internet and Network Economics*, Berlin, Heidelberg, December 2005, pp. 584–593.

[37] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *IEEE Conference on Decision and Control*, Maui, HI, USA, December 2012, pp. 3270–3285.

[38] C. Nowzari, J. Cortés, and G. J. Pappas, "Team-triggered coordination of robotic networks for optimal deployment," in *American Control Conference*, Chicago, IL, USA, July 2015, pp. 5744–5751.

[39] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson, "Event-based broadcasting for multi-agent average consensus," *Automatica*, vol. 49, no. 1, pp. 245 – 252, 2013.

[40] M. Ajina and C. Nowzari, "An event-triggered virtual force algorithm for multi-agent coverage control with obstacles," in *2018 Annual American Control Conference (ACC)*, Milwaukee, WI, USA, June 2018, pp. 1009–1014.

[41] M. Zhao, C. Peng, W. He, and Y. Song, "Event-triggered communication for leader-following consensus of second-order multiagent systems," *IEEE Transactions on Cybernetics*, vol. 48, no. 6, pp. 1888–1897, 2018.

[42] M. Zhong and C. G. Cassandras, "Asynchronous distributed optimization with minimal communication," in *2008 47th IEEE Conference on Decision and Control*, Cancun, Mexico, Dec 2008, pp. 363–368.

[43] C. Nowzari, "Multi-agent coordination via a shared wireless spectrum," in *IEEE Conference on Decision and Control*, Melbourne, Australia, December 2017, pp. 6714–6719.

[44] W. Evans and J. Sember, "Guaranteed voronoi diagrams of uncertain sites," in *Proceedings of the 20th Canadian Conference on Computational Geometry (CCCG2008)*, Montreal, Canada, August 2008, pp. 207–210.

[45] M. Jooyandeh, A. Mohades, and M. Mirzakhah, "Uncertain voronoi diagram," *Information Processing Letters*, vol. 109, no. 13, pp. 709 – 712, 2009.

[46] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: Applications and algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, 1999.

[47] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton, NJ, USA: Princeton University Press, 2009.

[48] J. Cortés, "Distributed kriged kalman filter for spatial estimation," *IEEE Transactions on Automatic Control*, vol. 54, no. 12, pp. 2816–2827, 2009.

[49] A. Dirafzoon, S. Emrani, S. M. A. Salehizadeh, and M. B. Menhaj, "Coverage control in unknown environments using neural networks," *Artificial Intelligence Review*, vol. 38, no. 3, pp. 237–255, 2012.

[50] C. Nowzari and J. Cortés, "Team-triggered coordination for real-time control of networked cyberphysical systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 1, pp. 34–47, 2016.

[51] G. L. Dirichlet, "Über die reduction der positiven quadratischen formen mit drei unbestimmten ganzen zahlen," *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 40, pp. 209–227, 1850.

[52] G. Voronoi, “Nouvelles applications des paramétres continus á la théorie des formes quadratiques,” *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 134, pp. 198–287, 1908.

[53] T. Xu and M. Li, “Topological and statistical properties of a constrained voronoi tessellation,” *Philosophical Magazine*, vol. 89, no. 4, pp. 349–374, 2009.

[54] D. Austin. (2006, August) Voronoi diagrams and a day at the beach. [Online]. Available: http://www.ams.org/publicoutreach/feature-column/fcarc-voronoi

[55] S. Drysdale. (1993, July) Voronoi diagrams: Applications from archaology to zoology. [Online]. Available: http://www.ics.uci.edu/ eppstein/gina/scot.drysdale.html

[56] M. Jooyandeh and A. M. Khorasani, “Fuzzy voronoi diagram,” in *Advances in Computer Science and Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 82–89.

[57] F. Sharifi, Y. Zhang, H. Mahboubi, and A. G. Aghdam, “Coverage control in multi-agent systems subject to communication delays,” in *Proceedings of 2012 IEEE/ASME 8th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, Suzhou , China, July 2012, pp. 269–274.

[58] F. Abbasi, A. Mesbahi, J. M. Velni, and C. Li, “Team-based coverage control of moving sensor networks with uncertain measurements,” in *2018 Annual American Control Conference (ACC)*, Milwaukee, WI, USA, June 2018, pp. 852–857.

[59] M. Ajina, D. Tabatabai, and C. Nowzari, “Asynchronous distributed event-triggered coordination for multi-agent coverage control,” *IEEE Transactions on Cybernetics*, vol. 0, no. 0, p. 0, 2019.

[60] T. Chevet, C. S. Maniu, C. Vlad, and Y. Zhang, “Guaranteed voronoi-based deployment for multi-agent systems under uncertain measurements,” in *2019 18th European Control Conference (ECC)*, Naples, Italy, June 2019, pp. 4016–4021.

[61] S. Papatheodorou, Y. Stergiopoulos, and A. Tzes, “Distributed area coverage control with imprecise robot localization,” in *2016 24th Mediterranean Conference on Control and Automation (MED)*, Athens, Greece, June 2016, pp. 214–219.

