UNIVERSITY of CALIFORNIA
Santa Barbara

Distributed Coordination for Teams of Robots

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Doctor of Philosophy

in

Mechanical Engineering

by

Joseph William Durham

Committee in charge:

Professor Francesco Bullo, Chair
Professor Jeff Moehlis
Professor Brad Paden
Professor Katie Byl

June 2011

The dissertation of Joseph William Durham is approved:

Professor Katie Byl

Professor Brad Paden

Professor Jeff Moehlis

Professor Francesco Bullo, Chair

June 2011

Distributed Coordination for Teams of Robots

Joseph William Durham

To my wife whose love and support helped so much through the long graduate school process, and whose willingness to join me on a new path in Santa Barbara has led us to so many interesting places.

To my parents who always encouraged me to pursue education and supported all my childhood activities, I understand more each day the sacrifices they made and how their choices helped make me who I am.

To my brother who has faced incredible challenges in life and every time shown an inner strength to not only perservere but thrive.

Acknowledgements

My biggest thanks go to my advisor, Francesco Bullo, whose guiding hand is present behind everything which appears in this thesis. He agreed to take me on and support my acquisition of research hardware which was new for the lab. Looking back, I can appreciate how I was given a long enough leash to choose my own problems and direction without getting lost. The supportive atmosphere in the lab and the interesting visitors I had the chance to collaborate with are all the direct result of Francesco. Despite his misgivings, he also allowed me the flexibility to travel and then finish up my degree partly from Boston when life pulled me in that direction for which I am very grateful.

I also owe a big thanks to Jeff Moehlis who guided me through my Masters work and beginning as a researcher, and who has been a friendly and supportive ear in the years since. My thanks also goes out to my other committee members, Katie Byl and Brad Paden. Their questions and feedback helped shape this work and my future thinking on these kinds of problems.

I have been fortunate to work directly with several other great scientists and mathematicians. Thank you to Ruggero Carli and Paolo Frasca for the rewarding discussions on partitioning and coverage. Thank you Antonio Franchi for all your work on frontier-based clearing and the incredible effort behind MIP in which we implemented our ideas. I would also like to thank a few people I worked with less directly but whose coding was helpful on various projects, including Dario Cazzaro, Luca Invernizzi, and Paolo Stegango.

The Bullo Lab has been a richly supportive atmosphere, and I count many past and present members of the lab among my collaborators and friends. I would like to thank all of my lab mates for the stimulating discussions in the lab and fun diversions outside of it.

I would also like to thank several influential people who helped me see the beauty and power of mathematics, computer sciences, and robotics. I would like to thank Gary Tsuruda who really inspired me in middle school and helped to light my learning fire which has been burning bright ever since. I would like to thank Peter Lindener who gave me my first taste of scientific research and engaged in so many interesting and challenging discussions on many topics over the years. Finally, I would like to thank Sebastien Thrun and the Stanford Racing Team for allowing me to participate in the early development of Stanley and to the fellow members of the path planning team who helped solidify my decision to pursue robotics.

Abstract

Distributed Coordination for Teams of Robots

by

Joseph William Durham

It is anticipated that in the near future autonomous teams of mobile robots will revolutionize the transportation of passengers and goods, search and rescue operations, and other applications. Challenging questions arise in designing coordination algorithms to synthesize and act on spatially distributed information so that a robotic team reaches a desirable global level of performance. We explore distributed coordination algorithms in two application areas. An emphasis of our work is on models and algorithms which can handle limited communication, localization errors, and the other challenges of hardware implementation.

The first application is territory partitioning and coverage in non-convex environments. We present coverage algorithms for two communication models: short-range, unreliable "gossip" communication between pairs of agents, and sporadic contact between individual robots and a central base station. In both cases we handle arbitrary non-convex environments by representing them as connected graphs. Our coverage algorithms drive teams of robots towards territory partitions which minimize the expected distance to task requests appearing in the environment. We also detail how to compute these solutions efficiently.

Our second application is a visibility-based pursuit-evasion problem in which a team of mobile robots with limited sensing and communication capabilities must coordinate to detect any evaders in an unknown, multiply-connected planar environment. Our distributed algorithm to guarantee evader detection is built around maintaining complete coverage of the frontier between cleared and contaminated regions while expanding the cleared region. We detail a novel distributed method for storing and updating this frontier without building a global map or requiring global localization.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Coordinated networks of mobile robots are already in use for environmental monitoring [2] and pickup and delivery in warehouses [3]. In the near future, autonomous robotic teams will revolutionize transportation of passengers and goods, search and rescue operations, and other applications. The appeal of novel robotic solutions in these areas varies by application, but broadly falls into the categories of augmenting the abilities of human operators, improving the levels of service provided, and increasing cost efficiency. For example, a group of underwater gliders used for ocean monitoring can provide a researcher with several streams of data about ocean currents and chemistry which would be very expensive and time consuming to gather by hand. The gliders can be out at sea for months at a time, providing a more thorough collection of data than one researcher could hope to accomplish otherwise, while also being ready to react to changing conditions.

There are several challenges in the coordination of a team of mobile robots which are common across the potential applications. An inherent aspect is that the robots will be distributed around an environment gathering information and deciding on actions to take. Interesting and challenging questions arise in determining how to best synthesize and act on this spatially distributed information so that the team as a whole reaches a desirable global state, especially when combined with practical limitations such as limited-range or intermittent communication. Even when all-to-all communication is available, local or hierarchical decision making is often preferable because it reduces both the load on the communication channel and the load on any central processors. This thesis will explore the challenges of multirobot coordination while presenting distributed solutions in two particular areas: territory partitioning and coverage control; and pursuit-evasion and search.

Partitioning and Coverage Control Many of the tasks envisioned for teams of mobile robots share a common feature: the robots are asked to provide service over a space. One question which naturally arises is: when a group of robots is waiting for a task request to come in, how can they best position themselves to be ready to respond? The distributed *environment partitioning problem* for robotic networks consists of designing individual control and communication laws such that the team divides a large space into regions. Typically, partitioning is done so as to optimize a cost function which measures the quality of service provided over all of the regions. *Coverage control* additionally optimizes the positioning of robots inside a region as shown in Fig. 1.1.

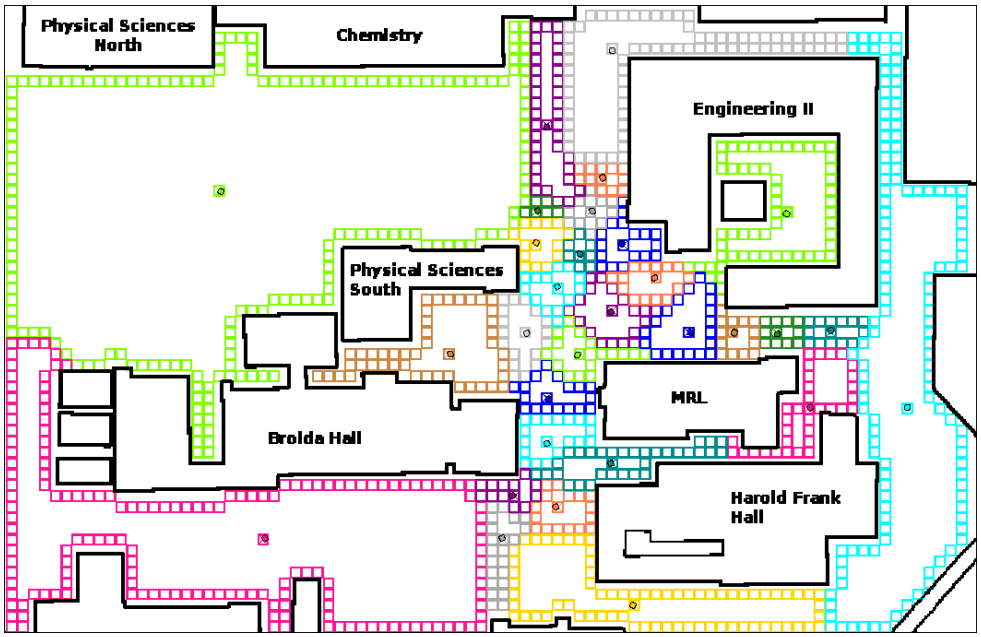


Figure 1.1: Example of a team of robots providing efficient coverage of a non-convex environment, as measured by an appropriate multicenter cost function.

In this thesis we will describe two distributed partitioning and coverage control algorithms for networks of robots with different available forms of communication. The first model is asynchronous, unreliable, range-limited communication between pairs of robots or "gossip" communication, whereas in the second model each robot can only talk occasionally to a central base station. In both cases the goal is to minimize the expected distance between the closest robot and spatially distributed events which will appear at discrete points in a non-convex environment. Optimality is defined with reference to relevant "multicenter" cost functions.

Pursuit-evasion and Search We examine a particular type of distributed *pursuit-evasion problem* known as the *clearing problem* which involves designing control and communication protocols such that a team of robotic searchers sweep an environment and detect any intruders which may be present. The clearing problem has received a lot of attention in recent years because of its applications to safety and security, and also for its relevance in search and rescue scenarios. In the literature, both probabilistic and guaranteed approaches are studied, we focus on guaranteeing that all "contaminated" areas which may contain evaders are eventually cleared. The particular challenges in achieving this guarantee are handling environments which are not known in advance, and coordinating clearing when global localization and global communication are unavailable.

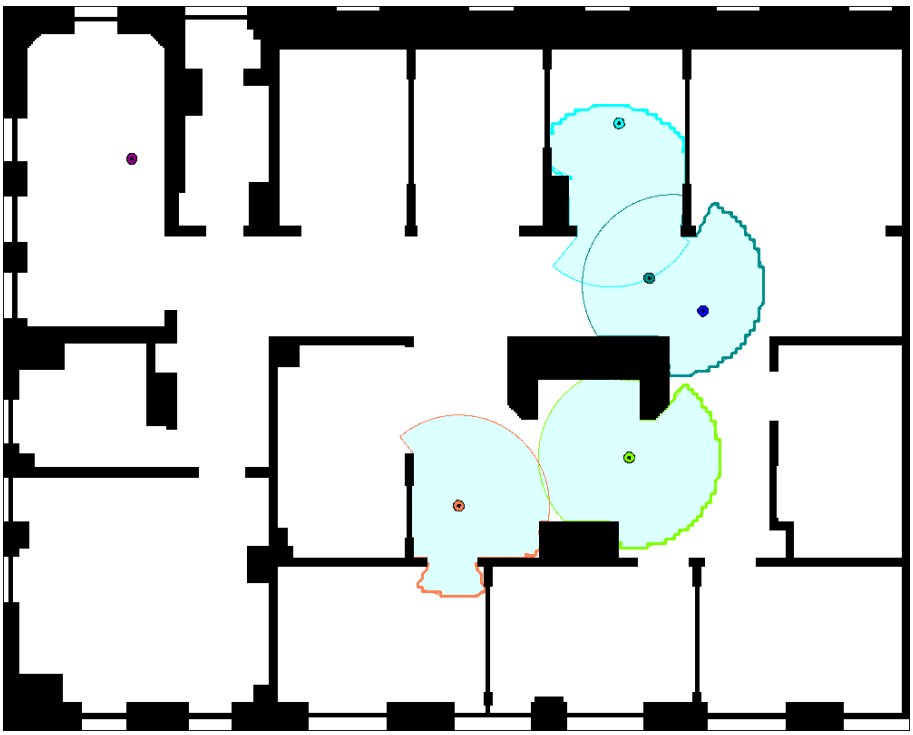


Figure 1.2: Example of our frontier-based approach for a team of robots to sweep through and clear an environment.

Our approach to environment clearing centers around the frontier or boundary between cleared and contaminated areas as shown in Fig. 1.2. Some robots will take on the role of local frontier-guards in charge of securing, updating, and expanding their local piece of the global frontier in coordination with their immediate neighbors. By distributing the handling of the frontier and making local decisions

of how to expand it, our algorithm removes the need for global localization or the construction of maps.

1.1 Literature Synopsis

In this section we give an overview of the existing literature for the problems considered in this thesis.

1.1.1 Partitioning and Coverage Control

A broad discussion of partitioning and coverage control is presented in [4] which builds on the classic work of Lloyd [5] on algorithms for optimal quantizer design through "centering and partitioning." The Lloyd-type approach studied in this paper was first adapted for distributed coverage control in [6]. Since this beginning, similar algorithms have been applied to non-convex environments [7, 8], unknown density functions [9, 10], equitable partitioning [11], and construction of truss-like objects [12]. There are also multi-agent partitioning algorithms built on market principles or auctions, see [13] for a survey. We will focus on distributed Lloyd methods because there are known ways to characterize equilibrium sets and prove convergence for these algorithms.

In [14] the authors have shown how a group of robotic agents can optimize the partition of convex bounded set using a Lloyd algorithm with gossip communication. The relationship between continuous and discrete coverage control laws using Euclidean distances in convex polygons is studied in [15]. A gossip Lloyd algorithm for optimizing partitions of a discrete set in non-convex environments using graph distances was presented in [16].

Territory partitioning and coverage control have applications in many fields. In cyber-physical systems, applications include automated environmental monitoring [2], fetching and delivery [3], construction [12], and other vehicle routing scenarios [17]. Coverage of discrete sets is closely related to the literature on data clustering and k-means [18, 19], as well as the facility location or k-center problem [20]. Partitioning of graphs is also its own field of research, see [21] for a survey. Territory partitioning through local interactions is also studied for animal groups, see for example [22, 23].

1.1.2 Pursuit-evasion and Search

In the literature on pursuit-evasion problems, many different approaches and starting assumptions have been explored. The study of guaranteeing detection

of evaders in planar environments began with [24]. For a single searcher, [25] studied a searcher with a limited field of view in a known polygon, while [26] cleared unknown environments without localization using minimalist sensing. The most similar work to this one is [27], which uses coordinated sweep lines of agents to clear unknown environments while building a graph representing the cleared space.

Pursuit-evasion on graphs representing decompositions of known environments is a related topic which goes back to [28] and includes recent works by [29] and [30]. Another active area is efficient evader detection, where one or more searchers are tasked with probabilistically locating targets which move randomly [31]. The pursuit-evasion literature has also addressed what to do once evaders are located, including tracking moving evaders [32] and capturing evaders [33].

Beyond pursuit-evasion, our work draws inspiration from methods for exploration and deployment of agents based on the frontier between explored and unknown regions [34, 35, 36].

1.2 Contributions of the Thesis

The contributions of this thesis are organized into three main chapters, followed by a shared conclusion.

Chapter 2 This chapter discusses partitioning and coverage control for robot teams with short-range "gossip" communication, i.e., asynchronous and unreliable communication between nearby robots. It also describes our approach to modeling non-convex environments as a set of discrete points from which we build a graph, which is also used as a foundation for Chapter 3. We present both a motion protocol which drives robots to meet their neighbors and a pairwise partitioning rule to update territory ownership when two robots meet. Our approach takes advantage of the discrete setting and the pairwise nature of updates to perform iterative optimal two-partitioning instead of using the more traditional Lloyd algorithm. We show that this change in philosophy results in significant improvements in the quality of the final solution. This chapter also includes results from a hardware-in-the-loop experiment to validate our proposal.

Chapter 3 This chapter also describes a partitioning and coverage control algorithm, but in this case for a one-to-base station communication model. The one-to-base model is intended for robots in environments where peer-to-peer

communication may be impractical, such as in the ocean, mountainous terrain, or urban environments. In this case it is the time delays between when different robots communicate with the central base station that introduces a distributed element to the problem. Our proposed algorithm evolves overlapping coverings of a discrete set of points in the environment and converges at equilibrium to a centroidal Voronoi partition. We also explain how this approach can smoothly handle the dynamic arrival or departure of robots from the team.

Chapter 4 In this chapter we change topics and present a frontier-based distributed algorithm to sweep complex non-convex environments and detect any evaders which may be present. Our algorithm is built around a set of four behaviors which each robot switches between depending on its local conditions. Chief among these is the frontier-guard behavior in which the robot becomes a local coordinator handling updates to and expansion of its piece of the global frontier between cleared and contaminated areas. We detail the distributed algorithm roles and how they work together to enable the team to clear unknown environments without global localization and without building maps. This chapter also contains results from hardware experiments demonstrating the approach.

Chapter 5 This chapter contains some conclusions and future directions for the work presented in this thesis.

Notation Finally, a few explanations of our notation. We use $\mathbb{R}_{\geq 0}$ to denote the set of non-negative real numbers and $\mathbb{Z}_{\geq 0}$ the set of non-negative integers. Given a set A, $|A|$ denotes the number of elements in A. Given sets A, B, their difference is $A \setminus B = \{a \in A \,|\, a \notin B\}$. A set-valued map, denoted by $T : A \rightrightarrows B$, associates to an element of A a subset of B.

Chapter 2

Coverage with Gossip Communication

This chapter describes a distributed partitioning and coverage control algorithm for a network of robots to minimize the expected distance between the closest robot and spatially distributed events which will appear at discrete points in a non-convex environment. Optimality is defined with reference to a relevant "multicenter" cost function. As with all multirobot coordination applications, the challenge comes from reducing the communication requirements: the proposed algorithm requires only short-range "gossip" communication, i.e., asynchronous and unreliable communication between nearby robots.

There are three main contributions in this work. First, we present a discrete partitioning and coverage optimization algorithm for mobile robots with unreliable, asynchronous, and short-range communication. Our algorithm has two components: a *motion protocol* which drives the robots to meet their neighbors, and a *pairwise partitioning rule* to update territories when two robots meet. The partitioning rule optimizes coverage of a set of points connected by edges to form a graph. The flexibility of graphs allows the algorithm to operate in non-convex, non-polygonal environments with holes. Our graph partition optimization approach can also be applied to non-planar problems or more general data sets.

Second, we provide an analysis of both the convergence properties and computational requirements of the algorithm. By studying a dynamical system of partitions of the graph's vertices, we prove that almost surely the algorithm converges to a pairwise-optimal partition in finite time. The set of pairwise-optimal partitions is shown to be a proper subset of the well-studied set of centroidal Voronoi partitions. We further describe how our pairwise partitioning rule can be implemented to run in anytime and how the computational requirements of the

algorithm can scale up for large domains and large teams.

Third, we detail experimental results from our implementation of the algorithm in the Player/Stage robot control system. We present a simulation of 30 robots providing coverage of a portion of a college campus to demonstrate that our algorithm can handle large robot teams, and a hardware-in-the-loop experiment conducted in our lab which incorporates sensor noise and uncertainty in robot position. Through numerical analysis we also show how our new approach to partitioning represents a significant performance improvement over both common Lloyd-type methods and the recent results in [14].

This chapter is organized as follows. In Section 2.1 we review and adapt coverage and geometric concepts (e.g., centroids, Voronoi partitions) to a discrete environment like a graph. We formally describe our robot network model and the discrete partitioning problem in Section 2.2, and then state our coverage algorithm and its properties. Section 2.3 contains proofs of the main convergence results. In Section 2.4 we detail our implementation of the algorithm and present experiments and comparative analysis. A summary is given in Section 2.5.

The work in this chapter was done in collaboration with R. Carli and P. Frasca.

2.1 Preliminary Material

We are given a team of N robots tasked with providing coverage of a finite set of points in a non-convex and non-polygonal environment. In this Section we translate concepts used in coverage of continuous environments to graphs.

2.1.1 Non-convex Environment as a Graph

Let Q be a finite set of points in a continuous environment. These points represent locations of interest, and are assumed to be connected by weighted edges. Let $G(Q) = (Q, E, w)$ be an (undirected) weighted graph with edge set $E \subset Q \times Q$ and weight map $w : E \to \mathbb{R}_{>0}$; we let $w_e > 0$ be the weight of edge e. We assume that $G(Q)$ is connected and think of the edge weights as distances between locations.

Remark 2.1.1 (Discretization of an environment) *For the examples in this thesis we will use a coarse* occupancy grid map *as a representation of a continuous environment. In an occupancy grid [37], each grid cell is either free space or an obstacle (occupied). To form a weighted graph, each free cell becomes a vertex and free cells are connected with edges if they border each other in the grid. Edge weights are the distances between the centers of the cells, i.e., the grid resolution.*

There are many other methods to discretize a space, including triangularization and other approaches from computational geometry [38], which could also be used.

In any weighted graph $G(Q)$ there is a standard notion of distance between vertices defined as follows. A *path* in G is an ordered sequence of vertices such that any consecutive pair of vertices is an edge of G. The *weight of a path* is the sum of the weights of the edges in the path. Given vertices h and k in G, the *distance* between h and k, denoted $d_G(h,k)$, is the weight of the lowest weight path between them, or $+\infty$ if there is no path. If G is connected, then the distance between any two vertices in G is finite. By convention, $d_G(h,k) = 0$ if $h = k$. Note that $d_G(h,k) = d_G(k,h)$, for any $h, k \in Q$.

2.1.2 Partitions of Graphs

We will be partitioning Q into N connected subsets or regions which will each be covered by an individual robot. To do so we need to define distances on induced subgraphs of $G(Q)$. Given $I \subset Q$, the *subgraph induced by the restriction of G to I*, denoted by $G \cap I$, is the graph with vertex set equal to I and edge set containing all weighted edges of G where both vertices belong to I. In other words, we set $(Q, E, w) \cap I = (Q \cap I, E \cap (I \times I), w|_{I\times I})$. The induced subgraph is a weighted graph with a notion of distance between vertices: given $h, k \in I$, we write $d_I(h,k) := d_{G\cap I}(h,k)$. Note that $d_I(h,k) \geq d_G(h,k)$.

We define a *connected subset of Q* as a subset $A \subset Q$ such that $A \neq \emptyset$ and $G \cap A$ is connected. We can then partition Q into connected subsets as follows.

Definition 2.1.2 (Connected Partitions) *Given the graph $G(Q) = (Q, E, w)$, we define a* connected $N-$partition of Q *as a collection $P = \{P_i\}_{i=1}^N$ of N subsets of Q such that*

(i) $\bigcup_{i=1}^N P_i = Q$;

(ii) $P_i \cap P_j = \emptyset$ *if* $i \neq j$;

(iii) $P_i \neq \emptyset$ *for all* $i \in \{1, \ldots, N\}$; *and*

(iv) P_i *is connected for all* $i \in \{1, \ldots, N\}$.

Let $\mathrm{Part}_N(Q)$ *to be the set of connected $N-$partitions of Q.*

Property (ii) implies that each element of Q belongs to just one P_i, i.e., each location in the environment is covered by just one robot. Notice that each $P_i \in P$ induces a connected subgraph in $G(Q)$. In subsequent references to P_i we will

often mean $G \cap P_i$, and in fact we refer to $P_i(t)$ as the *dominance subgraph* or *region* of the i-th robot at time t.

Among the ways of partitioning Q, there are some which are worth special attention. Given a vector of distinct points $c \in Q^N$, the partition $P \in \mathrm{Part}_N(Q)$ is said to be a *Voronoi partition of Q generated by c* if, for each P_i and all $k \in P_i$, we have $c_i \in P_i$ and $d_G(k, c_i) \leq d_G(k, c_j)$, $\forall j \neq i$. Note that the Voronoi partition generated by c is not unique since how to apportion tied vertices is unspecified.

2.1.3 Adjacency of Partitions

For our gossip algorithms we need to introduce the notion of adjacent subgraphs. Two distinct connected subgraphs P_i, P_j are said to be *adjacent* if there are two vertices q_i, q_j belonging, respectively, to P_i and P_j such that $(q_i, q_j) \in E$. Observe that if P_i and P_j are adjacent then $P_i \cup P_j$ is connected. Similarly, we say that robots i and j are adjacent or are neighbors if their subgraphs P_i and P_j are adjacent. Accordingly, we introduce the following useful notion.

Definition 2.1.3 (Adjacency Graph) *For $P \in \mathrm{Part}_N(Q)$, we define the* adjacency graph *between regions of partition P as $\mathcal{G}(P) = (\{1, \ldots, N\}, \mathcal{E}(P))$, where $(i, j) \in \mathcal{E}(P)$ if P_i and P_j are adjacent.*

Note that $\mathcal{G}(P)$ is always connected since $G(Q)$ is.

2.1.4 Cost Functions

We define three coverage cost functions for graphs: $\mathcal{H}_{\mathrm{one}}$, $\mathcal{H}_{\mathrm{multicenter}}$, and $\mathcal{H}_{\mathrm{expected}}$. Let the *weight function* $\phi : Q \to \mathbb{R}_{>0}$ assign a relative weight to each element of Q. The *one-center function* $\mathcal{H}_{\mathrm{one}}$ gives the cost for a robot to cover a connected subset $A \subset Q$ from a vertex $h \in A$ with relative prioritization set by ϕ:

$$\mathcal{H}_{\mathrm{one}}(h; A) = \sum_{k \in A} d_A(h, k)\phi(k).$$

A technical assumption is needed to solve the problem of minimizing $\mathcal{H}_{\mathrm{one}}(\cdot, A)$: we assume from now on that a *total order* relation, $<$, is defined on Q, i.e., that $Q = \{1, \ldots, |Q|\}$. With this assumption we can deterministically pick a vertex in A which minimizes $\mathcal{H}_{\mathrm{one}}$ as follows.

Definition 2.1.4 (Centroid) *Let Q be a totally ordered set, and let $A \subset Q$. We define the set of generalized centroids of A as the set of vertices in A which*

minimize $\mathcal{H}_{\text{one}}$*, i.e.,*

$$\mathrm{C}(A) := \operatorname*{argmin}_{h \in A} \mathcal{H}_{\text{one}}(h; A).$$

Furthermore, we define the map $\mathrm{Cd} : \mathrm{C}(Q) \to Q$ *such that* $\mathrm{Cd}(A) := \min\{c \in \mathrm{C}(A)\}$*. We call* $\mathrm{Cd}(A)$ *the* generalized centroid *of* A*.*

In subsequent use we drop the word "generalized" for brevity. Note that with this definition the centroid is well-defined, and also that the centroid of a region always belongs to the region. With a slight notational abuse, we define $\mathrm{Cd} : \mathrm{Part}_N(Q) \to Q^N$ as the map which associates to a partition the vector of the centroids of its elements.

We define the *multicenter function* $\mathcal{H}_{\text{multicenter}}$ to measure the cost for N robots to cover a connected N-partition P from the vertex set $c \in Q^N$:

$$\mathcal{H}_{\text{multicenter}}(c, P) = \frac{1}{\sum_{k \in Q} \phi(k)} \sum_{i=1}^{N} \mathcal{H}_{\text{one}}(c_i; P_i).$$

We aim to minimize the performance function $\mathcal{H}_{\text{multicenter}}$ with respect to both the vertices c and the partition P.

We can now state the coverage cost function we will be concerned with for the rest of this chapter. Let $\mathcal{H}_{\text{expected}} : \mathrm{Part}_N(Q) \to \mathbb{R}_{\geq 0}$ be defined by

$$\mathcal{H}_{\text{expected}}(P) = \mathcal{H}_{\text{multicenter}}(\mathrm{Cd}(P), P).$$

In the motivational scenario we are considering, each robot will periodically be asked to perform a task somewhere in its region with tasks appearing according to distribution ϕ. When idle, the robots would position themselves at the centroid of their region. By partitioning G so as to minimize $\mathcal{H}_{\text{expected}}$, the robot team would minimize the expected distance between a task and the robot which will service it.

2.1.5 Optimal Partitions

We introduce two notions of optimal partitions: centroidal Voronoi and pairwise-optimal. Our discussion starts with the following simple result about the multicenter cost function.

Proposition 2.1.5 (Properties of Multicenter Function) *Let* $P \in \mathrm{Part}_N(Q)$ *and* $c \in Q^N$*. If* P' *is a Voronoi partition generated by* c *and* $c' \in Q^n$ *is such that*

$c_i' \in \mathrm{C}(P_i)\ \forall\ i$, *then*

$$\mathcal{H}_{\text{multicenter}}(c, P') \leq \mathcal{H}_{\text{multicenter}}(c, P), \text{ and}$$
$$\mathcal{H}_{\text{multicenter}}(c', P) \leq \mathcal{H}_{\text{multicenter}}(c, P).$$

The second inequality is strict if any $c_i \notin \mathrm{C}(P_i)$.

Proposition 2.1.5 implies the following necessary condition: if (c, P) minimizes $\mathcal{H}_{\text{multicenter}}$, then $c_i \in \mathrm{C}(P_i)\ \forall i$ and P must be a Voronoi partition generated by c. Thus, $\mathcal{H}_{\text{expected}}$ has the following property as an immediate consequence of Proposition 2.1.5: given $P \in \mathrm{Part}_N(Q)$, if P^* is a Voronoi partition generated by $\mathrm{Cd}(P)$ then

$$\mathcal{H}_{\text{expected}}(P^*) \leq \mathcal{H}_{\text{expected}}(P).$$

This fact motivates the following definition.

Definition 2.1.6 (Centroidal Voronoi Partition) $P \in \mathrm{Part}_N(Q)$ *is a* centroidal Voronoi partition *of* Q *if there exists a* $c \in Q^n$ *such that* P *is a Voronoi partition generated by* c *and* $c_i \in \mathrm{C}(P_i)\ \forall\ i$.

The set of *pairwise-optimal partitions* provides an alternative definition for the optimality of a partition: a partition is pairwise-optimal if, for every pair of adjacent regions, one can not find a better two-partition of the union of the two regions. This condition is formally stated as follows.

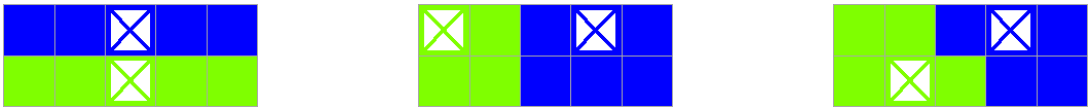
Definition 2.1.7 (Pairwise-optimal Partition) $P \in \mathrm{Part}_N(Q)$ *is a* pairwise-optimal partition *if for every* $(i, j) \in \mathcal{E}(P)$,

$$\mathcal{H}_{\text{one}}(\mathrm{Cd}(P_i); P_i) + \mathcal{H}_{\text{one}}(\mathrm{Cd}(P_j); P_j) = \min_{a,b \in P_i \cup P_j} \left\{ \sum_{k \in P_i \cup P_j} \min\left\{ d_{P_i \cup P_j}(a, k), d_{P_i \cup P_j}(b, k) \right\} \phi(k) \right\}.$$

The following Proposition states that the set pairwise-optimal partitions is in fact a subset of the set of centroidal Voronoi partitions. See Fig. 2.1 for an example which demonstrates that the inclusion is strict.

Proposition 2.1.8 (Pairwise-optimal Characterization) *Let* $P \in \mathrm{Part}_N(Q)$ *be a* pairwise-optimal partition. *Then* P *is also a* centroidal Voronoi partition.

Proof. To create a contradiction, assume that $P \in \mathrm{Part}_N(Q)$ is a pairwise-optimal partition but not a centroidal Voronoi partition. In other words, there



(a) (b) (c)

Figure 2.1: All possible centroidal Voronoi partitions of a 2×5 grid: (a) has a cost of 12 hops, (b) has a cost of 11, and (c) has a cost of 10. Only (c) is pairwise-optimal by definition.

exist components P_i and P_j in P and an element x of one component, say $x \in P_i$, such that

$$d_G\left(x, \mathrm{Cd}(P_i)\right) > d_G\left(x, \mathrm{Cd}(P_j)\right). \tag{2.1}$$

Furthermore, it is always possible to choose P_j such that

$$d_G\left(x, \mathrm{Cd}(P_j)\right) \leq d_G\left(x, \mathrm{Cd}(P_k)\right) \tag{2.2}$$

for all $k \neq j$.

Let $s^G_{a,b}$ be a shortest path in G connecting a to b. Consider a $s^G_{x,\mathrm{Cd}(P_j)}$ and let $m \in s^G_{x,\mathrm{Cd}(P_j)}$ be the first element of the path starting from $\mathrm{Cd}(P_j)$ which is not in P_j. Let ℓ be such that $m \in P_\ell$.

If $m = x$, then from (2.1) and the definition of $s^G_{x,\mathrm{Cd}(P_j)}$ we have that

$$d_{P_i}\left(x, \mathrm{Cd}(P_i)\right) > d_{P_i \cup P_j}\left(x, \mathrm{Cd}(P_j)\right)$$

which is a contradiction of the fact that P is pairwise-optimal.

If $m \neq x$, then, given (2.2), one of these two conditions holds:

(i) $d_G\left(m, \mathrm{Cd}(P_\ell)\right) > d_G\left(m, \mathrm{Cd}(P_j)\right)$, or

(ii) $d_G\left(m, \mathrm{Cd}(P_\ell)\right) = d_G\left(m, \mathrm{Cd}(P_j)\right)$.

In the first case, we again have a contradiction using the same logic above with m in place of x. In the second case, we must further consider whether there exists a $s^G_{m,\mathrm{Cd}(P_\ell)}$ such that every vertex in $s^G_{m,\mathrm{Cd}(P_\ell)}$ is also in P_ℓ. If there is not such a path, then

$$d_{P_\ell}\left(m, \mathrm{Cd}(P_\ell)\right) > d_G\left(m, \mathrm{Cd}(P_\ell)\right) = d_{P_\ell \cup P_j}\left(m, \mathrm{Cd}(P_j)\right)$$

and we again have a contradiction as above. If there is such a path, then we can instead repeat this analysis using using ℓ in place of j and considering the path

formed by this $s^G_{m,\mathrm{Cd}(P_\ell)}$ and the vertices in $s^G_{x,\mathrm{Cd}(P_j)}$ after m. Since the next vertex playing the role of m must be closer to x, we will eventually find a vertex which creates a contradiction. ∎

For a given environment Q, a pair made of a centroidal Voronoi partition P and the corresponding vector of centroids c is locally optimal in the following sense: $\mathcal{H}_{\mathrm{expected}}$ cannot be reduced by changing either P or c independently. A pairwise-optimal partition achieves this property and adds that for every pair of neighboring robots (i, j), there does not exist a two-partition of $P_i \cup P_j$ with a lower coverage cost. In other words, positioning the robots at the centroids of a centroidal Voronoi partition (locally) minimizes the expected distance between a task appearing randomly in Q according to relative weights described by ϕ and the robot who owns the vertex where the task appears. Positioning at the centroids of a pairwise-optimal partition improves performance by reducing the number of sub-optimal solutions which the team might converge to.

2.1.6 Review of Lloyd Optimization for Coverage & its Drawbacks

In the literature there are well-studied distributed algorithms based on the classic work of Lloyd [5] for optimizing an N-partition P so as to reach a centroidal Voronoi partition, see for instance [4] and [8]. The Lloyd approach is built around separate partitioning and centering steps, and while typically these algorithms are given for Euclidean spaces, the extension to discrete metric spaces is straightforward. These algorithms are distributed in the sense that each robot determines its dominance subgraph and its motion plan based only on communication with its neighbors in the adjacency graph. Specifically, at each discrete time instant $t \in \mathbb{Z}_{\geq 0}$, each robot i performs the following tasks: (1) i transmits its position and receives the positions of all adjacent robots; (2) i computes its Voronoi region P_i based on the information received and some tie breaking rule[1]; and (3) i moves to $\mathrm{Cd}(P_i)$. Along the lines of [4], one can show that such an algorithm causes P to converge to the set of centroidal Voronoi partitions.

While such iterative optimization algorithms are popular and work well in simulation, they require synchronous and reliable communication among the robots. As robots with adjacent regions may be arbitrarily far apart, these communication requirements are burdensome and unrealistic for deployed robotic networks. In

[1]There may be cells which are equidistant from i and the closest adjacent robot. A tie breaking rule is a method to decide which robot should own such a tied cell, such as giving tied cells to the robot with the lowest index.

addition, even for very simple graphs the set of centroidal Voronoi partitions may contain several sub-optimal configurations (see Fig. 2.1). We are thus interested in gossip coverage algorithms for two reasons: (1) they apply to more realistic robot network models, and (2) they will allow us to reach the set of pairwise-optimal partitions.

2.2 Models, Problem Formulation, and Proposed Solution

We aim to partition Q among N robotic agents using only asynchronous, unreliable, short-range communication. In Section 2.2.1 we describe the computation, motion, and communication capabilities required of the team of robots, and in Section 2.2.2 we formally state the problem we are addressing. In Section 2.2.3 we propose our solution, the *Discretized Gossip Coverage Algorithm*, and in 2.2.4 we provide an illustration. In Sections 2.2.5 and 2.2.6 we state the algorithm's convergence and complexity properties.

2.2.1 Robot Network Model with Gossip Communication

Our Discretized Gossip Coverage Algorithm requires a team of N robotic agents where each agent $i \in \{1, \ldots, N\}$ has the following basic computation and motion capabilities:

(C1) agent i knows its unique identifier i;

(C2) agent i has a processor with the ability to store $G(Q)$ and perform operations on subgraphs of G; and

(C3) agent i can determine which vertex in Q it occupies and can move at speed v along the edges of $G(Q)$ to any other vertex in Q.

Remark 2.2.1 *The localization requirement in (C3) is actually quite loose. Localization is only used for navigation and not for updating partitions, thus limited duration localization errors are not a problem.*

The robotic agents are assumed to be able to communicate with each other according to the *range-limited gossip communication model* which is described as follows:

(C4) given a communication range $r_{\text{comm}} > \max_{e \in E} w_e$, when any two agents reside for some positive duration at a distance $r < r_{\text{comm}}$, they communicate at the sample times of a Poisson process with intensity $\lambda_{\text{comm}} > 0$.

Recall that an homogeneous Poisson process is a widely-used stochastic model for events which occur randomly and independently in time, where the expected number of events in a period Δ is $\Delta\lambda_{\text{comm}}$.

Remark 2.2.2 (Communication Model) *(1) This communication capability is the minimum necessary for our algorithm, any additional capability can only reduce the time required for convergence. For example, it would be acceptable to have intensity $\lambda(r)$ depend upon the pairwise robot distance in such a way that $\lambda(r) \geq \lambda_{comm}$ for $r < r_{\text{comm}}$.*
(2) We use distances in the graph to model limited range communication. These graph distances are assumed to approximate geodesic distances in the underlying continuous environment and thus path distances for a diffracting wave or moving robot.

2.2.2 Problem Statement

Assume that, for all $t \in \mathbb{R}_{\geq 0}$, each agent $i \in \{1, \ldots, N\}$ maintains in memory a connected subset $P_i(t)$ of environment Q. Our goal is to design a distributed algorithm that iteratively updates the partition $P(t) = \{P_i(t)\}_{i=1}^N$ while solving the following optimization problem:

$$\min_{P \in \text{Part}_N(Q)} \mathcal{H}_{\text{expected}}(P), \tag{2.3}$$

subject to the constraints imposed by the robot network model with range-limited gossip communication from Section 2.2.1.

2.2.3 The Discretized Gossip Coverage Algorithm

In the design of an algorithm for the minimization problem (2.3) there are two main questions which must be addressed. First, given the limited communication capabilities in (C4), how should the robots move inside Q to guarantee frequent enough meetings between pairs of robots? Second, when two robots are communicating, what information should they exchange and how should they update their regions?

In this section we introduce the *Discretized Gossip Coverage Algorithm* which, following these two questions, consists of two components:

(1) the *Random Destination & Wait Motion Protocol*; and

(2) the *Pairwise Partitioning Rule*.

The concurrent implementation of the Random Destination & Wait Motion Protocol and the Pairwise Partitioning Rule determines the evolution of the positions and dominance subgraphs of the agents as we now formally describe. We start with the Random Destination & Wait Motion Protocol.

Algorithm 2.1: Random Destination & Wait Motion Protocol

Each agent $i \in \{1, \ldots, N\}$ determines its motion by repeatedly performing the following actions:

1: agent i samples a *destination vertex* q_i from a uniform distribution over its dominance subgraph P_i;
2: agent i moves to vertex q_i through the shortest path in P_i connecting the vertex it currently occupies and q_i; and
3: agent i waits at q_i for a duration $\tau > 0$.

If agent i is moving from one vertex to another we say that agent i is in the *moving* state while if agent i is waiting at some vertex we say that it is in the *waiting* state.

Remark 2.2.3 (Motion Protocol) *The motion protocol is designed to ensure frequent enough communication between pairs of robots. In general, any motion protocol can be used which meets this requirement, so i could select q_i from the boundary of P_i or use some heuristic non-uniform distribution over P_i.*

If any two agents i and j reside in two vertices at a graphical distance smaller that r_{comm} for some positive duration, then at the sample times of the corresponding communication Poisson process the two agents exchange sufficient information to update their respective dominance subgraphs P_i and P_j via the Pairwise Partitioning Rule.

Some remarks are now in order.

Remark 2.2.4 (Partitioning Rule) *(1) The Pairwise Partitioning Rule is designed to find a minimum cost two-partition of U. More formally, if list S and sets W_{a^*} and W_{b^*} for $(a^*, b^*) \in S$ are defined as in the Pairwise Partitioning Rule, then W_{a^*} and W_{b^*} are an optimal two-partition of U.*
(2) While the loop in steps 4-7 must run to completion to guarantee that W_{a^*} and W_{b^*} are an optimal two-partition of U, the loop is designed to return an

Algorithm 2.2: Pairwise Partitioning Rule

Assume that at time $t \in \mathbb{R}_{\geq 0}$, agent i and agent j communicate. Without loss of generality assume that $i < j$. Let $P_i(t)$ and $P_j(t)$ denote the current dominance subgraphs of i and j, respectively. Moreover, let t^+ denote the time instant just after t. Then agents i and j perform the following tasks:

1: agent i transmits $P_i(t)$ to agent j and vice-versa
2: initialize $W_{a^*} := P_i(t)$, $W_{b^*} := P_j(t)$, $a^* := \mathrm{Cd}(P_i(t))$, $b^* := \mathrm{Cd}(P_j(t))$
3: compute $U := P_i(t) \cup P_j(t)$ and a list S of all pairs of vertices in U
4: **for** each $(a, b) \in S$ **do**
5: compute the sets
$$W_a := \{x \in U : d_U(x, a) \leq d_U(x, b)\}$$
$$W_b := \{x \in U : d_U(x, b) < d_U(x, a)\}$$
6: **if** $\mathcal{H}_{\mathrm{one}}(a; W_a) + \mathcal{H}_{\mathrm{one}}(b; W_b) < \mathcal{H}_{\mathrm{one}}(a^*; W_{a^*}) + \mathcal{H}_{\mathrm{one}}(b^*; W_{b^*})$ **then**
7: $W_{a^*} := W_a, W_{b^*} := W_b, a^* := a, b^* := b$
8: $P_i(t^+) := W_{a^*}, \quad P_j(t^+) := W_{b^*}$

intermediate sub-optimal result if need be. If P_i and P_j change, then $\mathcal{H}_{\mathrm{expected}}$ will decrease and this is enough to ensure eventual convergence.
(3) We make a simplifying assumption in the Pairwise Partitioning Rule that, once two agents communicate, the application of the partitioning rule is instantaneous. We discuss the actual computation time required in Section 2.2.6 and some implementation details in Section 2.4.
(4) Notice that simply assigning W_{a^*} to i and W_{b^*} to j can cause the robots to "switch sides" in U. While convergence is guaranteed regardless, switching may be undesirable in some applications. In that case, any smart matching of W_{a^*} and W_{b^*} to i and j may be inserted.
(5) Agents who are not adjacent may communicate but the partitioning rule will not change their regions. Indeed, in this case W_{a^*} and W_{b^*} will not change from $P_i(t)$ and $P_j(t)$.

Some possible modifications and extensions to the algorithm are worth mentioning.

Remark 2.2.5 (Extensions) *(1) If the robots have heterogenous dynamics, line 5 can be modified to consider per-robot travel times between vertices. For example, $d_U(x, a)$ could be replaced by the expected time for robot i to travel from a to x while $d_U(x, b)$ would consider robot j.*
(2) Here we focus on partitioning territory, but this algorithm can easily be combined with methods to provide a service in Q as in [17]. The agents could split their

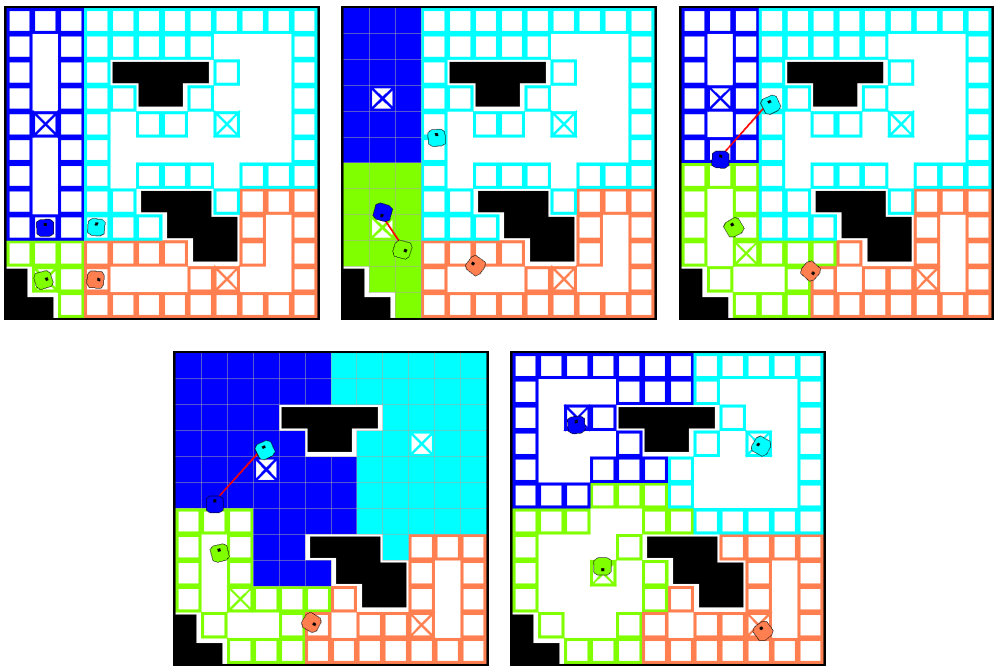


Figure 2.2: Simulation of four robots dividing a square environment with obstacles. The boundary of each robots territory is drawn in a different color, the centroid of a territory is drawn with an X, and pairwise communication is drawn with a solid red line. On the left is the initial partition assigned to the robots. The middle frames show two pairwise territory exchanges, with updated territories highlighted with solid colors. The final partition is shown at right.

time between moving to meet their neighbors and update territory, and performing requested tasks in their region.

2.2.4 Illustrative Simulation

The simulation in Fig. 2.2 shows four robots partitioning a square environment with obstacles where the free space is represented by a 12×12 grid. In the initial partition shown in the left panel, the robot in the top right controls most of the environment while the robot in the bottom left controls very little. The robots then move according to the Random Destination & Wait Motion Protocol, and communicate according to range-limited gossip communication model with $r_{comm} = 2.5m$ (four edges in the graph).

The first pairwise territory exchange is shown in the second panel, where the

bottom left robot claims some territory from the robot on the top left. A later exchange between the two robots on the top is shown in the next two panels. Notice that the cyan robot in the top right gives away the vertex it currently occupies. In such a scenario, we direct the robot to follow the shortest path in $G(Q)$ to its updated territory before continuing on to a random destination.

After 9 pairwise territory exchanges, the robots reach the pairwise-optimal partition shown at right in Fig. 2.2. The expected distance between a random vertex and the closest robot decreases from $2.34m$ down to $1.74m$.

2.2.5 Convergence Property

In this subsection we characterize the convergence of the Discretized Gossip Coverage Algorithm. The strength of our result is the possibility of enforcing that a partition will converge to a pairwise-optimal partition through pairwise territory exchange. In Theorem 2.2.6 we summarize this convergence property, with proofs given in Section 2.3.

Theorem 2.2.6 (Convergence Property) *Consider a network of N robotic agents endowed with computation and motion capacities (C1), (C2), (C3), and communication capacities (C4). Assume the agents implement the* Discretized Gossip Coverage Algorithm *consisting of the concurrent implementation of the* Random Destination & Wait Motion Protocol *and the* Pairwise Partitioning Rule. *Then,*

(i) the partition $P(t)$ remains connected and is described by $P : \mathbb{R}_{\geq 0} \to \mathrm{Part}_N(Q)$, and

(ii) $P(t)$ converges almost surely in finite time to a pairwise-optimal partition.