[62] S. Papatheodorou, A. Tzes, and K. Giannousakis, “Experimental studies on distributed control for area coverage using mobile robots,” in *2017 25th Mediterranean Conference on Control and Automation (MED)*, Msida, Malta, July 2017, pp. 690–695.

[63] H. Mahboubi, M. Vaezi, and F. Labeau, “Distributed deployment algorithms in a network of nonidentical mobile sensors subject to location estimation error,” in *SENSORS, 2014 IEEE*, Valencia, Spain, November 2014, pp. 1795–1798.

[64] H. Mahboubi and F. Labeau, “Deployment algorithms for coverage improvement in a network of mobile sensors with measurement error in the presence of obstacles,” in *2015 IEEE SENSORS*, Busan, South Korea, Nov 2015, pp. 1–4.

[65] H. Mahboubi, M. Vaezi, and F. Labeau, "Sensors deployment algorithms under limited communication range and measurement error," in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, Glasgow, Scotland, May 2015.

[66] S. Papatheodorou, A. Tzes, K. Giannousakis, and Y. Stergiopoulos, "Distributed area coverage control with imprecise robot localization: Simulation and experimental studies," *International Journal of Advanced Robotic Systems*, vol. 15, no. 5, pp. 1–15, 2018.

[67] M. Tzes, S. Papatheodorou, and A. Tzes, "Visual area coverage by heterogeneous aerial agents under imprecise localization," *IEEE Control Systems Letters*, vol. 2, no. 4, pp. 623–628, 2018.

[68] M. Tzes, S. Papatheodorou, and A. Tzes, "Collaborative visual area coverage by aerial agents under positioning uncertainty," in *2018 26th Mediterranean Conference on Control and Automation (MED)*, Zadar, Croatia, June 2018, pp. 149–154.

[69] H. Mahboubi, M. Vaezi, and F. Labeau, "Mobile sensors deployment subject to location estimation error," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 668–678, 2017.

[70] M. Turanli and H. Temeltas, "Adaptive coverage control with guaranteed power voronoi diagrams," in *2017 4th International Conference on Systems and Informatics (ICSAI)*, Hangzhou, China, Nov 2017, pp. 7–13.

[71] ——, "Workspace allocation for team of robots with different actuation capabilities," in *2018 International Conference on Control and Robots (ICCR)*, Hong Kong, September 2018, pp. 11–18.

[72] ——, "Multi-robot collaborative coverage under localization uncertainty," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, Tianjin, China, August 2019, pp. 1999–2005.

[73] H. Mahboubi and F. Labeau, "Distributed deployment strategies for prioritized coverage of a field under measurement error and limited communication capabilities," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, Boston, MA, USA, September 2015, pp. 1–5.

[74] P. K. Agarwal, B. Aronov, S. Har-Peled, J. M. Phillips, K. Yi, and W. Zhang, "Nearest-neighbor searching under uncertainty II," *ACM Transactions on Algorithms (TALG)*, vol. 13, no. 1, pp. 3:1–3:25, 2016.

[75] K. A. Schmid, T. Emrich, A. Züfle, M. Renz, and R. Cheng, "Approximate UV computation based on space decomposition," in *Proceedings of the 14th International symposium on spatial and temporal databases (SSTD)*, Hong Kong, August 2015.

[76] K. A. Schmid, A. Züfle, T. Emrich, M. Renz, and R. Cheng, "Uncertain voronoi cell computation based on space decomposition," *GeoInformatica*, vol. 21, no. 4, pp. 797–827, 2017.

[77] P. Zhang, R. Cheng, N. Mamoulis, M. Renz, A. Züfle, Y. Tang, and T. Emrich, "Voronoi-based nearest neighbor search for multi-dimensional uncertain databases," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, Brisbane, Australia, April 2013, pp. 158–169.

[78] M. Z. Hossain, M. Hasan, and M. A. Amin, "Efficient construction of UV-diagram," in *Advances in Swarm and Computational Intelligence*. Beijing, China: Springer International Publishing, 2015, pp. 329–340.

[79] R. Cheng, X. Xie, M. L. Yiu, J. Chen, and L. Sun, "UV-diagram: A voronoi diagram for uncertain data," in *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, Long Beach, CA, USA, March 2010, pp. 796–807.

[80] X. Xie, R. Cheng, M. L. Yiu, L. Sun, and J. Chen, "UV-diagram: a voronoi diagram for uncertain spatial databases," *The VLDB Journal*, vol. 22, no. 3, pp. 319–344, 2013.

[81] X. Xie, P. Jin, M. L. Yiu, J. Du, M. Yuan, and C. S. Jensen, "Enabling scalable geographic service sharing with weighted imprecise voronoi cells," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 2, pp. 439–453, 2016.