Remark 2.2.7 *For simplicity we assume uniform robot speeds, communication processes, and waiting times. An extension to non-uniform processes would be straightforward.*

2.2.6 Complexity Properties and Discussion

In this subsection we explore the computational requirements of the Discretized Gossip Coverage Algorithm, and make some comments on implementation. Cost function $\mathcal{H}_{\mathrm{one}}(h; P_i)$ is the sum of the distances between h and all other vertices in P_i. This computation of one-to-all distances is the core computation of the algorithm. For most graphs of interest the total number of edges $|E|$ is proportional

to $|Q|$, so we will state bounds on this computation in terms of $|P_i|$. Computing one-to-all distances requires one of the following:

- if all edge weights in $G(Q)$ are the same (e.g., for a graph from an occupancy grid), a breadth-first search approach can be used which requires $\mathcal{O}(|P_i|)$ in time and memory;
- otherwise, Dijkstra's algorithm must be used which requires $\mathcal{O}(|P_i| \log(|P_i|))$ in time and $\mathcal{O}(|P_i|)$ in memory.

Let $\mathbb{D}(P_i)$ be the time to compute one-to-all distances in P_i, then computing $\mathcal{H}_{\text{one}}(h; P_i)$ requires $\mathcal{O}(\mathbb{D}(P_i))$ in time.

Proposition 2.2.8 (Complexity Properties) *The motion protocol requires $\mathcal{O}(|P_i|)$ in memory, and $\mathcal{O}(\mathbb{D}(P_i))$ in computation time. The partitioning rule requires $\mathcal{O}(|P_i| + |P_j|)$ in communication bandwidth between robots i and j, $\mathcal{O}(|P_i| + |P_j|)$ in memory, and can run in any time.*

Proof. We first prove the claims for the motion protocol. Step 2 is the only non-trivial step and requires finding a shortest path in P_i, which is equivalent to computing one-to-all distances from the robot's current vertex. Hence, it requires $\mathcal{O}(\mathbb{D}(P_i))$ in time and $\mathcal{O}(P_i)$ in memory.

We now prove the claims for the partitioning rule. In step 1, robots i and j transmit their subgraphs to each other, which requires $\mathcal{O}(|P_i| + |P_j|)$ in communication bandwidth. For step 3, the robots determine $U := P_i \cup P_j$, which requires $\mathcal{O}(|P_i| + |P_j|)$ in memory to store. Step 4 is the start of a loop which executes $\mathcal{O}(|U|^2)$ times, affecting the time complexity of steps 5, 6 and 7. Step 5 requires two computations of one-to-all distances in U which each take $\mathcal{O}(\mathbb{D}(U))$. Step 6 involves four computations of $\mathcal{H}_{\text{one}}$ over different subsets of U, however those for W_{a^*} and W_{b^*} can be stored from previous computation. Since W_a and W_b are strict subsets of U, step 5 takes longer than step 6. Step 7 is trivial, as is step 8. The total time complexity of the loop is thus $\mathcal{O}(|U|^2\, \mathbb{D}(U))$.

However, the loop in steps 4-7 can be truncated after any number of iterations. While it must run to completion to guarantee that W_{a^*} and W_{b^*} are an optimal two-partition of U, the loop is designed to return an intermediate sub-optimal result if need be. If P_i and P_j change, then $\mathcal{H}_{\text{expected}}$ will decrease. Our convergence result will hold provided that all elements of S are eventually checked if P_i and P_j do not change. Thus, the partitioning rule can run in any time with each iteration requiring $\mathcal{O}(\mathbb{D}(U))$. ∎

All of the computation and communication requirements in Proposition 2.2.8 are independent of the number of robots and scale with the size of a robot's partition, meaning the Discretized Gossip Coverage Algorithm can easily scale up for large teams of robots in large environments.

2.3 Convergence Proofs

This section is devoted to proving the two statements in Theorem 2.2.6. The proof that the Pairwise Partitioning Rule maps a connected N-partition into a connected N-partition is straightforward. The proof of convergence is more involved and is based on the application of Lemma 2.3.1 in Section 2.3.2 below to the Discretized Gossip Coverage Algorithm.

2.3.1 Well-posedness

We start by proving that the Pairwise Partitioning Rule is well-posed in the sense that it maintains a connected partition.

Proof. [Proof of Theorem 2.2.6 statement (i)] To prove the statement we need to show that $P(t^+)$ satisfies points (i) through (iv) of Definition 2.1.2. From the definition of the Pairwise Partitioning Rule, we have that $P_i(t^+) \cup P_j(t^+) = P_i(t) \cup P_j(t)$ and $P_i(t^+) \cap P_j(t^+) = \emptyset$. Moreover, since $a^* \in P_i(t^+)$ and $b^* \in P_j(t^+)$, it follows that $P_i(t^+) \neq \emptyset$ and $P_j(t^+) \neq \emptyset$. These observations imply the validity of points (i), (ii), and (iii) for $P(t^+)$. Finally, we must show that $P_i(t^+)$ and $P_j(t^+)$ are connected, i.e., $P(t^+)$ also satisfies point (iv). To do so we show that, given $x \in W_{a^*}$, any shortest path in $P_i(t) \cup P_j(t)$ connecting x to a^* completely belongs to W_{a^*}. We proceed by contradiction. Let s_{x,a^*} denote a shortest path in $P_i(t) \cup P_j(t)$ connecting x to a^* and let us assume that there exists $m \in s_{x,a^*}$ such that $m \in W_{b^*}$. For m to be in W_{b^*} means that $d_{P_i(t) \cup P_j(t)}(m, b^*) < d_{P_i(t) \cup P_j(t)}(m, a^*)$. This implies that

$$\begin{aligned} d_{P_i \cup P_j}(x, b^*) &\leq d_{P_i \cup P_j}(m, b^*) + d_{P_i \cup P_j}(x, m) \\ &< d_{P_i \cup P_j}(m, a^*) + d_{P_i \cup P_j}(x, m) \\ &= d_{P_i \cup P_j}(x, a^*). \end{aligned}$$

This is a contradiction for $x \in W_{a^*}$. Similar considerations hold for W_{b^*}. ∎

2.3.2 Invariance Principle

For completeness we present a convergence result for set-valued algorithms on finite state spaces, which can be recovered as a direct consequence of [14, Theorem 4.5]. Lemma 2.3.1 establishes strong convergence properties for a particular class of set valued maps, which we will now briefly review.

Given a set X, a set-valued map $T : X \rightrightarrows X$ is a map which associates to an element $x \in X$ a subset $Z \subset X$. A set-valued map is non-empty if $T(x) \neq \emptyset$ for all $x \in X$. Given a non-empty set-valued map T, an evolution of the dynamical system associated to T is a sequence $\{x_n\}_{n\in\mathbb{Z}_{\geq 0}} \subset X$ where $x_{n+1} \in T(x_n)$ for all $n \in \mathbb{Z}_{\geq 0}$. A set $W \subset X$ is *strongly positively invariant* for T if $T(w) \subset W$ for all $w \in W$.

Lemma 2.3.1 (Persistent random switches imply convergence) *Let (X, d) be a finite metric space. Given a collection of maps $T_1, \ldots, T_m : X \to X$, define the set-valued map $T : X \rightrightarrows X$ by $T(x) = \{T_1(x), \ldots, T_m(x)\}$. Given a stochastic process $\sigma : \mathbb{Z}_{\geq 0} \to \{1, \ldots, m\}$, consider an evolution $\{x_n\}_{n\in\mathbb{Z}_{\geq 0}}$ of T satisfying $x_{n+1} = T_{\sigma(n)}(x_n)$. Assume that:*

(i) there exists a set $W \subseteq X$ that is strongly positively invariant for T;

(ii) there exists a function $U : W \to \mathbb{R}$ such that $U(w') < U(w)$, for all $w \in W$ and $w' \in T(w) \setminus \{w\}$; and

(iii) there exists $p \in (0, 1)$ and $k \in \mathbb{N}$ such that, for all $i \in \{1, \ldots, m\}$ and $n \in \mathbb{Z}_{\geq 0}$, there exists $h \in \{1, \ldots, k\}$ such that $\mathbb{P}\big[\sigma(n+h) = i \,|\, \sigma(n), \ldots, \sigma(1)\big] \geq p$.

For $i \in \{1, \ldots, m\}$, let F_i be the set of fixed points of T_i in W, i.e., $F_i = \{w \in W \mid T_i(w) = w\}$. If $x_0 \in W$, then the evolution $\{x_n\}_{n\in\mathbb{Z}_{\geq 0}}$ converges almost surely in finite time to an element of the set $(F_1 \cap \cdots \cap F_m)$, i.e., there exist almost surely $\tau \in \mathbb{N}$ and $\bar{x} \in (F_1 \cap \cdots \cap F_m)$ such that $x_n = \bar{x}$ for $n \geq \tau$.

2.3.3 Algorithm as Set-valued Map

Our next step is to show that the evolution determined by the Discretized Gossip Coverage Algorithm can be seen as a set-valued map. To this end, for any pair of robots $(i, j) \in \{1, \ldots, N\}^2$, $i \neq j$, we define the map $T_{ij} : \mathrm{Part}_N(Q) \to \mathrm{Part}_N(Q)$ by

$$T_{ij}(P) = (P_1, \ldots, \widehat{P}_i, \ldots, \widehat{P}_j, \ldots, P_N),$$

where $\widehat{P}_i = W_{a^*}$ and $\widehat{P}_j = W_{b^*}$.

If at time $t \in \mathbb{R}_{\geq 0}$ the pair (i, j) and no other pair of robots perform an iteration of the Pairwise Partitioning Rule, then the dynamical system on the space of partitions is described by

$$P(t^+) = T_{ij}\left(P(t)\right). \tag{2.4}$$

We define the set-valued map $T : \mathrm{Part}_N(Q) \rightrightarrows \mathrm{Part}_N(Q)$ as

$$T(P) = \{T_{ij}(P) \,|\, (i,j) \in \{1,\ldots,N\}^2, i \neq j\}. \tag{2.5}$$

Observe that (2.4) can then be rewritten as $P(t^+) \in T(P(t))$.

2.3.4 Persistence of Communication

We begin with a property of the Random Destination & Wait Motion Protocol which is needed to show the persistence of pairwise exchanges.

Lemma 2.3.2 *Consider a group of N robots implementing the Discretized Gossip Coverage Algorithm starting from an arbitrary initial $P \in \mathrm{Part}_N(Q)$. Consider $t \in \mathbb{R}_{\geq 0}$ and let $P(t)$ denote the partition at time t. Assume that at time t no two robots are communicating. Then, there exist $\Delta > 0$ and $\alpha \in (0,1)$, independent of $P(t)$ and the positions and states of the robots at time t, such that, for every $(i,j) \in \mathcal{E}(P(t))$, $\mathbb{P}\left[(i,j) \text{ communicate within } (t, t+\Delta)\right] \geq \alpha$.*

Proof. To begin, we define two useful quantities. Let

$$\mathcal{S}(Q) := \max_{P \in \mathrm{Part}_N(Q)} \max_{P_i \in P} \max_{h,k \in P_i} d_{P_i}(h,k)$$

be a pseudo-diameter for Q, and then choose $\Delta := 2\frac{\mathcal{S}(Q)}{v} + 2\tau$. We fix a pair $(i,j) \in \mathcal{E}(P)$, and pick adjacent vertices $a \in P_i$, $b \in P_j$.

Our goal is to lower bound the probability that i and j will communicate within the interval $(t, t+\Delta)$. To do so we construct *one* sequence of events of positive probability which enables such communication. Consider the following situation: i is in the *moving* state and needs time t_i to reach its destination q_i, whereas robot j is in the *waiting* state at vertex q_j and must wait there for time $\tau_j \leq \tau$. We denote by $t(a)$ (resp. $t(b)$) the time needed for i (resp. j) to travel from q_i (resp. q_j) to a (resp. b). Let E_i be the event such that i performs the following actions in $(t, t+\Delta)$ without communicating with any robot $k \neq j$:

(i) i reaches q_i and waits at q_i for the duration τ; and

(ii) i chooses vertex a as its next destination and then stays at a for at least $\Delta - t(a) - t_i - \tau$.

Let E_j be the event such that j performs the following actions in $(t, t+\Delta)$ without communicating with any $k \neq i$:

(i) j waits at q_j for the duration τ_j; and

(ii) j chooses vertex b as its next destination and then stays at b for at least $\Delta - t(b) - \tau_j$.

Let $E_{ij} = E_i \cap E_j$.

Next, we lower bound the probability that event E_i occurs. Recall the definition of λ_{comm} from Sec. 2.2.1. Since a robot can have at most $N-1$ neighbors, the probability that (i) of E_i happens is lower bounded by $e^{-\lambda_{\text{comm}}\tau N}$. For (ii), the probability that i chooses a is $1/|P_i|$, which is lower bounded by $1/|Q|$. Then, in order to spend at least $(\Delta - t(a) - t_i - \tau)$ at a, i must choose a for $\lceil \frac{\Delta - t(a) - t_i - \tau}{\tau} \rceil$ consecutive times. Finally, the probability that during this interval i will not communicate with any robot other than j is lower bounded by $e^{-\lambda_{\text{comm}}(\Delta)(N-2)}$. The probability that (ii) occurs is thus lower bounded by $(1/|Q|)^{\lceil \frac{\Delta}{\tau} \rceil} e^{-\lambda_{\text{comm}}\Delta N}$. Combining the bounds for (i) and (ii), it follows that

$$\mathbb{P}[E_i] \geq \left(\tfrac{1}{|Q|}\right)^{\lceil \frac{\Delta}{\tau} \rceil} e^{-\lambda_{\text{comm}}(\Delta+\tau)N}.$$

The same lower bound holds for $\mathbb{P}[E_j]$, meaning that

$$\mathbb{P}\left[E_{ij}\right] = \mathbb{P}\left[E_i\right] \mathbb{P}\left[E_j\right] \geq \left(\tfrac{1}{|Q|}\right)^{2\lceil \frac{\Delta}{\tau} \rceil} e^{-2\lambda_{\text{comm}}(\Delta+\tau)N}.$$

If event E_{ij} occurs, then robots i and j will be at adjacent vertices for an amount of time during the interval $(t, t+\Delta)$ equal to

$$t_{\min} = \min\left\{\Delta - t(a) - t_i - \tau, \Delta - t(b) - \tau_j\right\}.$$

Since $t(a)$ and $t(b)$ are no more than $\frac{\mathcal{S}(Q)}{v}$, we can conclude that i and j will be within r_{comm} for at least $t_{\min} \geq \tau$. Conditioned on E_{ij} occurring, the probability that i and j communicate in $(t, t+\Delta)$ is thus lower bounded by $1 - e^{-\lambda_{\text{comm}}\tau}$. A suitable choice for α from the statement of the Lemma is then

$$\alpha = \left(\tfrac{1}{|Q|}\right)^{2\lceil \frac{\Delta}{\tau} \rceil} e^{-2\lambda_{\text{comm}}(\Delta+\tau)N} \left(1 - e^{-\lambda_{\text{comm}}\tau}\right).$$

It can be shown that this also constitutes a lower bound for the other possible combinations of initial states: robot i is *waiting* and robot j is *moving*; robots i and j are both *moving*; and robots i and j are both *waiting*. ∎

The next two Propositions state facts whose validity is ensured by Lemma 2.3.2.

Proposition 2.3.3 (Persistence of Exchanges) *Consider N robots implementing the Discretized Gossip Coverage Algorithm. Then, there almost surely exists an increasing sequence of time instants $\{t_k\}_{k\in\mathbb{Z}_{\geq 0}}$ such that $P(t_k^+) = T_{ij}(P(t_k))$ for some $(i,j) \in \mathcal{E}(P(t_k))$.*

Proof. The proof follows directly from Lemma 2.3.2 which implies that the time between two consecutive pairwise communications is almost surely finite. ∎

The existence of time sequence $\{t_k\}_{k\in\mathbb{Z}_{\geq 0}}$ allows us to to express the evolution generate by the Discretized Gossip Coverage Algorithm as a discrete time process. Let $P(k) := P(t_k)$ and $P(k+1) := P(t_k^+)$, then

$$P(k+1) \in T\left(P(k)\right)$$

where $T : \mathrm{Part}_N(Q) \rightrightarrows \mathrm{Part}_N(Q)$ is defined as in (2.5).

Given $k \in \mathbb{Z}_{\geq 0}$, let $\mathcal{I}_k$ denote the information which completely characterizes the state of Discretized Gossip Coverage Algorithm just after the k-th iteration of the partitioning rule, i.e., at time t_{k-1}^+. Specifically, $\mathcal{I}_k$ contains the information related to the partition $P(k)$, the positions of the robots at t_{k-1}^+, and whether each robot is in the *waiting* or *moving* state at t_{k-1}^+. The following result characterizes the probability that, given $\mathcal{I}_k$, the $(k+1)$-th iteration of the partitioning rule is governed by any of the maps T_{ij}, $(i,j) \in \mathcal{E}(P(k))$.

Proposition 2.3.4 (Probability of Communication) *Consider a team of N robots with capacities (C1), (C2), (C3), and (C4) implementing the Discretized Gossip Coverage Algorithm. Then, there exists a real number $\bar{\pi} \in (0,1)$, such that, for any $k \in \mathbb{Z}_{\geq 0}$ and $(i,j) \in \mathcal{E}(P(k))$*

$$\mathbb{P}\left[P(k+1) = T_{ij}(P(k)) \mid \mathcal{I}_k\right] \geq \bar{\pi}.$$

Proof. Assume that at time $\bar{t}$ one pair of robots communicates. Given a pair $(\bar{i},\bar{j}) \in \mathcal{E}(P(\bar{t}))$, we must find a lower bound for the probability that $(\bar{i},\bar{j})$ is the communicating pair. Since all the Poisson communication processes have the same intensity, the distribution of the chance of communication is uniform over the pairs

which are "able to communicate," i.e., closer than r_{comm} to each other. Thus, we must only show that $(\bar{i}, \bar{j})$ has a positive probability of being able to communicate at time $\bar{t}$, which is equivalent to showing that $(\bar{i}, \bar{j})$ is able to communicate for a positive fraction of time with positive probability. The proof of Lemma 2.3.2 implies that with probability at least $\alpha/(1 - e^{-\lambda_{\text{comm}}\tau})$ any pair in $\mathcal{E}(P(\bar{t}))$ is able to communicate for a fraction of time not smaller than $\frac{\tau}{\Delta}$, where α and Δ are defined in the proof of Lemma 2.3.2. Hence the result follows. ∎

The property in Proposition 2.3.4 can also be formulated as follows. Let $\sigma : \mathbb{Z}_{\geq 0} \to \{(i,j) \in \{1,\dots,N\}^2, i \neq j\}$ be the stochastic process such that $\sigma(k)$ is the communicating pair at time k. Then, the sequence of pairs of robots performing the partitioning rule at time instants $\{t_k\}_{k \in \mathbb{Z}_{\geq 0}}$ can be seen as a realization of the process σ, which satisfies

$$\mathbb{P}\big[\sigma(k+1) = (i,j) \mid \sigma(k)\big] \geq \bar{\pi} \tag{2.6}$$

for all $(i,j) \in \mathcal{E}(P(k))$.

2.3.5 Lyapunov Function

Next we show that the cost function decreases whenever the application of T from (2.5) changes the territory partition. This fact is a key ingredient to apply Lemma 2.3.1.

Lemma 2.3.5 (Decreasing Cost Function) *Let $P \in \text{Part}_N(Q)$ and let $P^+ \in T(P)$. If $P^+ \neq P$, then $\mathcal{H}_{\text{expected}}(P^+) < \mathcal{H}_{\text{expected}}(P)$.*

Proof. Without loss of generality assume that (i,j) is the pair executing the Pairwise Partitioning Rule. Then

$$\begin{aligned}\mathcal{H}_{\text{expected}}(P^+) - \mathcal{H}_{\text{expected}}(P) = \mathcal{H}_{\text{one}}(\text{Cd}(P_i^+); P_i^+) + \mathcal{H}_{\text{one}}(\text{Cd}(P_j^+); P_j^+) \\ - \mathcal{H}_{\text{one}}(\text{Cd}(P_i); P_i) - \mathcal{H}_{\text{one}}(\text{Cd}(P_j); P_j).\end{aligned}$$

According to the definition of the Pairwise Partitioning Rule we have that if $P_i^+ \neq P_i$, $P_j^+ \neq P_j$, then

$$\mathcal{H}_{\text{one}}(\text{Cd}(P_i^+); P_i^+) + \mathcal{H}_{\text{one}}(\text{Cd}(P_j^+); P_j^+) < \mathcal{H}_{\text{one}}(\text{Cd}(P_i); P_i) + \mathcal{H}_{\text{one}}(\text{Cd}(P_j); P_j)$$

from which the statement follows. ∎

2.3.6 Proof of Main Convergence Result

We now complete the proof of our main convergence result for the Discretized Gossip Coverage Algorithm, Theorem 2.2.6.

Proof. [Proof of Theorem 2.2.6 statement (ii)] Note that the algorithm evolves in a finite space of partitions, and by Theorem 2.2.6 statement (i), the set $\text{Part}_N(Q)$ is strongly positively invariant. This fact implies that assumption (i) of Lemma 2.3.1 is satisfied. From Lemma 2.3.5 it follows that assumption (ii) is also satisfied, with $\mathcal{H}_{\text{expected}}$ playing the role of the function U. Finally, the property in (2.6) is equivalent to the property of *persistent random switches* stated in Assumption (iii) of Lemma 2.3.1, for the special case $h = 1$. Hence, we are in the position to apply Lemma 2.3.1 and conclude convergence in finite-time to an element of the intersection of the equilibria of the maps T_{ij}, which by definition is the set of the pairwise-optimal partitions. ∎

2.4 Experimental Methods & Results

To demonstrate the utility and study practical issues of the Discretized Gossip Coverage Algorithm, we implemented it using the open-source Player/Stage robot control system [39] and the Boost Graph Library (BGL) [40]. All results presented here were generated using Player 2.1.1, Stage 2.1.1, and BGL 1.34.1. To compute distances in uniform edge weight graphs we extended the BGL breadth-first search routine with a distance recorder event visitor.

2.4.1 Large-scale Simulation

To evaluate the performance of our gossip coverage algorithm with larger teams, we tested 30 simulated robots partitioning a map representing a $350m \times 225m$ portion of campus at the University of California at Santa Barbara. As shown in Fig. 2.3, the robots are tasked with providing coverage of the open space around some of the buildings on campus, a space which includes a couple open quads, some narrower passages between buildings, and a few dead-end spurs. For this large environment the simulated robots are $2m$ on a side and can move at $3.0\frac{m}{s}$. Each territory cell is $3m \times 3m$.

In this simulation we handle communication and partitioning as follows. The communication range is set to $30m$ (10 edges in the graph) with $\lambda_{\text{comm}} = 0.3\frac{\text{comm}}{s}$. The robots wait at their destination vertices for $\tau = 3.5s$. This value for τ was chosen so that on average one quarter of the robots are waiting at any moment.

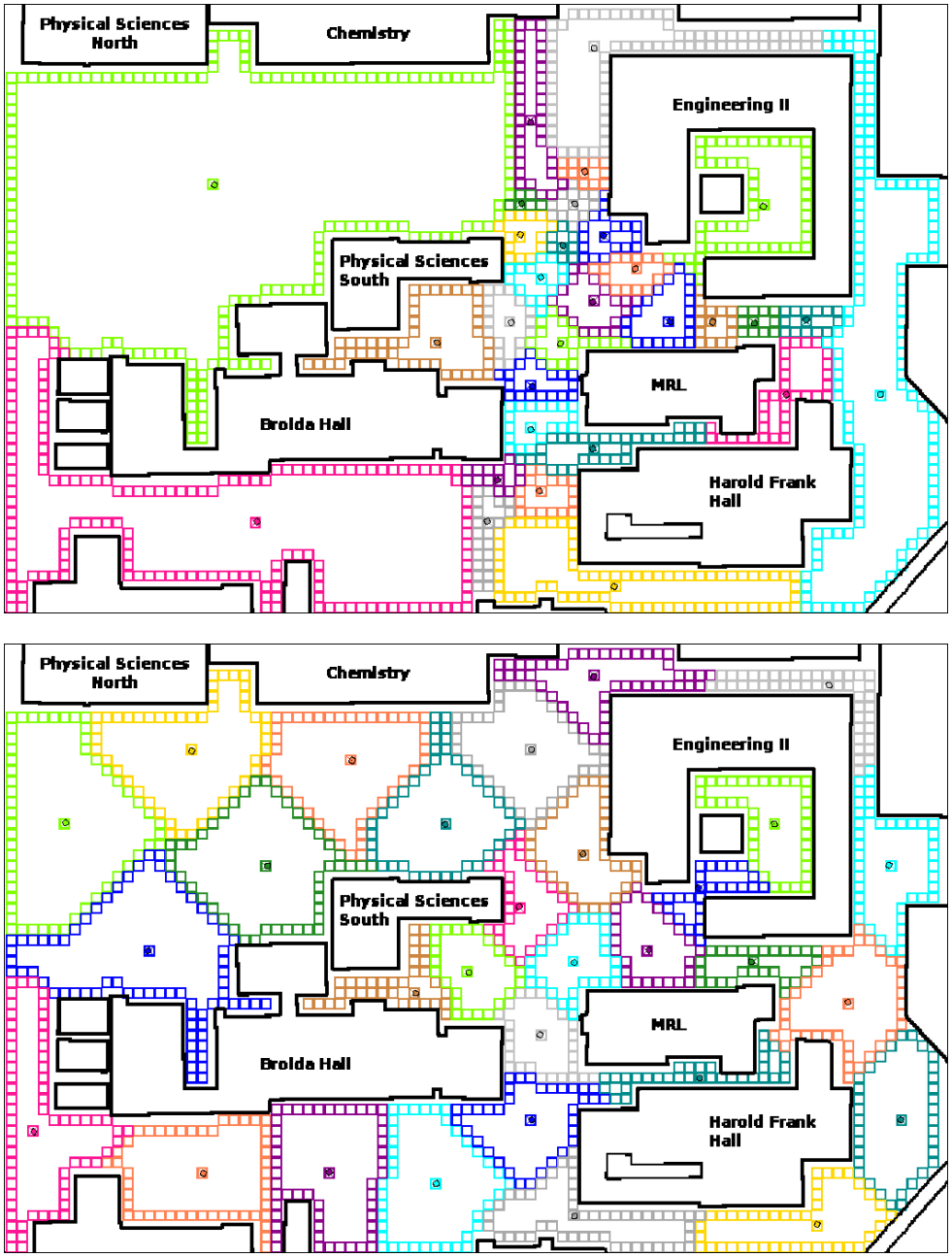


Figure 2.3: Images of starting and final partitions for a simulation with 30 robots providing coverage of a portion of campus at UCSB.

Lower values of τ mean the robots are moving more of the time and as a result more frequently miss connections, while for higher τ the robots spend more time stationary which also reduces the rate of convergence. With the goal of improving communication, we implemented a minor modification to the motion protocol: each robot picks its random destination from the cells forming the open boundary[2] of its territory. In our implementation, the full partitioning loop may take 5 seconds for the largest initial territories in Fig. 2.3. We chose to stop the loop after a quarter second for this simulation to verify the anytime computation claim.

The 30 robots start clustered in the center of the map between Engineering II and Broida Hall, and an initial Voronoi partition is generated from these starting positions. This initial partition is shown on the left in Fig. 2.3 with the robots positioned at the centroids of their starting regions. The initial partition has a cost of $37.1m$. The team spends about 27 minutes moving and communicating according to the Discretized Gossip Coverage Algorithm before settling on the final partition on the right of Fig. 2.3. The coverage cost of the final equilibrium improved by 54% to $17.1m$. Visually, the final partition is also dramatically more uniform than the initial condition. This result demonstrates that the algorithm is effective for large teams in large non-convex environments.

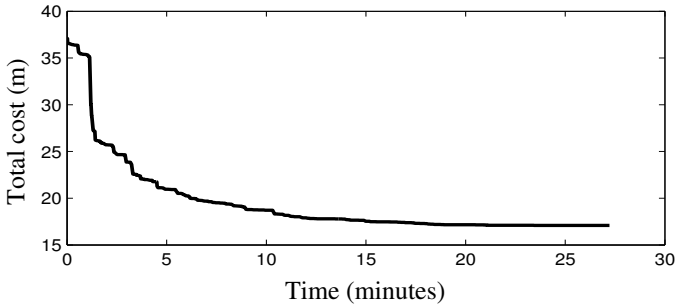


Figure 2.4: Graph of the cost $\mathcal{H}_{\text{expected}}$ over time for the simulation in Fig. 2.3.

Fig. 2.4 shows the evolution of $\mathcal{H}_{\text{expected}}$ during the simulation. The largest cost improvements happen early when the robots that own the large territories on the left and right of the map communicate with neighbors with much smaller territories. These big territory changes then propagate through the network as the robots meet and are pushed and pulled towards a lower cost partition.

[2]The open boundary of P_i is the set of vertices in P_i which are adjacent to at least one vertex owned by another agent.

2.4.2 Implementation Details

We conducted an experiment to test the algorithm using three physical robots in our lab, augmented by six simulated robots in a synthetic environment extending beyond the lab. Our lab space is $11.3m$ on a side and is represented by the upper left portion of the territory maps in Fig. 2.6. The territory graph loops around a center island of desks. We extended the lab space through three connections into a simulated environment around the lab, producing a $15.9m \times 15.9m$ environment. The map of the environment was specified with a $0.15m$ bitmap which we overlayed with a $0.6m$ resolution occupancy grid representing the free territory for the robots to cover. The result is a lattice-like graph with all edge weights equal to $0.6m$. The $0.6m$ resolution was chosen so that our physical robots would fit easily inside a cell.

Additional details of our implementation are as follows.

Robot hardware

Figure 2.5: Erratic mobile robot with Hokuyo URG-04LX laser rangefinder.

We use Erratic mobile robots from Videre Design, as shown in Fig. 2.5. The vehicle platform has a roughly square footprint ($40cm \times 37cm$), with two differential drive wheels and a single rear caster. Each robot carries an onboard computer with a 1.8Ghz Core 2 Duo processor, 1 GB of memory, and 802.11g wireless communication. For navigation and localization, each robot is equipped with a Hokuyo URG-04LX laser rangefinder. The rangefinder scans 683 points over 240° at $10Hz$ with a range of 5.6 meters.

Experiment setup

Our mixed physical and virtual robot experiments are run from a central computer which is attached to a wireless router so it can communicate with the physical robots. The central computer creates a simulated world using Stage which

mirrors and extends the real space in which the physical robots operate. The central computer also simulates the virtual members of the robot team. These virtual robots are modeled off of our hardware: they are differential drive with the same geometry as the Erratic platform and use simulated Hokuyo URG-04LX rangefinders.

Localization

We use the `amcl` driver in Player which implements Adaptive Monte-Carlo Localization [41]. The physical robots are provided with a map of our lab with a $15cm$ resolution and told their starting pose within the map. We set an initial pose standard deviation of $0.9m$ in position and $12°$ in orientation, and request localization updates using 50 of the sensor's range measurements for each change of $2cm$ in position or $2°$ in orientation reported by the robot's odometry system. We then use the most likely pose estimate output by `amcl` as the location of the robot. For simplicity and reduced computational demand, we allow the virtual robots access to perfect localization information.

Motion Protocol

Each robot continuously executes the Random Destination & Wait Motion Protocol, with navigation handled by the `snd` driver in Player which implements Smooth Nearness Diagram navigation [42]. For `snd` we set the robot radius parameter to $22cm$, obstacle avoidance distance to $0.7m$, and maximum speeds to $0.4\frac{m}{s}$ and $40\frac{^\circ}{s}$. The `snd` driver is a local obstacle avoidance planner, so we feed it a series of waypoints every couple meters along paths found in $G(Q)$. We consider a robot to have achieved its target location when it is within $20cm$ and it will then wait for $\tau = 3.5s$. For the physical robots the motion protocol and navigation processes run on board, while there are separate threads for each virtual robot on the central computer.

Communication and Partitioning

As the robots move, a central process monitors their positions and simulates the range-limited gossip communication model between both real and virtual robots. We set $r_{comm} = 2.5m$ and $\lambda_{comm} = 0.3\frac{\text{comm}}{s}$. These parameters were chosen so that the robots would be likely to communicate when seperated by at most four edges, but would also sometimes not connect despite being close. When this process determines two robots should communicate, it informs the robots who

then perform the Pairwise Partitioning Rule. Our pairwise communication implementation is blocking: if robot i is exchanging territory with j, then it informs the match making process that it is unavailable until the exchange is complete.

2.4.3 Hardware-in-the-Loop Simulation

The results of our experiment with three physical robots and six simulated robots are shown in Figs. 2.6 and 2.7. The first row in Fig. 2.6 shows the starting positions of the team of robots, with the physical robots, labeled 1, 2, and 3, lined up in a corner of the lab and the simulated robots arrayed around them. The starting positions are used to generate the initial Voronoi partition of the environment. The physical robots own the orange, blue, and lime green territories in the upper left quadrant of the environment. We chose this initial configuration to have a high coverage cost, while ensuring that the physical robots will remain in the lab as the partition evolves.

In the middle row, robots 1 and 2 have met along their shared border and are exchanging territory. In the territory map, the solid red line indicates 1 and 2 are communicating and their updated territories are drawn with solid orange and blue, respectively. The camera view confirms that the two robots have met on the near side of the center island of desks.

The final partition in the last row in Fig. 2.6 is reached after $9\frac{1}{2}$ minutes. All of the robots are positioned at the centroids of their final territories. The three physical robots have gone from a cluster in one corner of the lab to a more even spread around the space. Fig. 2.7 shows the evolution of the cost function $\mathcal{H}_{\text{expected}}$ as the experiment progresses, including the costs for each robot. As expected, the total cost never increases and the disparity of costs for the individual robots shrinks over time until settling at a pairwise-optimal partition.

In this experiment the hardware challenges of sensor noise, navigation, and uncertainty in position were efficiently handled by the `amcl` and `snd` drivers. The coverage algorithm assumed the role of a higher-level planner, taking in position data from `amcl` and directing `snd`. By far the most computationally demanding component was `amcl`, but the position hypotheses from `amcl` are actually unnecessary: our coverage algorithm only requires knowledge of the vertex a robot occupies. If a less intensive localization method is available, the algorithm could run on robots with significantly lower compute power.

Figure 2.6: Each row contains a territory map and the corresponding overhead camera image for a step of the hardware-in-the-loop simulation. The position of the camera in the environment is shown with a camera icon in the territory map. The physical robots are numbered 1, 2, and 3 and have the orange, blue, and lime green partitions. Their positions in each territory map are indicated with numbered circles.

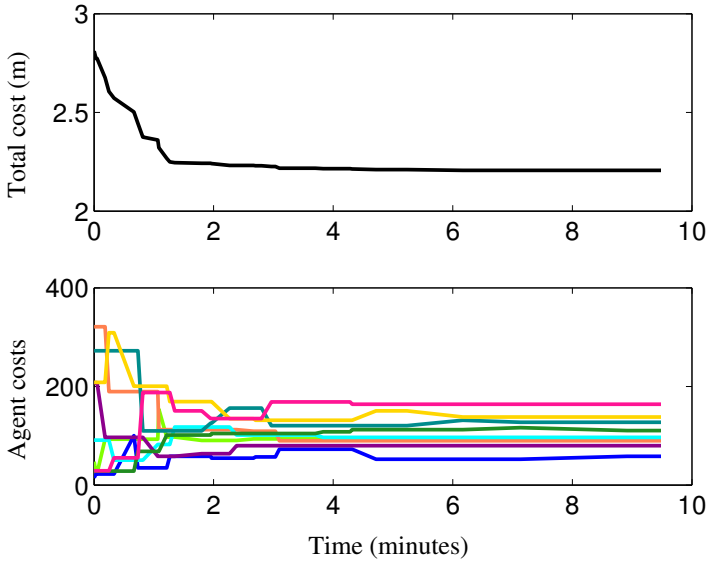


Figure 2.7: Evolution of cost functions during the experiment in Fig. 2.6. The total cost $\mathcal{H}_{\text{expected}}$ is shown above in black, while $\mathcal{H}_{\text{one}}$ for each robot is shown below in the robot's color.

2.4.4 Comparative analysis

In this subsection we present a numerical comparison of the performance of the Discretized Gossip Coverage Algorithm and the following two Lloyd-type algorithms.

Decentralized Lloyd Algorithm

This method is from [6] and [4] and was reviewed in Section 2.1.6, we repeat it here for convenience. At each discrete time instant $t \in \mathbb{Z}_{\geq 0}$, each robot i performs the following tasks: (1) i transmits its position and receives the positions of all adjacent robots; (2) i computes its Voronoi region P_i based on the information received; and (3) i moves to $\text{Cd}(P_i)$.

Gossip Lloyd Algorithm

This method is from [16]. It is a gossip algorithm, and so we have used the same communication model and the Random Destination & Wait Motion Protocol to create meetings between robots. Say robots i and j meet at time t, then the

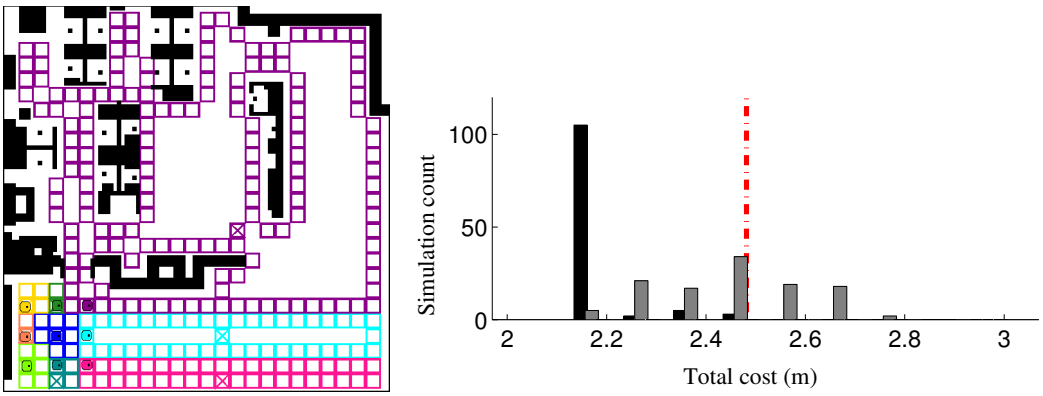


Figure 2.8: Initial partition and histogram of final costs for a Monte Carlo test comparing the Discretized Gossip Coverage Algorithm (black bars), Gossip Lloyd Algorithm (gray bars), and Decentralized Lloyd Algorithm (red dashed line). For the gossip algorithms, 116 simulations were performed with different sequences of pairwise communications. The Decentralized Lloyd Algorithm is deterministic given an initial condition so only one final cost is shown.

pairwise Lloyd partitioning rule works as follows: (1) robot i transmits $P_i(t)$ to j and vice versa; (2) both robots determine $U = P_i(t) \cup P_j(t)$; (3) robot i sets $P_i(t^+)$ to be its Voronoi region of U based on $\mathrm{Cd}(P_i(t))$ and $\mathrm{Cd}(P_j(t))$, and j does the equivalent.

For both Lloyd algorithms we use the same tie breaking rule when creating Voronoi regions as is present in the Pairwise Partitioning Rule: ties go to the robot with the lowest index.

Our first numerical result uses a Monte Carlo probability estimation method from [43] to place probabilistic bounds on the performance of the two gossip algorithms. Recall that the Chernoff bound describes the minimum number of random samples K required to reach a certain level of accuracy in a probability estimate from independent Bernoulli tests. For an accuracy $\epsilon \in (0, 1)$ and confidence $1 - \eta \in (0, 1)$, the number of samples is given by

$$K \geq \tfrac{1}{2\epsilon^2} \log \tfrac{2}{\eta}.$$

For $\eta = 0.01$ and $\epsilon = 0.1$, at least 116 samples are required.

Figure 2.8 shows both the initial territory partition of the extended laboratory environment used and also a histogram of the final results for the following Monte Carlo test. The environment and robot motion models used are described in Section 2.4.2. Starting from the indicated initial condition, we ran 116 simulations

of both gossip algorithms. The randomness in the test comes from the sequence of pairwise communications. These sequences were generated using: (1) the Random Destination & Wait Motion Protocol with q_i sampled uniformly from the open boundary of P_i and $\tau = 3.5s$; and (2) the range-limited gossip communication model with $r_{comm} = 2.5m$ and $\lambda_{comm} = 0.3\frac{\text{comm}}{s}$.

The cost of the initial partition shown in Fig. 2.8 is $5.48m$, while the best known partition for this environment has a cost just under $2.18m$. The histogram in Fig. 2.8 shows the final equilibrium costs for 116 simulations of the Discretized Gossip Coverage Algorithm (black) and the Gossip Lloyd Algorithm (gray). It also shows the final cost using the Decentralized Lloyd Algorithm (red dashed line), which is deterministic from a given initial condition. The histogram bins have a width of $0.10m$ and start from $2.17m$. For the Discretized Gossip Coverage Algorithm, 105 out of 116 trials reach the bin containing the best known partition with a mean final cost of $2.23m$. The Gossip Lloyd Algorithm reaches the lowest bin in only 5 of 116 trials and has a mean final cost of $2.51m$. The Decentralized Lloyd Algorithm settles at $2.48m$. Our new gossip algorithm requires an average of 96 pairwise communications to reach an equilibrium, while gossip Lloyd requires 126.

Based on these results, we can conclude with 99% confidence that there is at least an 80% probability that 9 robots executing the Discretized Gossip Coverage Algorithm starting from the initial partition shown in Fig. 2.8 will reach a pairwise-optimal partition which has a cost within 4% of the best known cost. We can further conclude with 99% confidence that the Gossip Lloyd Algorithm will settle more than 4% above the best known cost at least 86% of the time starting from this initial condition.

Figure 2.9 compares final cost histograms for 10 different initial conditions for the same environment and parameters as described above. Each initial condition was created by selecting unique starting locations for the robots uniformly at random and using these locations to generate an initial Voronoi partition. The initial cost for each test is shown with the green dashed line. In 9 out of 10 tests the Discretized Gossip Coverage Algorithm reaches the histogram bin with the best known partition in at least 112 of 116 trials. The two Lloyd methods get stuck in sub-optimal centroidal Voronoi partitions more than 4% away from the best known partition in more than half the trials in 7 of 10 tests.

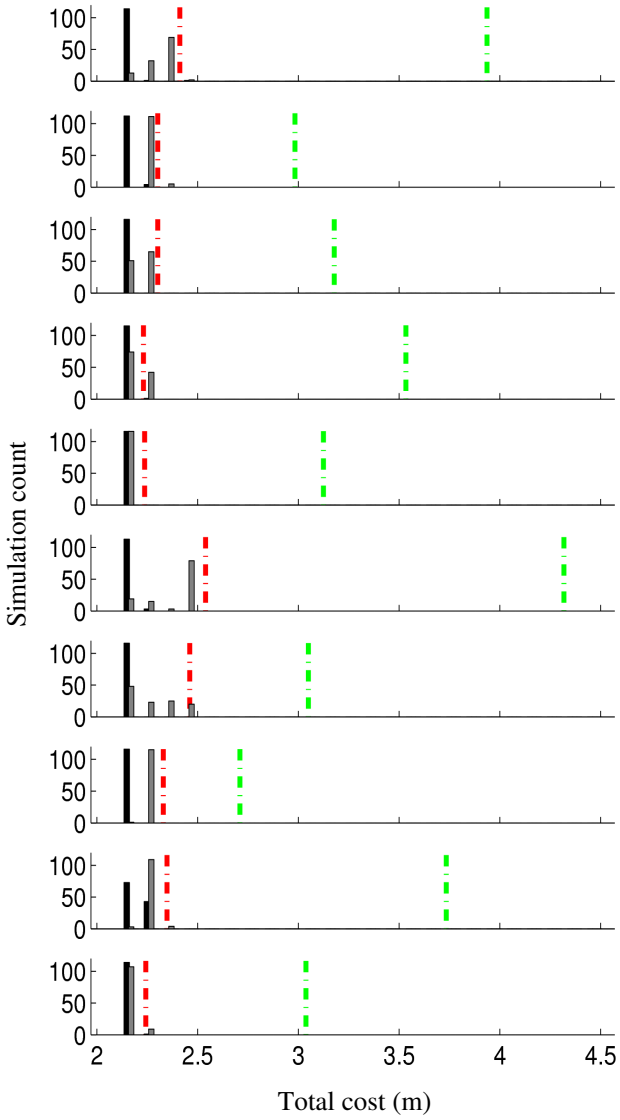


Figure 2.9: Histograms of final costs from 10 Monte Carlo tests using random initial conditions in the environment shown in Fig. 2.8 comparing Discretized Gossip Coverage Algorithm (black bars), Gossip Lloyd Algorithm (gray bars), and Decentralized Lloyd Algorithm (red dashed line). For the gossip algorithms, 116 simulations were performed with different sequences of pairwise communications. The Decentralized Lloyd Algorithm is deterministic given an initial condition so only one final cost is shown. The initial cost for each test is drawn with the green dashed line.

2.5 Summary

We have presented a novel distributed partitioning and coverage control algorithm which requires only unreliable short-range communication between pairs of robots and works in non-convex environments. The classic Lloyd approach to coverage optimization involves iteration of separate centering and Voronoi partitioning steps. For gossip algorithms, however, this separation is unnecessary computationally and we have shown that improved performance can be achieved without it. Our new Discretized Gossip Coverage Algorithm provably converges to a subset of the set of centroidal Voronoi partitions which we labeled pairwise-optimal partitions. Through numerical comparisons we demonstrated that this new subset of solutions avoids many of the local minima in which Lloyd-type algorithms can get stuck.

Our vision is that this partitioning and coverage algorithm will form the foundation of a distributed task servicing setup for teams of mobile robots. The robots would split their time between servicing tasks in their territory and moving to contact their neighbors and improve the coverage of the space. Our convergence results only require sporadic improvements to the cost function, affording flexibility in robot behavior.

Chapter 3

Coverage with One-to-Base Communication

In the previous chapter we presented a solution to the coverage control problem assuming that the robots are able to communicate peer-to-peer, but in some environments this is impractical. For example, underwater acoustic communication between ocean gliders is very low bandwidth and hilly or urban terrain can block peer-to-peer radio communication. To address this issue, in this chapter we present a coverage control algorithm for a team of robots who each have occasional contact with a central base station (one-to-base-station communication). This communication model can represent the surfacing of ocean gliders to communicate with a tower on the shore as illustrated in Fig. 3.1 [1], the use of a UAV as a data mule which periodically visits ground robots [44], or the cost-mindful use of satellite or cellular communication. Our algorithm optimizes the response time of the team to service requests in a non-convex environment represented by a graph, with optimality defined by a relevant "multi-center" cost function for overlapping territories. While the algorithm is given for one-to-base-station communication, it also works if each robot can occasionally broadcast a message directly to the rest of the team.

There are three main contributions of this work. First, we present the first coverage control algorithm for an asynchronous one-to-base-station communication model. This communication model is realistic and relevant for a variety of application domains, and the time delay between when different robots communicate with the base station requires novel tools. Our algorithm evolves overlapping coverings of graphs and being graph-based also enables it to work in complex non-convex environments.