[82] I. I. Vaisman, A. Tropsha, and W. Zheng, "Compositional preferences in quadruplets of nearest neighbor residues in protein structures: statistical geometry analysis," in *Proceedings. IEEE International Joint Symposia on Intelligence and Systems (Cat. No.98EX174)*, Rockville, MD, USA, May 1998, pp. 163–168.

[83] H. Rangwala and G. Karypis, "Introduction to protein structure prediction," in *Introduction to Protein Structure Prediction*, H. Rangwala and G. Karypis, Eds. John Wiley & Sons, 2010, pp. 1–13.

[84] I. Vaisman, "Statistical and computational geometry of biomolecular structure," in *Handbook of Computational Statistics: Concepts and Methods*, J. E. Gentle, W. K. Härdle, and Y. Mori, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1095–1112.

[85] H. Edelsbrunner and P. Koehl, "Applications to structural molecular biology," in *Handbook of discrete and computational geometry*, J. E. Goodman, J. OŔourke, and C. D. Tóth, Eds. Chapman and Hall/CRC, 2017, pp. 1709–1735.

[86] H. Zhu, J. Hobdell, and A. Windle, "Effects of cell irregularity on the elastic properties of 2d voronoi honeycombs," *Journal of the Mechanics and Physics of Solids*, vol. 49, no. 4, pp. 857 – 870, 2001.

[87] V. Icke and R. Weygaert, "The galaxy distribution as a voronoi foam," *Quarterly Journal of the Royal Astronomical Society*, vol. 32, pp. 85–112, 1991.

[88] R. Ogniewicz and M. Ilg, "Voronoi skeletons: theory and applications," in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Champaign, IL, USA, June 1992, pp. 63–69.

[89] A. H. Thiessen, "Precipitation averages for large areas," *Monthly Weather Review*, vol. 39, no. 7, pp. 1082–1089, 1911.

[90] K.-H. Kim, E.-H. Lee, and S.-Y. Hong, "Potential of voronoi diagram for the conserved remapping of precipitation," *Monthly Weather Review*, vol. 146, no. 7, pp. 2237–2246, 2018.

[91] E. Wigner and F. Seitz, "On the constitution of metallic sodium," *Phys. Rev.*, vol. 43, no. 10, pp. 804–810, 1933.

[92] L. Hoofd, Z. Turek, K. Kubat, B. E. M. Ringnalda, and S. Kazda, "Variability of intercapillary distance estimated on histological sections of rat heart," in *Oxygen Transport to Tissue VII*, F. Kreuzer, S. M. C. Turek, and T. K. Goldstick, Eds. Springer US, 1985, pp. 239–247.

[93] A. A. Al-Shammari, E. A. Gaffney, and S. Egginton, "Modelling capillary oxygen supply capacity in mixed muscles: Capillary domains revisited," *Journal of Theoretical Biology*, vol. 356, pp. 47 – 61, 2014.

[94] C. Schröder, G. Neumayr, and O. Steinhauser, "On the collective network of ionic liquid/water mixtures. III. structural analysis of ionic liquids on the basis of voronoi decomposition," *The Journal of Chemical Physics*, vol. 130, no. 19, p. 194503, 2009.

[95] M. Satake, "Tensorial form definitions of discrete-mechanical quantities for granular assemblies," *International Journal of Solids and Structures*, vol. 41, no. 21, pp. 5775 – 5791, 2004.

[96] W. Whiteley, P. F. Ash, E. Bolker, and H. Crapo, "Convex polyhedra, dirichlet tessellations, and spider webs," in *Shaping Space: Exploring Polyhedra in Nature, Art, and the Geometrical Imagination*, M. Senechal, Ed. Springer New York, 2013, pp. 231–251.

[97] K. R. Guruprasad, "Effectiveness-based voronoi partition: a new tool for solving a class of location optimization problems," *Optimization Letters*, vol. 7, no. 8, pp. 1733–1743, 2013.

[98] K. Guruprasad and D. Ghose, "Heterogeneous locational optimisation using a generalised voronoi partition," *International Journal of Control*, vol. 86, no. 6, pp. 977–993, 2013.

[99] M. Tuceryan and A. K. Jain, "Texture segmentation using voronoi polygons," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, pp. 211–216, 1990.

[100] Y. Khazbak, J. Fan, S. Zhu, and G. Cao, "Preserving location privacy in ride-hailing service," in *2018 IEEE Conference on Communications and Network Security (CNS)*, Beijing, China, May 2018.

[101] M. Feeney. (2015, January) Is ridesharing safe? [Online]. Available: https://ssrn.com/abstract=2700891

Biography

Mohanad Ajina received his undergrad degree from Southern Illinois University Carbondale in 2014. In the same year, he joined George Mason University to do his Master in Electrical Engineering and he graduated in 2016. Also in the same year, he received his Architecture-Based Systems Integration graduate certificate. In addition, in 2016, he joined the PhD program in Electrical and Computer Engineering. During his PhD, he also worked as research assistant in the C4I center at George Mason University. His current research interests include distributed coordination algorithms, robotics, event- and self-triggered control, advanced data analytic, statistical modeling and machine learning.