Second, we prove that our algorithm converges to a centroidal Voronoi partition

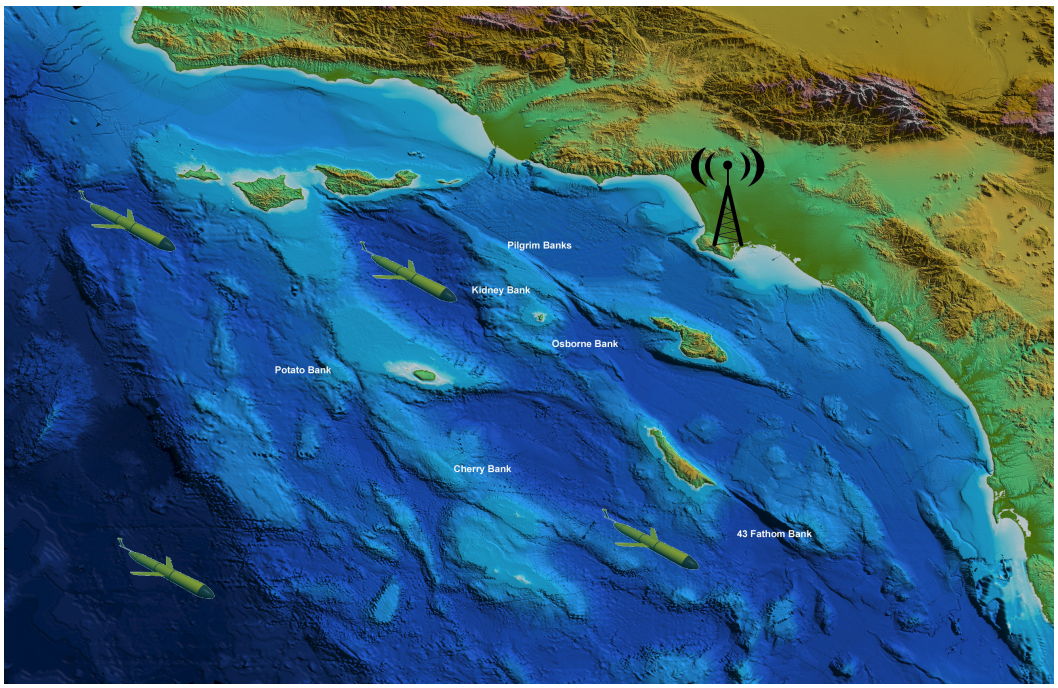


Figure 3.1: Illustration of four underwater gliders operating off the coast of Southern California and communicating with a central radio tower. Underwater robotic systems of this type are already in use in the area, see the Southern California Coastal Ocean Observing System (www.sccoos.org) or the projects run by the USC Center for Integrated Networked Aquatic PlatformS [1].

in finite time. Our Lyapunov argument is based on an adaptation of the standard partition-based coverage cost function. Operating on overlapping coverings also requires reconsidering when to perform the classic Lloyd steps of centering versus territory exchange.

Third, we describe how our algorithm can seamlessly handle the unscheduled arrival or departure of robots from the team. This feature leverages the fact that our algorithm is using overlapping regions, and also would ease integration of coverage control with some form of task servicing by the robot team.

This chapter is organized as follows. In Section 3.1 we adapt the strict partitioning setup from the previous chapter to overlapping coverings of graphs. In Section 3.2 we present the one-to-base communication model, problem formulation, and our proposed solution. Section 3.3 contains the proof that our algorithm converges to a centroidal Voronoi partition at equilibrium. In Section 3.4 we discuss practical issues and explain how our algorithm can adapt to dynamic changes

to the robot team and in Section 3.5 we provide some numerical results. We then summarize this chapter in Section 3.6.

The work in this chapter was done in collaboration with R. Carli and P. Frasca.

3.1 Preliminary Material

There is significant overlap between the concepts for partitions of graphs in Section 2.1 and what we need in this section. However, the one-to-base-station communication model in this chapter requires overlapping coverings, instead of a partition. Here we will detail only those features which differ from Section 2.1.

3.1.1 Coverings of Graphs

We will be covering Q with n subsets or regions which will each be owned by an individual agent.

Definition 3.1.1 (n-Covering) *Given the graph $G(Q) = (Q, E, w)$, we define a* $n-$covering of Q *as a collection $P = \{P_i\}_{i=1}^n$ of subsets of Q such that:*

(i) $\bigcup_{i=1}^n P_i = Q$;

(ii) $P_i \neq \emptyset$ *for all* $i \in \{1, \ldots, n\}$;

Let $\mathrm{Cov}_n(Q)$ *to be the set of* $n-$*coverings of* Q.

Note that with this definition a vertex in Q may belong to multiple subsets in P, i.e., a vertex may be covered by multiple agents. This fact is an important change from the gossip-based approach in Chapter 2 and our prior work [16, 45].

We also have use for the concept of a partition of Q, but in this case we do not require that the partition be connected.

Definition 3.1.2 (n-Partition) *A* n-partition *is a n-covering with the additional property that:*

(iii) if $i \neq j$, *then* $P_i \cap P_j = \emptyset$.

Let $\mathrm{Part}_N(Q)$ *to be the set of* $n-$*partitions of* Q.

Again we recall the useful definition of a Voronoi partition: Given a vector of distinct points $c \in Q^n$, the partition $P \in \mathrm{Part}_N(Q)$ is said to be a *Voronoi partition of Q generated by c* if, for each P_i and all $k \in P_i$, we have $c_i \in P_i$ and $d_G(k, c_i) \leq d_G(k, c_j)$, $\forall j \neq i$. The elements of c are said to be the generators of the Voronoi partition. Note that the Voronoi partition generated by c is not unique since how to assign tied vertices is unspecified.

3.1.2 Cost Functions

In this chapter we will have use for two cost functions, $\mathcal{H}_{\text{one}}$ from Section 2.1 and $\mathcal{H}_{\max}$ which is designed specifically for overlapping coverings. We define the following multi-center function $\mathcal{H}_{\max} : Q^N \times \text{Cov}_n(Q) \to \mathbb{R}_{\geq 0}$ to measure the cost for n robots to cover a n-covering P from the vertex set $c \in Q^n$:

$$\mathcal{H}_{\max}(c, P) = \frac{1}{|Q|} \sum_{k \in Q} \max_i \{d_G(c_i, k) \mid k \in P_i\} \phi(k)$$

We aim to minimize the performance function $\mathcal{H}_{\max}$ with respect to both the covering P and the vertices c. In the motivational scenario we are considering, each robot will periodically be asked to perform a task somewhere in its region with tasks located according to distribution ϕ. When idle, the robots would position themselves at the vertices c. By minimizing $\mathcal{H}_{\max}$, the robot team would minimize the expected distance between a task and the furthest robot which can service the task.

Proposition 3.1.3 (Properties of $\mathcal{H}_{\max}$) *Let $P \in \text{Cov}_n(Q)$, $P' \in \text{Part}_N(Q)$, and $c \in Q^n$ such that $c_i \in P'_i \subset P_i \ \forall \ i$. Let $c' \in Q^n$ such that $c'_i \in \text{C}(P'_i) \ \forall \ i$. Then the following statements hold:*

$$\mathcal{H}_{\max}(c, P') \leq \mathcal{H}_{\max}(c, P), \text{ and}$$
$$\mathcal{H}_{\max}(c', P') \leq \mathcal{H}_{\max}(c, P').$$

The second inequality is strict if any $c_i \notin \text{C}(P'_i)$.

Proof. The first statement is a straightforward consequence of the restriction that $P'_i \subset P_i$ and that $\mathcal{H}_{\max}$ uses the maximum cost over i. The second statement is a result of the fact that, since P' is a partition, $\mathcal{H}_{\max}(c, P') = \frac{1}{|Q|} \sum_i \mathcal{H}_{\text{one}}(c_i; P'_i)$. ∎

Notice that Proposition 3.1.3 is similar to Proposition 2.1.5 for $\mathcal{H}_{\text{multicenter}}$, with the main change being that the second statement is only true for partitions and not for coverings. This difference will drive certain changes in our coverage algorithm in this chapter. Proposition 3.1.3 again motivates the definition of a centroidal Voronoi partition, which we repeat here for convenience.

Definition 3.1.4 (Centroidal Voronoi Partition) *$P \in \text{Part}_N(Q)$ is a* centroidal Voronoi partition *of Q if there exists a $c \in Q^n$ such that P is a Voronoi partition generated by c and $c_i \in \text{C}(P_i) \ \forall \ i$.*

For a given environment Q, a pair made of a centroidal Voronoi partition and the corresponding vector of centroids is locally optimal in the following sense: $\mathcal{H}_{\max}$ cannot be reduced by changing either P or c independently. Therefore, if the team of robots position themselves at the centroids of a centroidal Voronoi partition, then they (locally) optimize their coverage of Q as measured by $\mathcal{H}_{\max}$.

3.2 Models, Problem Formulation, and Proposed Solution

3.2.1 Robot Network Model with Asynchronous One-to-Base-Station Communication

Our *One-to-Base Coverage Algorithm* is designed for a team of n robotic agents and a central base station. Each agent $i \in \{1, \ldots, n\}$ is required to have the following basic computation capabilities:

(C1) agent i must be able to identify itself to the base station; and

(C2) agent i must have a processor with the ability to store $S_i \subset G(Q)$ and a center $s_i \in S_i$.

Each agent $i \in \{1, \ldots, n\}$ is assumed to communicate with the base station according to the *asynchronous one-to-base-station communication model* described as follows:

(C3) there exists a finite upper bound Δ on the time between communications between i and the base station. For simplicity, we assume no two agents communicate with the base station at the same time.

The base station must have the following computation capabilities:

(C4) it must be able to store an arbitrary n-covering of Q, $P = \{P_i\}_{i=1}^n$ and a list of centroids $c \in Q^n$; and

(C5) it must have the ability to perform operations on subgraphs of $G(Q)$.

3.2.2 Problem Statement

Assume that, for all $t \in \mathbb{R}_{\geq 0}$, each agent $i \in \{1, \ldots, n\}$ maintains in memory a subset $S_i(t)$ of environment Q and a vertex $s_i(t) \in S_i(t)$. Our goal is to iteratively

update the covering $S(t) = \{S_i(t)\}_{i=1}^n$ and the centers $s(t) = \{s_i(t)\}_{i=1}^n$ while solving the optimization problem:

$$\min_{s \in Q^n} \min_{S \in \mathrm{Cov}_n(Q)} \mathcal{H}_{\max}(s, S), \tag{3.1}$$

subject to the constraints imposed by the robot network model with asynchronous one-to-base-station communication from Section 3.2.1.

3.2.3 The One-to-Base Coverage Algorithm

To solve the minimization problem (3.1), we introduce the following *One-to-Base Coverage Algorithm.*

Algorithm 3.1: One-to-Base Coverage Algorithm

The base station maintains in memory an n-covering $P = \{P_i\}_{i=1}^n$ and a vector of centers $c = (c_i)_{i=1}^n$, while each robot i maintains in memory a set S_i and a vertex s_i. At $t = 0$, let $P(0) \in \mathrm{Cov}_n(Q)$, $S(0) = P(0)$, and let all $c_i(0)$'s be distinct. Assume that at time $t \in \mathbb{R}_{>0}$, robot i communicates with the base station. Let P^+, c^+, S_i^+, and s_i^+ be the values after the communication. Then the base station executes the following actions:

1: **if** $\mathcal{H}_{\mathrm{one}}(\mathrm{Cd}(P_i); P_i) < \mathcal{H}_{\mathrm{one}}(c_i; P_i)$ and $\mathrm{Cd}(P_i) \neq c_j$ for every $j \neq i$ **then**
2: update $c_i^+ := \mathrm{Cd}(P_i)$
3: **else**
4: $c_i^+ := c_i$
5: tell agent i to set $S_i^+ := P_i$ and $s_i^+ := c_i^+$
6: **for** each agent $j \neq i$ **do**
7: compute the sets
$P_{i \to j} := \{x \in P_i : d_G(x, c_j) < d_G(x, c_i^+)\}$
$P^*_{j \to i} := \{x \in P_j \cap P_i : d_G(x, c_j) \geq d_G(x, c_i^+)\}$
8: $P_j^+ := \left(P_j \setminus P^*_{j \to i}\right) \cup P_{i \to j}$

Observe that $P_{i \to j}$ contains the cells of P_i which are closer to c_j, whereas $P^*_{j \to i}$ contains only the cells in both P_i and P_j which are either closer to c_i^+ or tied. Also notice that only the centroid of the agent communicating with the base station is updated.

Remark 3.2.1 *The One-to-Base algorithm can be adapted to the scenario where each robot can occasionally broadcast a message to the team. Robot i would update its centroid and broadcast s_i^+ and S_i, then every other robot j would update*

S_j following lines 7 and 8 above. Those robots for which $S_i \cup S_j$ is connected must receive the broadcast for the convergence property to hold, the others are not required.

3.2.4 Convergence Property

In this subsection we characterize the convergence of the One-to-Base Coverage Algorithm.

Theorem 3.2.2 (Convergence Property) *Consider a network consisting of n robots endowed with computation capacities (C1), (C2) and communication capacity (C3), and a base station with capacities (C4) and (C5). Assume the network implements the* One-to-Base Coverage Algorithm. *Then the resulting evolution*

$$(s, S) : \mathbb{R}_{\geq 0} \to Q^n \times \mathrm{Cov}_n(Q)$$

converges in finite time to a pair (s^*, S^*) composed of a centroidal Voronoi partition S^* generated by s^*.

3.3 Convergence Proofs

This section is devoted to proving Theorem 3.2.2. The convergence proof is based on applying Lemma 2.3.1 from Section 2.3.2 to the evolution given by the One-to-Base Coverage Algorithm. In order to do that, we must describe the algorithm using a set valued-map and find a Lyapunov function.

3.3.1 Set-valued Map

With the definitions of a set of centroids and of the One-to-Base Coverage Algorithm, we have that the algorithm is well-posed in the following sense.

Proposition 3.3.1 (Well-posedness) *Let $P \in \mathrm{Cov}_n(Q)$ and $c \in Q^n$ such that $c_i \in P_i$ and $c_i \neq c_j$ for all i and all $j \neq i$. Then, P^+ and c^+ produced by the* One-to-Base Coverage Algorithm *meet the same criteria.*

With this result, we can state the One-to-Base Coverage Algorithm as a set valued map. For any $i \in \{1, \ldots, N\}$, we define the map $T_i : Q^n \times \mathrm{Cov}_n(Q) \to Q^n \times \mathrm{Cov}_n(Q)$ by

$$T_i(c, P) = \left\{ \{c_1, \ldots, c_i^+, \ldots, c_N\}, \{P_1^+, \ldots, P_i, \ldots, P_N^+\} \right\},$$

where c_i^+ and P^+ are defined per the algorithm when i is the communicating robot. Then, we can define the set-valued map $T : Q^n \times \mathrm{Cov}_n(Q) \rightrightarrows Q^n \times \mathrm{Cov}_n(Q)$ by

$$T(c, P) = \{T_1(c, P), \ldots, T_N(c, P)\}.$$

Thus, the dynamical system defined by the application of the algorithm is described by $\{c^+, P^+\} \in T(c, P)$.

3.3.2 Lyapunov Function

For our Lyapunov argument we need the following definitions. Let $M(P)$ be the set of vertices which are owned by multiple agents. Let $\mathcal{H}_{\min}$ be a cost function defined similarly to $\mathcal{H}_{\max}$ but sum minimum coverage costs over all agents:

$$\mathcal{H}_{\min}(c, P) = \frac{1}{|Q|} \sum_{k \in Q} \min_i \{d_G(c_i, k) \mid k \in P_i\} \phi(k).$$

Proposition 3.3.2 (Lyapunov Function) *Let P be a n-covering of Q and c be a set of centroids for P. Let $(c^+, P^+) \in T(c, P)$. If $c^+ \neq c$ or $P^+ \neq P$, then one of these conditions holds:*

(i) $\mathcal{H}_{\max}(c^+, P^+) < \mathcal{H}_{\max}(c, P)$*;*

(ii) $\mathcal{H}_{\max}(c^+, P^+) = \mathcal{H}_{\max}(c, P)$ *and* $\mathcal{H}_{\min}(c^+, P^+) < \mathcal{H}_{\min}(c, P)$*; or*

(iii) $\mathcal{H}_{\max}(c^+, P^+) = \mathcal{H}_{\max}(c, P)$, $\mathcal{H}_{\min}(c^+, P^+) = \mathcal{H}_{\min}(c, P)$, *and* $|M(P^+)| < |M(P)|$.

Proof. Consider the situation where there are just two agents i and j. Without loss of generality, assume that agent i contacts the base station at time t.

We start with the case where $c_i^+ = c_i$. First, consider when the change to P includes the addition of cells in $P_{i \to j}$ to P_j. Such a change necessarily decreases $\mathcal{H}_{\min}$ while $\mathcal{H}_{\max}$ is unchanged. Next, if the change to P occurs because of the removal of cells in $P^*_{j \to i}$ from P_j, then $\mathcal{H}_{\max}$ does not increase, $\mathcal{H}_{\min}$ is unchanged, and $|M|$ necessarily decreases.

Next, we show that if $c_i^+ \neq c_i$, then $\mathcal{H}_{\max}(c^+, P^+) < \mathcal{H}_{\max}(c, P)$. Then, define $P_{\max}$ as follows: given a $P \in \mathrm{Cov}_n(Q)$, let $P_{\max} = \{P_{i,\max}\}_{i=1}^n$ be a partition of Q such that for all i:

$$P_{i,\max} = \left\{ v \in P_i \;\middle|\; \begin{array}{l} v \notin P_j \;\forall\; j \neq i, \text{ or} \\ i = \min\left\{\mathrm{argmax}_j \{d_G(c_j, v) \mid v \in P_j\}\right\} \end{array} \right\}.$$

Note that $P_{i,\max}$ is a function of P_i, P_j, c_i, and c_j.

With the $P_{\max}$ definition, we can rewrite $\mathcal{H}_{\max}$ as:

$$\mathcal{H}_{\max}(c, P) = \frac{1}{|Q|} \sum_i \mathcal{H}_{\text{one}}(c_i, P_{i,\max})$$

Using this new form, the initial cost to cover Q by i and j is given by (ignoring $\frac{1}{|Q|}$ for simplicity):

$$\mathcal{H}_{\max}(c, P) = \mathcal{H}_{\text{one}}(c_i, P_{i,\max}) + \mathcal{H}_{\text{one}}(c_j, P_{j,\max} \setminus P_i) + \mathcal{H}_{\text{one}}(c_j, P_{j,\max} \cap P_i).$$

During the update c_i and P_j change, meaning that:

$$\mathcal{H}_{\max}(c^+, P^+) = \mathcal{H}_{\text{one}}(c_i^+, P_{i,\max}^+) + \mathcal{H}_{\text{one}}(c_j, P_{j,\max}^+ \setminus P_i) + \mathcal{H}_{\text{one}}(c_j, P_{j,\max}^+ \cap P_i).$$

The algorithm T ensures that if $c_i^+ \neq c_i$, then:

$$\mathcal{H}_{\text{one}}(c_i^+, P_i) < \mathcal{H}_{\text{one}}(c_i, P_i). \tag{3.2}$$

However, it is possible that the relevant cost for i has increased, i.e., that

$$\mathcal{H}_{\text{one}}(c_i^+, P_{i,\max}^+) > \mathcal{H}_{\text{one}}(c_i, P_{i,\max}).$$

We will show that any such increase is necessarily smaller in magnitude than the decrease in the cost to cover for j.

Two observations: First, $P_{j,\max}^+ \cap P_i = \emptyset$ by how we choose P_j^+, meaning that $\mathcal{H}_{\text{one}}(c_j, P_{j,\max}^+ \cap P_i)$ is zero. Second, the set of vertices owned by j but not by i has not changed, meaning that $\mathcal{H}_{\text{one}}(c_j, P_{j,\max}^+ \setminus P_i) = \mathcal{H}_{\text{one}}(c_j, P_{j,\max} \setminus P_i)$. This leaves us wanting to show that:

$$\mathcal{H}_{\text{one}}(c_i^+, P_{i,\max}^+) < \mathcal{H}_{\text{one}}(c_i, P_{i,\max}) + \mathcal{H}_{\text{one}}(c_j, P_{j,\max} \cap P_i).$$

We can write set P_i as:

$$\begin{aligned} P_i &= P_{i,\max} \cup (P_{j,\max} \cap P_i) \\ &= P_{i,\max}^+ \cup (P_{j,\max}^+ \cap P_i) = P_{i,\max}^+. \end{aligned}$$

Using these equivalences, we can rewrite (3.2) as:

$$\mathcal{H}_{\text{one}}(c_i^+, P_{i,\max}^+) < \mathcal{H}_{\text{one}}(c_i, P_{i,\max}) + \mathcal{H}_{\text{one}}(c_i, P_{j,\max} \cap P_i).$$

Then, using the definition of $P_{j,\max}$ we conclude that:

$$\begin{aligned}\mathcal{H}_{\text{one}}(c_i^+, P_{i,\max}^+) &< \mathcal{H}_{\text{one}}(c_i, P_{i,\max}) + \mathcal{H}_{\text{one}}(c_i, P_{j,\max} \cap P_i) \\ &< \mathcal{H}_{\text{one}}(c_i, P_{i,\max}) + \mathcal{H}_{\text{one}}(c_j, P_{j,\max} \cap P_i).\end{aligned}$$

Nothing in this analysis is exclusive to the two agent scenario. Following the same logic, it can be shown that:

$$\mathcal{H}_{\text{one}}(c_i^+, P_{i,\max}^+) - \mathcal{H}_{\text{one}}(c_i, P_{i,\max}) < \sum_{j \neq i} \mathcal{H}_{\text{one}}(c_j, P_{j,\max} \cap P_i),$$

meaning that any increase in the cost to cover for agent i from a centroid update is more than offset by decreases to the cost to cover from the territory updates of those agents who owned cells in P_i. ∎

3.3.3 Characterization of Fixed Points

The following Proposition characterizes the fixed points for $T(c, P)$, defined[1] as those pairs (c, P) such that $\{(c, P)\} = T(c, P)$, or equivalently as those pairs which are a fixed point of every map T_i.

Proposition 3.3.3 (Fixed Points) *Let $P \in \text{Cov}_n(Q)$ be $c \in Q^n$ be a fixed point of T. Then, P is a centroidal Voronoi partition of Q generated by c. Moreover, every centroidal Voronoi partition is a fixed point for T.*

Proof. If P is not a partition, then $P_{j\to i}^* \neq \emptyset$ for some $i \neq j$. If P is a partition but not a Voronoi partition generated by c, then $P_{i\to j} \neq \emptyset$ for some $i \neq j$. Finally, if P is a Voronoi partition generated by c but $c_i \notin \text{C}(P_i)$ for any i, then c_i will change when i communicates with the base station.

Next, we show that every centroidal Voronoi partition is a fixed point. If $c_i \in \text{C}(P_i)$ for all i, then $c_i^+ = c_i$ for all T_i. If P is a Voronoi partition generated by c, then $P_{i\to j} = \emptyset$, $P_{j\to i}^* = \emptyset$, and thus $P^+ = P$ for all T_i. ∎

3.3.4 Convergence of $P(t)$

The proof continues with the application of of Lemma 2.3.1 in Section 2.3.2 to $(c(t), P(t))$. Since the algorithm $T : Q^n \times \text{Cov}_n(Q) \rightrightarrows Q^n \times \text{Cov}_n(Q)$ is well-posed,

[1]The standard definition of fixed point for a set-valued map (which we do not use) uses the weaker condition $(c, P) \in T(c, P)$.

we have that $Q^n \times \mathrm{Cov}_n(Q)$ is strongly positively invariant. This fact implies that assumption (i) of Lemma 2.3.1 is satisfied.

We can form a Lyapunov function using Proposition 3.3.2 as follows. Since Q is a finite set, there exists only a finite number of possible values for $\mathcal{H}_{\max}$, $\mathcal{H}_{\min}$, and $|M|$. Let ϵ_x and ϵ_n be the magnitude of the smallest possible difference between two values of $\mathcal{H}_{\max}$ and $\mathcal{H}_{\min}$, respectively. Let α_n and α_M be larger than twice the maximum possible values of $\mathcal{H}_{\min}$ and $|M|$, respectively. Consider the following function $U : Q^n \times \mathrm{Cov}_n(Q) \to \mathbb{R}_{\geq 0}$:

$$U(c, P) = \mathcal{H}_{\max}(c, P) + \frac{\epsilon_x}{\alpha_n}\mathcal{H}_{\min}(c, P) + \frac{\epsilon_x}{\alpha_n}\frac{\epsilon_n}{\alpha_M}|M(P)|.$$

With this scaling of $\mathcal{H}_{\min}$ and $|M|$, when $\mathcal{H}_{\max}$ decreases then U necessarily also decreases, and similarly if $\mathcal{H}_{\max}$ is constant but $\mathcal{H}_{\min}$ decreases. Thus, invoking Proposition 3.3.2, we conclude that if $(c', P') \in T(c, P)$, then either $U(c', P') < U(c, P)$ or $(c', P') = (c, P)$. Thus, $U(c, P)$ fulfills assumption (ii). Finally, the communication model (C3) assures that assumption (iii) is met.

Hence, we are in the position to apply Lemma 2.3.1 and conclude the following result.

Proposition 3.3.4 (Convergence of $P(t)$) *The evolution of the One-to-Base Coverage Algorithm $(c(t), P(t))$, generated by the map T, converges in finite time to the intersection of the equilibria of the maps T_i, which is the set of pairs (c, P) where P is a centroidal Voronoi partition generated by c. In particular, $P(t)$ converges in finite time to one centroidal Voronoi partition.*

3.3.5 Convergence of Robot Covering

So far we have discussed the properties of the covering P held by the base station. Here we extend these arguments to the covering S held by the robots. First, we show that S is indeed a covering of Q.

Proposition 3.3.5 (Well-posedness of S) *Let S be a n-covering of Q. Then, S^+ produced by the* One-to-Base Coverage Algorithm *is also a n-covering.*

Proof. Let $s \in Q$. If there exist times $t_1 < t_2$ such that $q \in S_i(t_1)$ and $q \notin S_i(t_2)$, then there exists a $\tilde{t} \in [t_1, t_2)$ such that $q \notin P_i(\tilde{t}^+)$. By how the update of $P(t)$ is defined, this implies that some agent $j \neq i$ with $q \in P_j(\tilde{t})$ communicates to the base station at time $\tilde{t}$. But since $S_j(\tilde{t}^+) = P_j(\tilde{t})$, we have that $q \in S_j(\tilde{t}^+)$. Therefore, q must belong to some region of $S(t)$ for all t. ∎

We are now ready to conclude our convergence proof.

Proof. [Proof of Theorem 3.2.2]. The definition of the One-to-Base Coverage Algorithm implies that if there exists $\tau \in \mathbb{N}$ such that $P(t) = \bar{P} \in \mathrm{Cov}_n(Q)$ for $t \geq \tau$, then $S(t) = \bar{P}$ for $t \geq \tau + \Delta$. As an immediate consequence of this fact, the convergence properties of $P(t)$, stated in Proposition 3.3.4, are inherited by $S(t)$. ∎

3.4 Dynamic Changes to Team

One benefit of evolving overlapping coverings instead of strict partitions is the simplicity of handling dynamic arrivals, departures, and even the disappearance of robots. While departure or disappearance can increase $\mathcal{H}_{\max}$, such an increase is only a transient and the system will evolve towards a new centroidal Voronoi partition.

Arrival

When a new robot i communicates with the base station, it can be assigned any initial P_i desired. Possibilities include adding all vertices within a set distance of its initial position or assigning it just the single vertex which has the highest coverage cost in Q.

Departure

A robot i might announce to the base station that it is departing, perhaps to recharge its batteries or to perform some other task. In this situation, the base station can simply add P_i to the territory of the next robot it talks to before executing the normal steps of the algorithm.

Disappearance

The disappearance or failure of a robot i can be detected if it does not communicate with the base station for longer than Δ. If this occurs, then the departure procedure above can be triggered. Should i reappear later, it can be handled as a new arrival or given its old territory.

Remark 3.4.1 (Robustness to Changes to Team) *The* One-to-Base Coverage Algorithm *with the additions above is robust to the arrival or departure of*

robots from the team and after such an event will converge to a centroidal Voronoi partition in finite steps.

3.5 Numerical Results

To demonstrate the utility of the One-to-Base Coverage Algorithm, we implemented it using the open-source Player/Stage robot control system [39] and the Boost Graph Library (BGL) [40]. All results presented here were generated using Player 2.1.1, Stage 2.1.1, and BGL 1.34.1.

One illustrative example is shown in Figure 3.2. The environment contains three obstacles drawn in black and the four octagonal robots are tasked with providing coverage of the free space around the obstacles. This free space is modeled using an occupancy grid with a $0.6m$ resolution which was chosen so that the robots could fit inside of a grid cell. The grid is converted into a graph by making each free cell a vertex and connecting edges between cells which border each other. To compute distances in this uniform edge weight graph we extended the BGL breadth-first search routine with a distance recorder event visitor.

For this example we chose a random robot to communicate with the base station at each iteration, while ensuring that no robot went more than 8 rounds without being selected. In the intermediate state of covering P shown in the second panel of Figure 3.2, the light blue robot on the top left and the dark blue robot on the middle left both own some vertices also claimed by the circled orange robot. The third panel shows the result after the orange robot communicates with the base station: the orange robot's centroid has been updated and both blue robots have relinquished their claim to vertices closer to orange.

The final centroidal Voronoi partition in the fourth panel is reached after 25 iterations, with the evolution of $\mathcal{H}_{\max}$ over time shown in Figure 3.3. The final coverage cost of $1.82m$ represents an improvement of 59% over the starting cost. Since each robot initially covers the entire environment, this also represents the improvement from using 4 robots instead of 1 to provide coverage in this environment.

3.6 Summary

We have described the One-to-Base Coverage Algorithm which can drive territory ownership among a team of robots in a non-convex environment to a centroidal Voronoi partition in finite time given only occasional contact between each robot and a central base station. Here we have focused on dividing territory, but

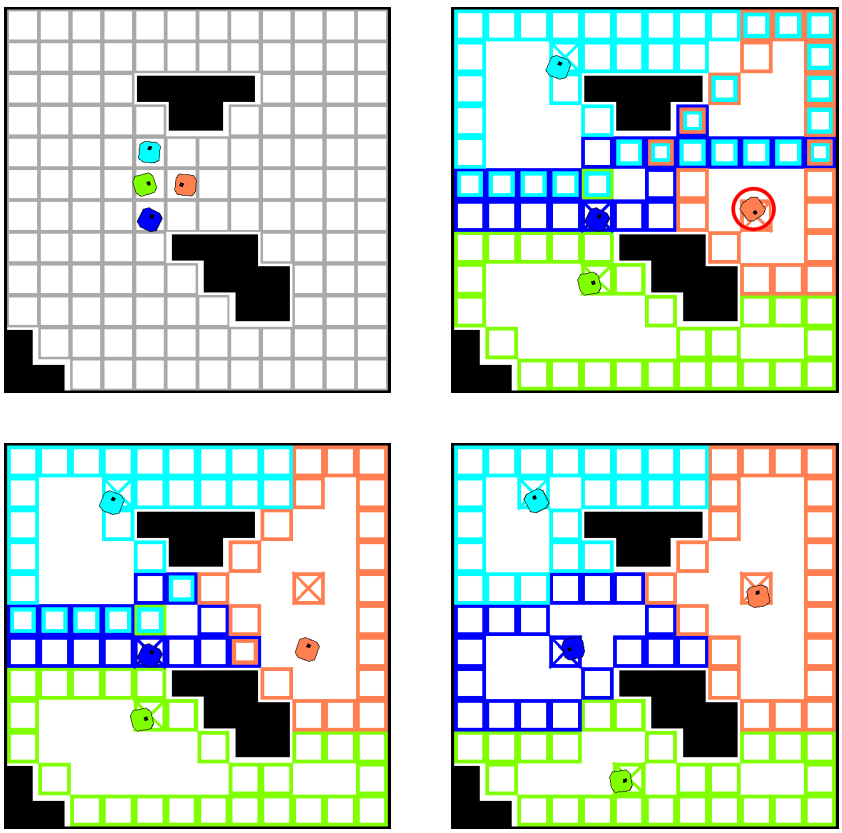


Figure 3.2: Simulation of four robots partitioning an environment with three black obstacles. The free space of the environment is modeled using the indicated occupancy grid where each cell is a vertex in the resulting graph. On the left, each robot starts owning the entire environment and positioned at its initial unique centroid. The middle frames show an intermediate state of the covering P and the result of an update when the circled robot contacts the base station. The boundary of each robot's territory drawn in its color with centroids marked with an X. The final partition is shown at right.

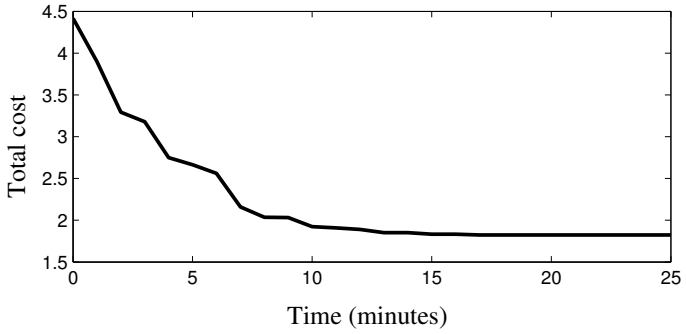


Figure 3.3: Graph of the cost $\mathcal{H}_{\max}$ over time for the simulation in Fig.

the algorithm can easily be combined with methods to provide a service over Q, as in [17].

In practical use, between the times that a robot communicates with the base station it could take sample measurements, pick up packages, or perform other tasks. When a robot communicates to the base station, it could transmit any information it has gathered about the environment and then receive its updated territory and a list of tasks to perform. When idle, a robot would position itself at the centroid of its territory. If tasks appear according to the distribution ϕ (which could evolve over time), then by minimizing cost function $\mathcal{H}_{\max}$ the algorithm also minimizes the the expected distance between a task and the furthest idle robot which might be assigned the task.

Chapter 4

Distributed Environment Clearing

This chapter deals with a distributed pursuit-evasion problem for a team of robotic searchers in an unknown environment. The particular *pursuit-evasion problem* we examine, also known as the *clearing problem*, involves designing control and communication protocols such that the searchers sweep an environment and detect any intruders which may be present. The clearing problem has received a lot of attention in recent years because of its applications to safety and security. In this chapter, we describe a distributed environment clearing algorithm based on the concept of the *frontier* or boundary between cleared and contaminated regions. Our algorithm can guarantee the detection of any intruders or, if there are insufficient searchers available, clear as much area as it can while ensuring no cleared area is recontaminated.

There are three key contributions of this work. First, our frontier-based clearing algorithm can guarantee detection of evaders in unknown, multiply-connected planar environments which may be non-polygonal. We introduce the (d, ϕ)-searcher model, a realistic model of current robot and sensor hardware with limited range and limited field-of-view sensing, and prove that our algorithm will clear an environment provided sufficient searchers are available.

Second, our clearing algorithm is distributed and efficient. We detail a novel method for storing and updating the global frontier between cleared and contaminated areas based on local intersections of oriented arcs. This method uses a small, constant amount of memory per robot and does not require a map or global localization. We also propose a viewpoint planning method which locally minimizes the number of robots required to rapidly expand the cleared area.

Third, we present both realistic simulations and hardware experiments to val-

idate our approach. We implemented the algorithm using the Multirobot Integration Platform and the Player/Stage robot simulation system. Our implementation demonstrates that frontiers and sensor footprints can be handled in a discetized fashion, that the algorithm is robust to sensor and motion noise, and that the local optimizations in our algorithm lead to efficient clearing of complex environments. We also present Monte Carlo results for the clearing efficacy of our algorithm for as a function of the number of robots.

In the literature, [27] is the only other work for multiple robots which can guarantee detection of evaders without prior knowledge of the environment. Their approach sweeps hallways using lines of robots where the robots at the ends perform wall following and give commands to those in the middle. There are a few important differences from our work. For one, they build and store a topological map whose memory footprint scales with the size of the environment, whereas our approach requires only a constant amount of memory per robot. Partly because of the global map, their approach sweeps only one hallway at a time and they admit there are complications in dealing with topological holes. Our method expands in parallel and handles holes seemlessly. Their dependence on wall following would also prove challenging in cluttered environments or in large empty spaces which our approach can handle. We also provide simulation and experimental validation which are not included in their theoretically-oriented work.

Section 4.1 provides definitions and states the problem we are addressing. In Section 4.2 we examine a centralized version of our algorithm to explain some details. The decentralized clearing algorithm is presented and illustrated in Section 4.3. In Section 4.4 we discuss theoretical properties of the algorithm and in Section 4.5 we present experimental results. We conclude in Section 4.6 and mention some future directions.

The work in this chapter was done in collaboration with A. Franchi.

4.1 Searcher Model & Problem Formulation

We are given a team of n robotic searchers with limited sensing and communication capabilities and finite memory. The searchers start clustered together in the free space of an unknown but limited planar environment. Let Q be the free space of the environment, which must be connected but can have holes and may be non-polygonal. The searchers are tasked with detecting evaders which may be arbitrarily small and can move arbitrarily fast, but continuously, through Q. The trajectories and initial positions of the evaders are unknown.

4.1.1 Robot and Sensor Models

The robot model we use, the $(d, \phi)-$*searcher*, is a differential or omnidirectional drive mobile robot that can rotate in place and translate continuously at bounded speed through Q. Our model gets its name from the attached distance sensor which has a maximum range $d > 0$ and an angular field-of-view $\phi \in [\pi, 2\pi]$. The sensor cannot penetrate obstacles but is capable of detecting any evaders visible to it. We will also discuss d-searchers, which are a (d, ϕ)-searchers with $\phi = 2\pi$.

Let S denote the *footprint* of the sensor when a robot is in a generic configuration, as shown in Fig. 4.1. The footprint is a local obstacle free region and we say that a point is *guarded* by a robot if it belongs to the footprint of the sensor of that robot. The oriented *boundary* of the sensor footprint, ∂S of S, is a closed

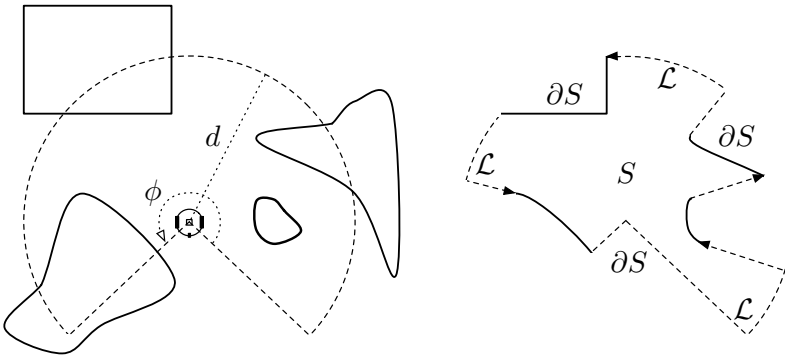


Figure 4.1: On the left, four obstacles surround a $(d, \phi)-$searcher and lie within the dashed circular sector representing the area perceivable by the searcher's sensor without occlusions. The right image shows the boundary ∂S of the sensor footprint for this configuration, with dashed oriented arcs for the free boundary $\mathcal{L}$ and solid arcs for the local obstacle boundary

arc partitioned into two sets: (1) the *local obstacle boundary* (all the points where the sensor has perceived an obstacle), and (2) the *free boundary*, denoted by $\mathcal{L}$, which consists of all the remaining points. Notice that while S is always a simply connected region, $\mathcal{L}$ is not, in general, a connected set. We refer to the connected subsets of $\mathcal{L}$ as *free arcs*. The *orientation* of ∂S is defined in a counter-clockwise manner, such that a point moving along the boundary would have the internal part of S on the left. The free arcs constituting $\mathcal{L}$ inherit the orientation of ∂S

and are an open subset of the topological manifold ∂S, with their endpoints on obstacles. The local obstacle boundary arcs, on the other hand, are closed in ∂S.

The *perception* of a searcher's sensor at a given pose is the tuple $\{S, \partial S, \mathcal{L}\}$, i.e., a footprint S, the boundary ∂S, and the set of free boundary arcs $\mathcal{L}$ of ∂S.

4.1.2 Communication, Localization, and Memory

Our method for classifying ∂S requires that a pair of robots are guaranteed to be able to communicate whenever their sensor footprints intersect. For example, this condition would be satisfied if robots can communicate when the distance between them is less than the sum of the radii of their sensor footprints.

One of the benefits of our approach is that it can work in the absence of global localization. Instead, we will assume that two robots with intersecting sensor footprints can compute their relative poses as a result of some mutual localization procedure. Mutual localization could be achieved by the method described in [46], or by scan matching [47]. Alternatively, the mutual visibility of the overlapping portion of footprints could be used by projecting calibration dots or dispatching an extra robot to serve as an intermediary. We further require that a robot is able to localize itself with respect to a perception whenever it is inside the footprint, e.g., by a simple pairwise scan matching.

Each searcher must have an amount of memory which is strictly sufficient to store two perceptions, plus some variables of negligible size used for the execution of the algorithm. This constraint means that each step of the distributed clearing algorithm must use only a limited and constant amount of memory per robot regardless of the size of Q.

4.1.3 Inspected Region and Problem Statement

For notation and explanation, we have use for the union of the perceptions taken by all robots from different poses in Q during algorithm execution, which we refer to as the *inspected region* and denote by I. Since our algorithm does not allow recontamination, I also represents the *cleared area*. Though I will be connected, it may not be simply connected, meaning that ∂I is a set of closed oriented curves. As with ∂S, ∂I is oriented and partitioned into two sets: (1) the *obstacle boundary*, and (2) the *frontier* denoted by $\mathcal{F}$. We wish to emphasize that our algorithm does not compute or store I, which is incompatible with the memory constraint, but instead uses only the oriented frontier $\mathcal{F}$.

With these definitions we can now state the goal of our algorithm: control a team of n (d, ϕ)-searchers so that they always guard all the points of the frontier

$\mathcal{F}$ while expanding the cleared region I as much as possible, subject to the limited sensing, communication, and memory constraints.

4.2 The Centralized Clearing Algorithm

For clarity, we have split the presentation of our clearing algorithm into two stages. In this Section we pretend that a central controller is commanding the searchers so that we can describe the fundamental algorithm steps and the data structures involved. In Section 4.3 we detail the distributed implementation of the algorithm.

At any given time, the team of n searchers is divided into two classes, the *frontier-guards* and the *followers*:

- *Frontier-guard*: Each frontier-guard is assigned a unique pose $v = (x, y, \theta) \in Q \times [0, 2\pi[$ called the guard's *viewpoint*, which can move during the evolution of the algorithm. The frontier-guard must quickly reach its viewpoint and report a perception $\{S, \partial S, \mathcal{L}\}$. To detect evaders, each frontier-guard must also continuously monitor its sensor.

- *Follower*: Each follower is assigned to passively follow a frontier-guard, and this assignment can change as the algorithm progresses.

As needed, the central process will switch frontier-guards to followers, and vice-versa. The steps of the *centralized clearing algorithm* are as follows.

Algorithm 4.1: Centralized Clearing Algorithm

Initialize one robot as frontier-guard, the rest as followers. The central process performs these actions:

1: **for** each $\{S_k, \partial S_k, \mathcal{L}_k\}$ received **do**
2: Compute $\mathcal{F}_k$ from $\mathcal{F}_{k-1}$ and $\{S_k, \partial S_k, \mathcal{L}_k\}$ as detailed in Sec. 4.2.1.
3: Compute the next set of viewpoints V_{k+1} as detailed in Sec. 4.2.2.
4: Assign each $v \in V_{k+1}$ to a nearby searcher and set the searcher to be a frontier-guard.
5: Assign remaining searchers a frontier-guard to follow.
6: Compute paths for all frontier-guards to reach their viewpoints while maintaining coverage of $\mathcal{F}_{k-1}$, and send the paths to the guards.

For the reader's convenience we describe the algorithm in detail. At the beginning, all n searchers are clustered around a point in Q. One robot is selected as the initial frontier-guard and assigned its initial pose as a starting viewpoint.

Table 4.1: Main symbols used in the algorithm.

Symb.	*Description*
Q	Planar environment.
$S(v)$	Sensor footprint from pose v.
S_k	Sensor footprint of the $k-$th perception.
∂S_k	Oriented boundary of S_k.
$\mathcal{L}_k$	Free (non-obstacle) boundary of ∂S_k.
I_k	Inspected region at the $k-$th step $:= \cup_{i=1}^{k} S_i$.
$\mathcal{F}_k$	Oriented frontier arcs of I_k. $\mathcal{F}_k = \mathcal{F}_{k-1}^{\text{Ext}} \cup \mathcal{L}_k^{\text{Ext}}$.
$\mathcal{F}_{k-1}^{\text{Ext}}$	$\mathcal{F}_{k-1} \backslash \text{closure}(S_k)$.
$\mathcal{L}_k^{\text{Ext}}$	$\mathcal{L}_k \backslash \text{interior}(I_{k-1})$.
V_k	Set of viewpoints at the $k-$th step.

All other robots are set as followers of this guard. The initial frontier-guard then records the first perception $\{S_1, \partial S_1, \mathcal{L}_1\}$, which initializes the main data stored during the evolution of the algorithm.

Whenever a frontier-guard arrives at its viewpoint and records a new perception, it sends the perception to a central processing unit. In this way the central process receives a sequence of perceptions. For each perception received, a new step k of the algorithm starts and the perception is classified as $\{S_k, \partial S_k, \mathcal{L}_k\}$ and called the k-th perception (refer to Table 4.1 for a reminder of the meaning of the symbols).

We denote the total inspected region at step k as $I_k := \cup_{i=1}^{k} S_i$. Again, the algorithm does not use or store I_k or the obstacle portion of ∂I_k; an important innovation of this work is that it stores and updates only $\mathcal{F}_k$, the oriented frontier arcs of I_k. Since the obstacle boundary of the inspected region I_k is impossible for either searchers or evaders to cross, there are only two ways an evader can enter I_k: (1) by being inside of $S_k \backslash I_{k-1}$ at the instant in which the k-th perception is performed, or (2) by crossing $\mathcal{F}_k$. In this first case detection of the evader is immediate, the focus of our algorithm is thus on maintaining complete coverage of $\mathcal{F}_k$ and updating it when a new perception is added.

The basic flow of the centralized clearing algorithm is as follows. After receiving a new perception the global frontier is updated (Step 2). Next the determination of a new set of viewpoints to cover and expand the frontier is performed (Step 3). After that, searchers are assigned roles as guards or followers and dispatched to their respective target positions (Step 4).

To guarantee the detection of any evaders in Q, the planning of new viewpoints

in step 2 must meet the *frontier guarding property* and the *expansion property* laid out in the following Definition. We describe our method which achieves these properties in Section 4.2.2.

Definition 4.2.1 (Viewpoint Planning Properties) *Given a non-empty frontier $\mathcal{F}_k$ and a set of prior viewpoints V_k, the viewpoint planner selects the smallest set of viewpoints V_{k+1} inside I_k which satisfy the following two properties:*

(i) Frontier guarding: Ensure $\mathcal{F}_k$ is contained in the closure of $\cup_{v \in V_{k+1}} S(v)$, and

(ii) Expansion: Ensure $Area(I_{k+1}) \geq Area(I_k) + \epsilon$ for some $\epsilon > 0$ except for at most finite steps.

Within these constraints, the viewpoint planner maximizes $Area(I_{k+1})$ assuming that there will be no new obstacles discovered.

We can now state the main theoretical result of this chapter.

Theorem 4.2.2 (Detection of Evaders) *Given an implementation of the* centralized clearing algorithm *with the viewpoint planning properties in Definition 4.2.1 and a number of robots $n \geq \max\{|V_k| \mid \text{for all steps } k\}$, the entire environment Q is cleared and every evader in Q is detected in finite time.*

Proof. The expansion property in Defintion 4.2.1 ensures that there will be a time step k_f where $\mathcal{F}_{k_f} = \emptyset$, meaning that ∂I_{k_f} consists entirely of obstacle arcs and I_{k_f} completely covers Q. Therefore, for every evader e in Q, there exists at least one instant of time when e either (1) is inside of $S_{k_e} \backslash I_{k_e-1}$ at time $k_e \in \{1, \ldots, k_f\}$, or (2) crosses $\mathcal{F}_{k_{e-1}}$ during the time interval $[k_{e-1}, k_e]$ for $k_e \in \{2, \ldots, k_f\}$. In the first scenario, detection of the evader is immediate. We can conclude, by means of the frontier guarding property, that the second scenario will also be detected. ∎

In the rest of this Section we describe how to implement the frontier update and how to plan viewpoints (Steps 2 and 3 of the centralized clearing algorithm, respectively). The path-planning in Step 4.1 is also non-trivial, however we will only discuss how to perform this in the context of the distributed version of the clearing algorithm in Section 4.3.

4.2.1 Global Frontier without a Global Map

On the first iteration, frontier $\mathcal{F}_1$ is initialized as the free boundary of the first perception, $\mathcal{L}_1$. For each step $k > 1$, the algorithm needs to compute the new frontier $\mathcal{F}_k$, i.e., the non-obstacle boundary of the inspected region $I_k = I_{k-1} \cup S_k$. The set $\mathcal{F}_k$ can be partitioned into two subsets, (1) the set $\mathcal{F}_{k-1}^{\text{Ext}}$ of arcs from $\mathcal{F}_{k-1}$ which do not belong to the closure of S_k, and (2) the set $\mathcal{L}_k^{\text{Ext}}$ of arcs from $\mathcal{L}_k$ which are not on the interior of $I_{k-1} = \cup_{i=1}^{k-1} S_i$. While the computation of $\mathcal{F}_{k-1}^{\text{Ext}}$ from $\mathcal{F}_{k-1}$ and $\{S_k, \partial S_k, \mathcal{L}_k\}$ is immediate, in this section we describe a novel method for computing $\mathcal{L}_k^{\text{Ext}}$ using only the oriented arcs of $\mathcal{F}_{k-1}$ and $\{S_k, \partial S_k, \mathcal{L}_k\}$.

In all previous work including [36], $\mathcal{L}_k^{\text{Ext}}$ has been computed using S_k and I_{k-1}. The disadvantages of this prior procedure for updating the frontier are that computing I_{k-1} requires global localization and storing it requires an amount of memory proportional to the area of environment Q, which is in contrast with the problem statement in Sec. 4.1.3. It is also worth noting that at step k it is not possible in general to compute $\mathcal{L}_k^{\text{Ext}}$ using only the most recent sensor footprints from each frontier-guard, see the example in Fig. 4.2. The orientation of $\mathcal{F}_{k-1}$ and ∂S_k is critically important for properly determining the frontier without I_{k-1}.

Our *global frontier update method* for computing $\mathcal{L}_k^{\text{Ext}}$ is based around the intersections of the oriented arcs of $\mathcal{F}_{k-1}$ and ∂S_k. Let $\mathcal{L}_k^\star$ denote the set of points belonging to the intersection between the arcs of $\mathcal{L}_k$ and the arcs of $\mathcal{F}_{k-1}$, and $\bar{\mathcal{L}}_k^\star$ the remaining points of $\mathcal{L}_k$. The actions of the global frontier update method are defined as follows.

Algorithm 4.2: Global Frontier Update Method

1: Classify the neighborhood of each $p \in \mathcal{L}_k^\star$ as internal or not
2: Classify the ends of each $\ell \in \mathcal{L}_k$
3: Propagate classification to rest of $\mathcal{L}_k$
4: Set $\mathcal{F}_k = \mathcal{F}_{k-1}^{\text{Ext}} \cup \mathcal{L}_k^{\text{Ext}}$

The points of $\mathcal{L}_k^{\text{Ext}}$ can be either on the boundary of or exterior to I_{k-1}, the boundary points belong to $\mathcal{L}_k^\star$ while the exterior ones belong to $\bar{\mathcal{L}}_k^\star$. The following crucial result states that an arc in $\mathcal{L}_k$ can only switch between the interior and exterior of I_{k-1} at an intersection point in $\mathcal{L}_k^\star$.

Lemma 4.2.3 (Neighborhood Classification) *Let ℓ be an arc in Q which does not intersect $\mathcal{F}_{k-1}$. If any point of ℓ belongs to the exterior of I_{k-1}, then all of ℓ belongs to the exterior. If any point of ℓ belongs to the interior of I_{k-1}, then all of ℓ belongs to the interior.*

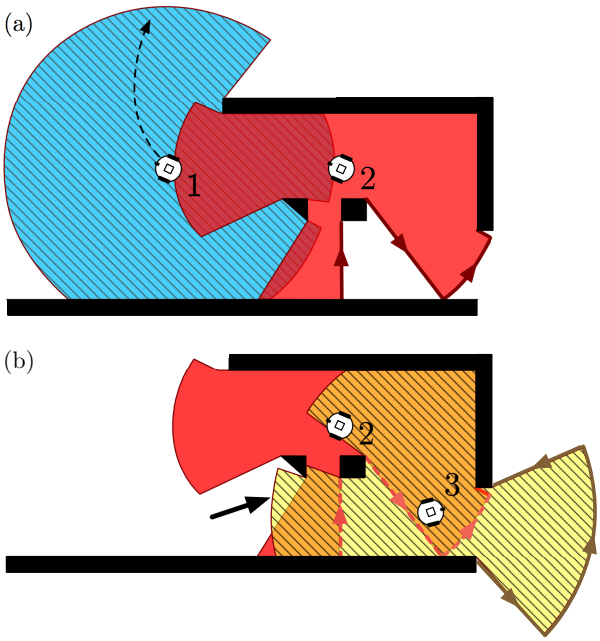


Figure 4.2: (a) After robots 1 and 2 have classified their frontiers, robot 1 moves to a new position. Once robot 1 has moved and recorded a new perception, its prior perception is no longer stored by the robot team. (b) When robot 3 arrives and records $\{S_k, \partial S_k, \mathcal{L}_k\}$ (striped yellow), it cannot properly classify $\mathcal{L}_k$ based only on the most recent perceptions of the other robots. Without all of I_{k-1}, robot 3 can only determine that the indicated section of $\mathcal{L}_k$ is not on the global frontier using the intersections of ∂S_k and robot 2's oriented frontier segments (dashed red)

Proof. Since ℓ is in Q, it cannot cross the obstacle boundary of ∂I_{k-1}. Therefore, if ℓ does not intersect $\mathcal{F}_{k-1}$, then it does not cross ∂I_{k-1}. ∎

The first step of the frontier update method is to classify the neighborhood on ∂S_k of each intersection point $p \in \mathcal{L}_k^\star$ as either internal to I_{k-1} or not. An example of this neighborhood classification is shown in Fig. 4.3. The neighborhood classifications for all possible intersection cases are depicted in Fig. 4.4.

The second step of the method is to classify the ends of each arc $\ell \in \mathcal{L}_k$ in the neighborhood of the endpoints of the adjacent obstacle arcs. These neighborhoods can be classified using the following Lemma.

Lemma 4.2.4 (Obstacle Arc Classification) *Let o denote a local obstacle arc of ∂S_k, let ℓ_L and $\ell_R \in \tilde{\mathcal{L}}_k^\star$ denote the ends of the free arc segments on the left*

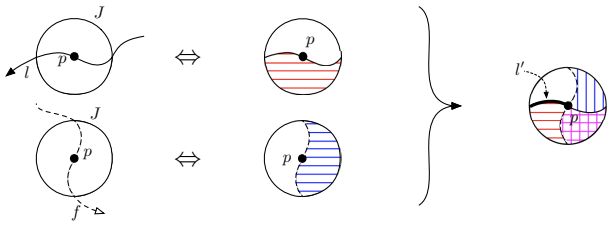


Figure 4.3: Classification of the neighborhood J of $p \in \mathcal{L}_k^\star$ where arcs $\ell \in \mathcal{L}_k$ and $f \in \mathcal{F}_{k-1}$ intersect. At left, the partitions of J induced by ℓ and f are shown separately. The white region on the right of the oriented arcs indicates the exterior and the patterned region indicates the interior. The fusion of the two partitions of J is shown at right. The bold part of ℓ, denoted by ℓ', belongs to $\mathcal{L}_k^{\text{Ext}}$ because it lies between a white and a patterned region. Note that in this case $p \in \ell'$

and right of o, respectively, in the neighborhood of the endpoints of o. Let $E_o \subset o$ be the set of endpoints of any frontier arcs of $\mathcal{F}_{k-1}$ which either begin or end on o, and which are, in the neighborhood of o, fully contained in the closure of S_k. Then:

- *If $E_o = \emptyset$, then either ℓ_L and ℓ_R are both internal to I_{k-1} or neither are.*
- *If $E_o \neq \emptyset$, then ℓ_L is internal to I_{k-1} if the closest[1] $e \in E_o$ is the beginning of a frontier arc, and not internal otherwise. The opposite holds for ℓ_R.*

Proof. If $E_o = \emptyset$, as shown in the first two cases of Fig. 4.5, then there exists a free arc connecting ℓ_{L} with ℓ_{R} which is contained in the interior of S_k and is close enough to o to not intersect $\mathcal{F}_{k-1}$. Therefore, we can apply Lemma 4.2.3. If $E_o \neq \emptyset$, assume without loss of generality that it is a singleton, i.e., $E_o = \{e\}$, as shown in the third and fourth cases of Fig. 4.5. Then, there exists a free arc connecting ℓ_{L} to the 'nearest' half of the neighborhood of e which is in the interior of S_k and is close enough to o to not intersect $\mathcal{F}_{k-1}$. Therefore, we can apply Lemma 4.2.3. Similar claims hold for ℓ_{R}. ∎

The third and final step is to propagate the classification from the neighborhoods to all points of the arcs of $\mathcal{L}_k$. This propagation again exploits Lemma 4.2.3. Notice that, so long as the selection of viewpoints guarantees that either $\mathcal{L}_k^\star \neq \emptyset$ or at least one local obstacle arc o has a non-empty E_o, this third step is well defined.

Combined, these three steps determine which segments $\mathcal{L}_k$ are not in the interior of I_{k-1} and thus should be included in frontier $\mathcal{F}_k$.

[1] *With respect to the distance on the arc o.*

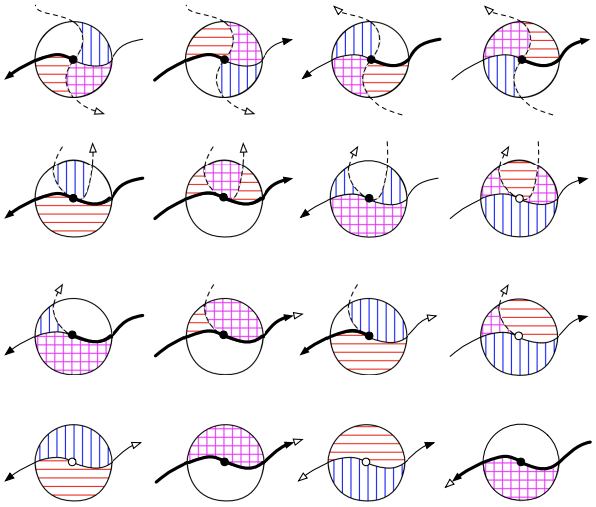


Figure 4.4: The classification of the points of arc $\ell \in \mathcal{L}_k$ in the neighborhood of all possible types of intersections with arc $f \in \mathcal{F}_{k-1}$. Arc ℓ is drawn solid, while f is dashed. Each row shows a different intersection type, with columns for the various reciprocal orientations of ℓ and f. The first row shows isolated crossings, the second shows isolated tangents, the third shows joinings, and the fourth row shows segments where ℓ and f overlap. The bold portions of ℓ belong to $\mathcal{L}_k^{\text{Ext}}$

4.2.2 Viewpoint Planning

In this Section we describe how to pick a set of viewpoints V_{k+1} which meet the *frontier guarding property* and *expansion property* of Definition 4.2.1. With the distributed application in mind, we simplify the planning of V_{k+1} by constructing it from V_k. Let v_k be the viewpoint of the k-th perception. As detailed in Section 4.2.1, $\mathcal{F}_k$ can be partitioned into two sets: $\mathcal{F}_{k-1}^{\text{Ext}}$ (a subset of the prior frontier), and $\mathcal{L}_k^{\text{Ext}}$ (a subset of ∂S_k). Let $\mathcal{F}_{k-1}^{\text{Int}}$ be the portion of $\mathcal{F}_{k-1}$ which is inside the closure of S_k.

To construct V_{k+1}, we need the following sets:

(i) The set of viewpoints $V_k^{\text{obs}} \subset V_k$ which are assigned to guard only obsolete portions of the frontier in $\mathcal{F}_{k-1}^{\text{Int}}$, if any.

(ii) A set of new viewpoints V' inside S_k to cover and expand the new frontier segments $\mathcal{L}_k^{\text{Ext}}$.

With these defined, we then set:

$$V_{k+1} = \left((V_k \setminus v_k) \setminus V_k^{\text{obs}}\right) \cup V'.$$

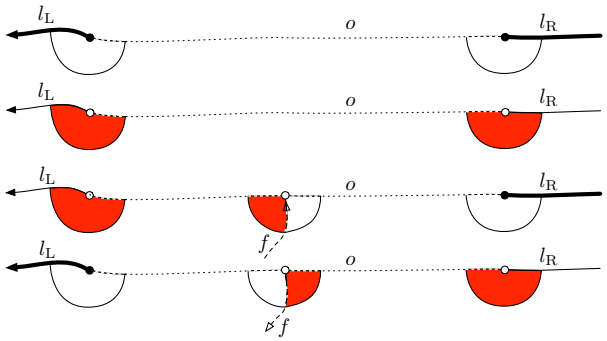


Figure 4.5: Four classification cases are depicted for an obstacle arc o (dotted) with two adjacent free arcs (solid). In the first two, no internal frontier arc has an endpoint on o, so in the neighborhood of o the free arcs are classified as both frontier (bold) or both internal (thin). In the second two cases, an internal frontier arc f (dashed) has an endpoint on o which induces opposite classifications for the two free arcs

The rest of this section is devoted to describing how to choose V' inside of S_k when $\mathcal{L}_k^{\text{Ext}} \neq \emptyset$.

We say that a free arc $\ell \in \mathcal{L}_k$ is *relevant* for viewpoint planning if it contains a frontier fragment from $\mathcal{L}_k^{\text{Ext}}$. A relevant free arc may contain one or more frontier fragments, and each frontier fragment is entirely contained in one relevant free arc. Let $\mathcal{L}_k^{\text{Rel}} \subseteq \mathcal{L}_k$ denote the set of relevant free arcs around v_k.

Our *local viewpoint planning method* consists of partitioning the frontier fragments of each $\ell_{\text{Rel}} \in \mathcal{L}_k^{\text{Rel}}$ among the fewest possible new viewpoints. We first detail how to perform the method for d-searchers, that is, robots with a sensor with a field-of-view of 2π. Afterwards, we describe how to adapt the method for (d, ϕ)-searchers. In both cases, the actions of the local viewpoint planning method are as follows.

Algorithm 4.3: Local Viewpoint Planning Method

Initialize $V' = \emptyset$. Then, for each $\ell_{\text{Rel}} \in \mathcal{L}_k^{\text{Rel}}$ perform the following:

1: Determine p, the number of viewpoints needed to cover ℓ_{Rel}
2: Partition ℓ_{Rel} into p pieces
3: **for** $i = 1$ to p **do**
4: Select a pose v in S_k to cover the i-th partition of ℓ_{Rel} and as much new area as possible
5: Add v to V'

Remark 4.2.5 *This viewpoint planner is for circular sector footprints of radius d and field-of-view $\phi \geq \pi$. For more general footprints, our clearing algorithm could also be applied provided a viewpoint planning method with the properties in Definition 4.2.1 is available.*

Each ℓ_{Rel} is comprised of straight radial segments and circular segments with radius d. The possible configurations are: single radial; single curved; curved with radial on one side; or curved with radial segments on both sides (see the examples in Fig. 4.1 and Fig. 4.6). Let $S(v)$ denote the sensor footprint for a robot at viewpoint v. The following Lemma simplifies the determination of when a radial segment is inside $S(v)$ for $\phi = 2\pi$.

Lemma 4.2.6 (Coverage of Radial Arcs) *Let v' be a potential new viewpoint inside S_k for a d-searcher, and let $r \in \mathcal{L}_k^{Rel}$ be a radial free arc segment. Let p be the far endpoint of r and $\overline{v'p}$ be the line segment between v' and p. If $\text{dist}(v', p) < d$ and $\overline{v'p}$ only intersects ∂S at p, then open set r is contained inside of $S(v')$.*

Proof. Our proof centers around the triangle T formed by v_k, v', and p. Radial free arc segment r is a connected subset of $\overline{v_k p}$. Since S_k has maximum radius d, $\text{dist}(v', v_k) < d$. Combined with the fact that $\text{dist}(v', p) < d$, we can conclude that all of r is within d of v'. All that remains is to show that there are no obstructing obstacles inside of triangle T.

We know that $\overline{v'p}$ is contained in the closure of S_k because it only intersects ∂S at p. Since S_k is star-shaped, both $\overline{v_k p}$ and $\overline{v_k v'}$ are also contained in the closure of S_k. Then, as S_k is simply connected, we can conclude that the interior of T is in Q and, therefore, r is inside of $S(v')$. ∎

There are two notable consequences of Lemma 4.2.6. First, for any ℓ_{Rel} with only a radial segment, one viewpoint is sufficient. Second, for any ℓ_{Rel} which contains both curved and radial segments, we only need to partition the curved segment: the viewpoint which covers an endpoint of the curved segment will also cover any attached radial segment.

To assist in selecting viewpoints to cover curved segments, we introduce parameter $d_{\min} \in (0, d]$, the minimum distance between v_k and any $v \in V'$. As will become clear, $d_{\min}$ encodes a trade-off in the algorithm: smaller values of $d_{\min}$ can reduce $|V'|$ and thereby reduce the number of searchers required; larger values of $d_{\min}$ can increase the area exposed and thereby reduce the number of iterations required to clear Q.

Let $\delta(\ell_{\text{Rel}})$ be the angular width of ℓ_{Rel} measured counter-clockwise from the right-most frontier point on ℓ_{Rel} to the left-most frontier point on ℓ_{Rel}. A single

viewpoint at least $d_{\min}$ from v_k can then cover an angular width of at most $\alpha(d_{\min})$ given by

$$\alpha(d_{\min}) = 2\arccos(d_{\min}/2d) \in \left[\tfrac{2\pi}{3}, \pi\right).$$

The number of viewpoints η necessary to cover ℓ_{Rel} is then determined by the following Lemma.

Lemma 4.2.7 (Number of Viewpoints Required) *For any $\ell_{Rel} \in \mathcal{L}_k^{Rel}$, the clearing algorithm requires $\eta \in \{1, 2, 3\}$ viewpoints. Moreover:*

- *if $\delta(\ell_{Rel}) \leq \frac{2\pi}{3}$, then $\eta = 1$,*
- *if $\frac{2\pi}{3} < \delta(\ell_{Rel}) < \pi$, then $\eta = 1$ or 2,*
- *if $\pi \leq \delta(\ell_{Rel}) < 2\pi$, then $\eta = 2$ or 3, and*
- *if $\delta(\ell_{Rel}) = 2\pi$, then $\eta = 3$.*

Proof. This result is a direct consequence of Lemma 4.2.6 and the fact that $\alpha(d_{\min}) \in \left[\frac{2\pi}{3}, \pi\right)$. ∎

For $\eta > 1$, the angular width of ℓ_{Rel} is then partitioned such that the first viewpoint covers $[0, \delta(\ell_{\text{Rel}})/\eta]$, and each subsequent viewpoint covers the next equally sized slice of angular width. This partitioning of ℓ_{Rel} achieves step 2 of the viewpoint planning method.

After partitioning ℓ_{Rel}, the final step is to place each new viewpoint v. This placement must ensure that a perception from v covers the required portion of ℓ_{Rel} and also uncovers as much area as possible beyond ℓ_{Rel} (assuming no new obstacles). For single radial segments, we place v at the midpoint of the segment facing perpendicular to the segment out into the unknown territory beyond ℓ_{Rel}. For all other configurations, we construct a line through v_k which bisects the curved arc in ℓ_{Rel} assigned to v. We then place v on this bisector at the point in S_k which is closest to the intersection with ℓ_{Rel} and also ensures that both endpoints of the curved arc in ℓ_{Rel} assigned to v will be inside $S(v)$. Here pose v is oriented radially outward from v_k.

By construction, this method of selecting V' guarantees that $\mathcal{L}_k^{\text{Ext}} \in \cup_{v \in V'} S(v)$ for searchers with $\phi = 2\pi$, meaning that the *frontier guarding property* in Definition 4.2.1 is satisfied. For $d_{\min}$ close to zero, it creates the fewest new viewpoints possible, while for $d_{\min} = d$ it exposes more area with minimal additional viewpoints. The following Lemma shows that this viewpoint planner also guarantees the *expansion property*.

Lemma 4.2.8 (Guaranteed Expansion) *The set of new viewpoints V' produced by the Local Viewpoint Planning Method satisfies the expansion property from Definition 4.2.1.*

Proof. Consider a new viewpoint $v \in V'$ inside of S_k and the associated footprint $S(v)$. What we will show is that, except for at most a finite number of steps, there exists some new area $A \in S(v)$ where $\text{Area}(A) \geq \epsilon$ and $\text{Area}(A \cap I_k) = 0$ for some small $\epsilon > 0$.

Two properties allow us to determine values for epsilon. We have specified that v will be at least $d_{\min}$ from v_k. Let ℓ_v be the free arc segment assigned to viewpoint v and let r_e be the smallest possible diameter of any evader the team will be asked to detect. Then, we know that the length of ℓ_v is at least r_e or it can be ignored. In the case where ℓ_v is either a radial or curved segment, then a value for ϵ assuming that $d_{\min}$ and r_e are small is $d_{\min} r_e$. The case were ℓ_v is mixed is more intricate, but a similar lower bound can be found.

Finally, the only circumstances in which $\text{Area}(A)$ might be less than ϵ occur when either there is a finite-sized obstacle or a finite-length portion of $\mathcal{F}$ which reduces the size of A. These can only occur a finite number of times, so the statement holds. ∎

We have described a viewpoint planning method which meets the requirements of Definition 4.2.1 for searchers with $\phi = 2\pi$. For (d, ϕ)-searchers whose sensors have a field-of-view in$[\pi, 2\pi)$, the above method is optimal when ℓ_{Rel} contains either only radial frontier fragments, or only curved frontier fragments. One option for handling mixed fragments is to split them and handle the radial and curved parts separately. However, this simple approach may create more viewpoints than strictly required. We instead propose the following geometric method.

Consider the case when ℓ_{Rel} consists of a radial segment on the right of a curved segment. Let p_r be the first frontier point in the radial part of ℓ_{Rel}, and let p_m be the intersection of the curved and radial segments. Next, loop over possible p_l's, starting from $p_l = p_m$ and moving along the curved segment, stopping at the furthest p_l for which the midpoint of $\overline{p_r p_l}$ is within d of p_m. Then, place v at the midpoint of $\overline{p_r p_l}$, facing outward perpendicular to $\overline{p_r p_l}$. This placement ensures that all frontier points between p_r and p_l are covered by a perception taken from v for $\phi \geq \pi$, while maximizing the amount of ℓ_{Rel} covered. If any frontier on the curved segment remains uncovered, it can be handled using the prior approach. This method can be trivially modified if the radial segment is on the left, and can also be applied on both sides for a curved segment with radial segments on both sides.

4.3 The Distributed Clearing Algorithm

In the distributed setting, the communication graph is in general disconnected, necessitating some changes from the centralized description. First, the global frontier must be stored and updated in a distributed manner. Second, viewpoint planning must be performed locally by the frontier-guards. Furthermore, the distributed algorithm only has access to pairwise relative mutual localization between neighbors. Finally, while the centralized version is synchronous and sequential, the distributed setting is asynchronous and concurrent, i.e., it is possible for perceptions from disconnected searchers to be recorded at the same time.

4.3.1 Distributed Handling of Global Frontier and Viewpoint Planning

We distribute the storing and updating of the global frontier by having each frontier-guard store its local frontier segments and update them through communication with its neighboring frontier-guards. We denote the section of the global frontier $\mathcal{F}_k$ owned by robot i by $\mathcal{F}_{k,i}$. This distributed storage of the global frontier can always be achieved since, by the frontier guarding property, each global frontier point is guarded by a frontier-guard.

The *pairwise frontier update method* which follows is a distributed version of the method in Sec. 4.2.1 for classifying the free boundary $\mathcal{L}_k$.

Algorithm 4.4: Pairwise Frontier Update Method

When robot i records a new perception, it updates $\mathcal{F}_k$ as follows:

1: Classify neighborhood of each intersection p between $\mathcal{L}_k$ and $\mathcal{F}_{k-1,i}$ as internal or not, if any
2: **for** each robot j in communication with i **do**
3: Classify neighborhood of each intersection p between $\mathcal{L}_k$ and $\mathcal{F}_{k-1,j}$ as internal or not
4: Inform j if any piece of $\mathcal{F}_{k-1,j}$ lies inside S_k
5: Classify the ends of each $\ell \in \mathcal{L}_k$
6: Propagate classification to rest of $\mathcal{L}_k$
7: Store $\mathcal{F}_{k,i}$

This distributed frontier classification is always possible because the classification of $\mathcal{L}_k$ requires only the frontier fragments from $\mathcal{F}_{k-1}$ which intersect S_k. In the distributed setting, each of these frontier fragments belong either to a neighboring guard's perception or to robot i's previous perception. The localization

with respect to the first kind of fragments is guaranteed since by assumption two robots whose footprints intersect are in communication and are mutually localized. The localization w.r.t. the second kind of fragments is also guaranteed by assumption since robot i's current viewpoint lies in the footprint of the previous one.

Using the pairwise frontier update method, updates to the global frontier are based only on current relative poses of nearby searchers, not on absolute poses. The distributed clearing algorithm, therefore, is able to continue clearing an environment even if the searchers cannot determine where they are relative to where they started. Note that, because it operates in pairs, this frontier update method requires only an amount of memory per robot proportional to that required to store two perceptions.

Once $\mathcal{F}_{k,i}$ is determined, we can use the local viewpoint planning method from Sec. 4.2.2. This method is already distributed as it requires only the local frontier of the frontier-guard doing the planning. The execution of the path to new viewpoints can also be done without global localization, whether it's being executed by the guard itself or by a follower. Since the new viewpoint lies inside the local perception, either local odometry of reasonable accuracy or a registration of footprints taken along the path with S_k will suffice.

4.3.2 Distributed Algorithm & Robot Roles

The two classes of searchers from the centralized algorithm are each split in two, yielding four possible states:

- *Expand*: When a searcher is assigned a new viewpoint to move to, it enters the expand state until it reaches the viewpoint and records a perception.
- *Frontier-guard*: Each frontier-guard i remains stationary at its viewpoint and has complete control over its local frontier segments, $\mathcal{F}_{k,i}$. It must communicate with neighboring frontier-guards to update $\mathcal{F}_{k,i}$, plan a new viewpoint to cover and expand $\mathcal{F}_{k,i}$, and dispatch a follower to the new viewpoint.
- *Follow*: Must passively follow and respond to commands from a frontier-guard or expander.
- *Wander*: When a frontier-guard no longer has a local frontier to guard, it wanders until it locates a leader to follow.

Algorithm 4.5: Distributed Clearing Algorithm

To begin, one searcher is set to Expand to its starting pose and all others start either Following the first or in the Wander state. All agent's then continuously execute the procedure corresponding to their state.

Procedure `Expand`

```
Data: frontier,path
foreach follower in followers do
    Send(follower, "follow",path);
Move(path);
{S, ∂S, L} ← Perceive();
neighFront ← UpdateNeighFrontier();
frontier ← Frontier({S, ∂S, L},frontier,neighFront);
DoBehavior("Frontier-Guard",S,frontier);
```

Procedure `Follow`

```
Receive(Leader,message,path);
switch message do
    case "follow"
        Move(path);
    case "expand"
        DoBehavior("Expand",∅,path);
    case "wander"
        DoBehavior("Wander");
```

Procedure `Frontier-Guard`

```
   Data: S,frontier
 1 if frontier is empty then
 2     Send(followers, "wander");
 3     DoBehavior("Wander");
 4 (bestVP,NumVPs) ← ViewPointPlan(S,frontier);
 5 path ← PathToViewPoint(S,bestVP);
 6 if NumVPs == 1 then
 7     DoBehavior("Expand",frontier,path);
 8 else
 9     if followers has at least one follower then
10         follower ← PopFollower(followers);
11         Send(follower, "expand",path);
12         while follower is expanding do
13             Sleep();
14     else
15         while no new neighbor and no followers do
16             Sleep();
17     DoBehavior("Frontier-Guard",S,frontier);
```

Procedure `Wander`

```
1 SearchForLeader();
2 if leader found then
3     DoBehavior("Follow");
4 if all searchers wandering then
5     exit
```

The *distributed clearing algorithm* consists of an initialization step, followed by each searcher iteratively executing the procedure corresponding to its current state. These procedures have subroutines for all important computations, and detail when searchers transition between states. There are four key subroutines we would like to highlight:

- `UpdateNeighFrontier/Frontier`: These two functions perform the pairwise frontier update method in Section 4.3.1.
- `ViewPointPlan`: This function follows the local viewpoint planning method

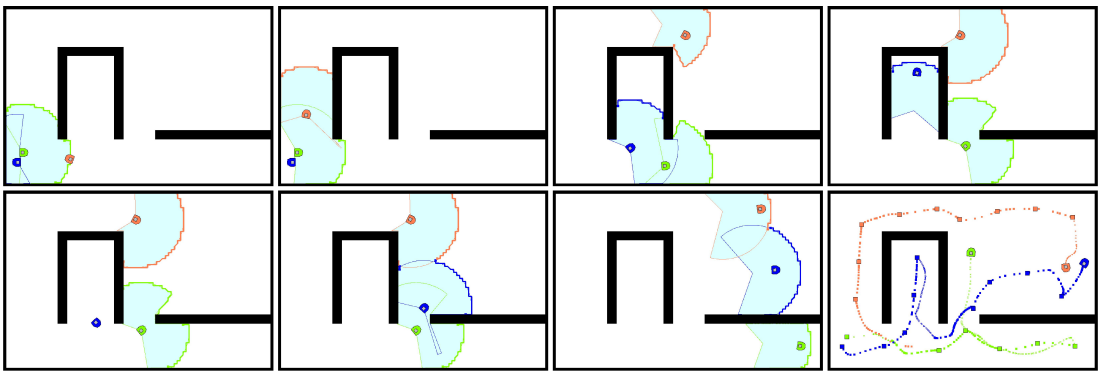


Figure 4.6: Simulation of three (d, ϕ)-searchers clearing an environment. Recorded perceptions are shown in a light blue, with frontiers shown with bold lines in the color of the frontier guard who owns them. The trajectories of the robots are shown in the final panel, with large squares for viewpoints

in Section 4.2.2, and then picks the best new viewpoint to expand first.

- `PathToViewPoint`: Determines a safe path from the current viewpoint to the new viewpoint inside S, which can be a straight line since S is star-shaped.
- `SearchForLeader`: This function does a random walk with two additional behaviors: 1) when it encounters a frontier-guard, it switches to Follow; 2) wanderers may join to form a wandering blob.

The following subsections describe our implementation of the algorithm, show simulation and hardware experiment results, and expand on some technical details.

4.3.3 Illustrative Simulation

Figure 4.6 provides a detailed example of three robots implementing the distributed algorithm. The searchers are simulated Khepera III robots with laser rangefinders with a range of 0.8 m and a field-of-view of 240°. Perfect mutual localization is provided for this simulation, while the Smooth Nearness Diagram navigation driver in Player is used to navigate between viewpoints and avoid collisions [42].

The first panel of Fig. 4.6 shows the initialization of the algorithm. The three robots start within communication range of each other and with initial poses which do not significantly interfere with each other's sensors. The green robot begins as a frontier-guard and records the first perception. The blue robot then clears the area behind the green robot.

In the second panel, the orange robot has expanded one of the initial frontiers, classified its boundary, and become a frontier-guard. As orange only needs one new viewpoint to expand its single frontier arc, it will expand alone around the top of the obstacle. Blue is then dispatched to clear the other initial frontier.

The next two images show the continued expansion of the cleared area. By the fourth panel, both orange and green have reached positions from which they require assistance in order to expand. After the blue robot clears the inside of the U-shaped obstacle, it enters the Wander state and searches for a leader.

The remaining images show the final stages of the algorithm, where orange and blue clear one room while green clears the lower corridor. Green finishes before the others, and enters the Wander state to try to find them. The final panel shows recorded trajectories for the robots during clearing as well as all viewpoints.

4.4 Theoretical Analysis & Results

4.4.1 Frontier Guarding & Expansion Properties

The behavior of the frontier-guards in the distributed clearing algorithm guarantees both the frontier guarding property and the expansion property from Definition 4.2.1. When expander i reaches its viewpoint and makes a perception, it then enters the stationary frontier-guard state. So long as i remains a frontier-guard, it maintains complete coverage of the frontier segments in $\mathcal{F}_{k,i}$. Searcher i will only leave the frontier-guard state if either $\mathcal{F}_{k,i}$ is erased by a new neighbor, or if i determines that one new viewpoint is sufficient to cover $\mathcal{F}_{k,i}$ and that the path to the viewpoint also maintains coverage of $\mathcal{F}_{k,i}$. The local viewpoint planning method guarantees that each new viewpoint will expand the cleared area.

4.4.2 Algorithm Completeness

Two assumptions are required to extend Theorem 4.2.2 and claim that the distributed clearing algorithm is guaranteed to detect every evader in Q. First, there must be a sufficient number of searchers available to expand at least one frontier segment at each step of the algorithm. Second, any searcher who enters the Wander state must reach an active frontier in finite time. When these two assumptions are satisfied, the searchers will never have to wait an infinitely long time between the recording of perceptions. Therefore, we can conclude following the proof of Theorem 4.2.2 and the frontier guarding and expansion properties that the distributed clearing algorithm will clear all of Q and detect every evader in Q.

4.4.3 Time and Memory Complexity

The computational requirements of the four main subroutines of the distributed clearing algorithm are as follows. An important innovation of this works is the pairwise frontier merging method, which requires only $\mathcal{O}(2|\partial S|)$ memory and $\mathcal{O}(|\partial S|^2)$ time to find intersections and classify the local frontier (where ∂S is the boundary of one sensor footprint). Our geometric viewpoint planning method typically requires only constant time per viewpoint, but scales with $|\partial S|$ if $\phi < 2\pi$ and the relevant frontier arc consists of both curved and radial segments. Path planning to new viewpoints is trivial as the straight line between viewpoints is always in S. Reactive collision avoidance can then be handled by a local planner like Smooth Nearness Diagram navigation. Finally, the search for leader subroutine is also straightforward as it must simply pick a point on the free boundary of the robot's current sensor footprint to drive towards.

Therefore, we have the following result which satisfies the memory assumption in Section 4.1.2.

Lemma 4.4.1 (Constant Memory per Robot) *The distributed clearing algorithm requires each searcher have an amount of memory proportional to that required to store two perceptions.*

Notice that this statement is *per robot*: the frontier-guarding property ensures that the team of searchers will divide the global frontier $\mathcal{F}$ into finite sized pieces.

4.4.4 Detecting Completion

When the environment has been cleared, all searchers will be in the Wander state. If all-to-one communication is available (e.g., if all robots have even a very low-bandwidth connection to a central command center), then detecting task completion is trivial. In the most general case, the wandering searchers will have to reach a consensus that the task is complete by querying other searchers when they encounter them. In the absence of global localization or other means of assuring rendezvous, our proposal is that robots in the Wander state clump together when they encounter each other to form wandering blobs. Eventually, through the random walks of these growing blobs, all searchers will be joined into a single blob and task completion can be easily detected.

4.4.5 Handling Agent Failure

The distributed clearing algorithm relies on maintaining complete coverage of the global frontier at all times. The failure of any searcher in the frontier-guard

state, therefore, has the potential to recontaminate the cleared area and require restarting the algorithm. However, the algorithm can be made quite robust to random failures with a few minor modifications, at the cost of requiring a larger robot team.

The two mission-critical robot behaviors are Frontier-guard and Expand. To handle the potential failure of a frontier-guard, all followers of the guard could hold duplicate copies of the guard's perception and local frontier. The followers would regularly communicate with the guard to check that it is functioning and, if it fails, then one follower would take its place. If a high degree of robustness is required for a particular application, the algorithm could be modified to ensure each guard always has one or more followers. The failure of a searcher in the Expand state could be handled by one of its followers in a similar manner. In addition, when a frontier-guard commands a searcher to expand to a particular point, it can regularly check the expander's progress and dispatch another agent if necessary.

4.5 Experimental Results & Numerical Analysis

To demonstrate the utility of the proposed distributed clearing algorithm, we implemented it using the open-source Multirobot Integration Platform (MIP) [48] and the Player/Stage robot software system [39]. The clearing algorithm and related modules were implemented using the MIP architecture, which provides a multi-tasking estimation/control framework, a realistic simulation environment, and allows direct porting for execution on real robots. Perceptions are implemented as local coverage grids with 5 cm resolution[2], with oriented frontier arcs handled as ordered sequences of cells. Each robot stores only its most recent perception and its local frontier.

4.5.1 Hospital Wing Simulation

Figure 4.7 presents a larger simulation in a more complex environment modeled after part of a wing of a hospital. The environment is $16m$ wide and $20m$ tall, with a number of small patient rooms around a central desk, as well as a couple storage closets and other rooms at the bottom of the map.

Six simulated d-searchers with $d = 2.0\ m$ begin in the largest room in the bottom right corner. The first image shows the result of the first expansion, with

[2]Such a discretization of local space is useful and common in practice, the resolution of the local grid should be chosen based on the minimum detectable evader size.

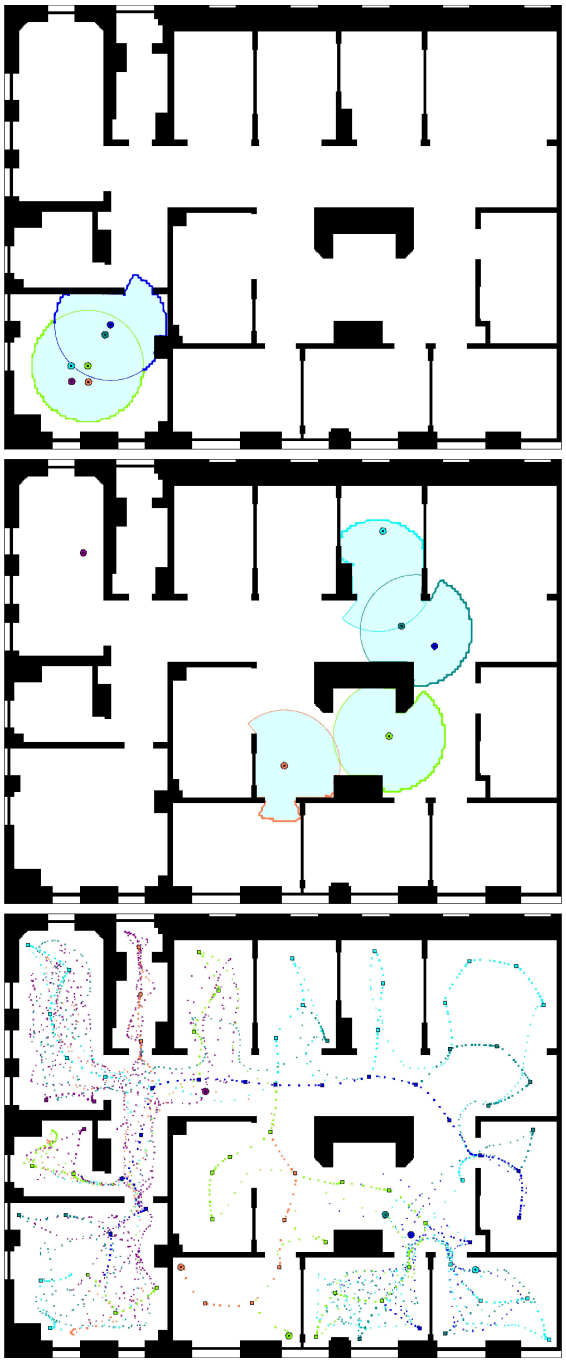


Figure 4.7: Three screenshots from a simulation of six d-searchers clearing a portion of a hospital wing. The paths of the agents are shown at right, with all viewpoints drawn with larger squares

the blue robot having reached its viewpoint and erasing the first third of the initial full circle frontier. The initial frontier-guard then dispatches another follower to cover the second third, before expanding the final third itself.

By the middle image, the team has swept through all of the lower rooms. Five of the searchers are engaged in covering and expanding the frontier, while the purple robot remains behind. The purple robot was part of the group of four which cleared the bottom left room, and all four of those searchers entered the Wander state once that room was clear. While most of these searchers found their way to an active frontier, the purple searcher is still wandering.

The final image shows the recorded trajectories of each agent, with viewpoints indicated with larger squares. The bottom set of rooms, as well as the rooms in the top right, show a dense set of tracks of searchers. The density in these rooms is a result of multiple searchers repeatedly executing the Wander behavior after clearing these parts of the map. The clear lines in the top right and middle left show the efficient, simple paths taken by searchers executing repeated expansions.

4.5.2 Hardware Experiments

The distributed clearing algorithm was experimentally validated using Khepera III robots. Each robot is equipped with a wi-fi card and a Hukuyo URG-04LX laser sensor. The latter has a field-of-view of 240° and a range artificially limited to 0.8 m. Simple odometry is used to provide mutual localization and Smooth Nearness Diagram navigation is used to avoid obstacles. Each robot is controlled by a separate process and they communicate with each other using a wireless network.

A complete experiment is summarized in Fig. 4.8, where each column contains a camera image and the relative perceptions for a distinct phase of the algorithm. In the first two panels (a,e) one robot acts as frontier-guard while the others are followers. By the second phase (b,f), the first dead-end corridor has been cleared and two frontier-guards are set to sweep the next two corridors. In the third phase (c,g) one robot simulates a motor fault, which forces the two other robots to complete the task by themselves. In the end (d,h), the environment is fully cleared and the trajectories for each robot are shown, with larger boxes indicating viewpoints.

4.5.3 Area Cleared in Empty Space

In this section we use Monte Carlo simulations to study how the area cleared in an obstacle-free environment changes with the number of robots available. The

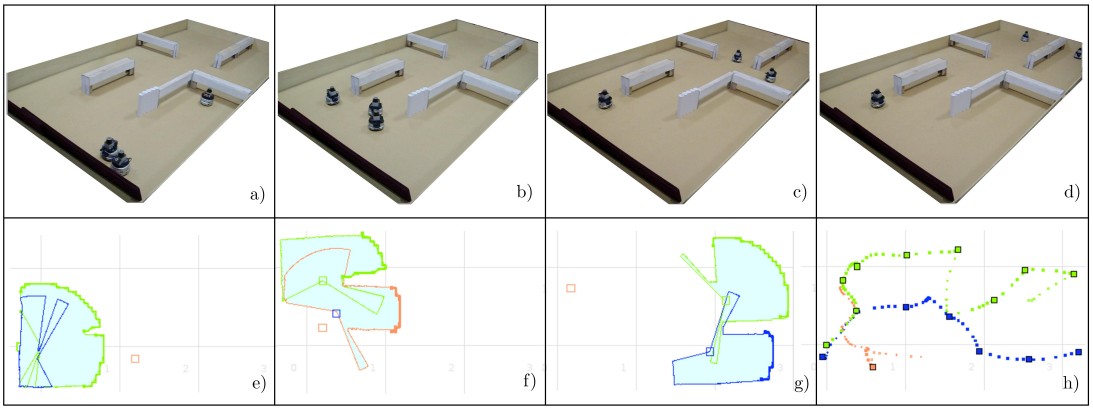


Figure 4.8: Four phases of an experiment with three Khepera III robots with Hukuyo URG-04LX laser sensors. One of the robots simulates a motor fault (b,f) which forces the others to complete the task by themselves (d,h)

simulated searchers expand from their starting position and clear as much area as they can before reaching a final equilibrium where the team would need additional searchers to continue. The theoretical limit on the area cleared in the absense of obstacles by n d-searchers occurs when the searchers are at the vertices of a n-sided regular polygon with sides of length $2d$. The cleared area in this limit is the area of the regular polygon plus the area of the sensor footprint of each searcher beyond the polygon. For n searchers, this area is given by $\frac{nd^2}{\tan\left(\frac{\pi}{n}\right)} + \frac{(n+2)\pi d^2}{2}$. We set $d = 1.0\ m$, meaning the limit on the area 12 searchers can clear is 66.8 m.

We conducted 100 simulations for 3 through 12 robots. In each trial, we chose a random agent as the initial frontier-guard, which produced differences in subsequent robot roles and timing of establishment of perceptions. The robots are only asked to get within 2 cm and 2° of a particular viewpoint, which leads to variability in the resulting perceptions and frontiers. When the first guard records the initial perception, it will has a frontier arc with angular width 2π. We divide followers evenly such that a third of the available robots will end up at each of the three viewpoints needed to expand the initial frontier.

The results of our simulations are shown in Fig. 4.9. With three searchers the Distributed Clearing Algorithm consistently cleared 95.2% of maximum possible area. This efficiency dipped to 85.2% with six searchers, then increased to 96.8% and 90.6% on average with with 9 and 12 searchers, respectively. A video of an example trial which cleared 96.2% of the optimal area is available as Online Resource 4. The variability in the area cleared with six or fewer searchers is

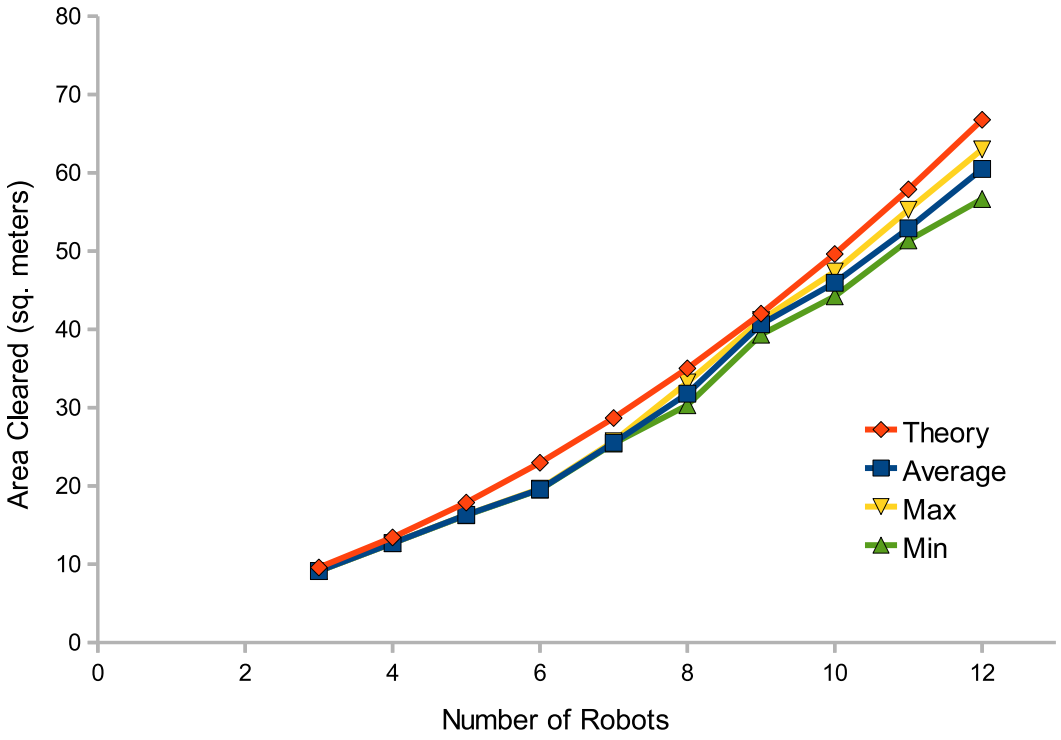


Figure 4.9: Comparison of average, maximum, and minimum area cleared in 100 simulations for different numbers of robots to the theoretical limit.

minimal because the sequence of expansions is independent of the relative timing of when robots record their perceptions. With six searchers, two go to each of the three viewpoints needed to expand the initial circular frontier. Each of these pairs will then split the next two viewpoints needed to expand their local frontier, and the team will then reach an equilibrium. With eight searchers, the number of expansions increases from 9 to 17 or 18 and these later expansions influence each other by sometimes clearing a significant portion of a neighboring guard's frontier.

These numerical results demonstrate that expansion sequence generated by the Local Viewpoint Planning Method can produce efficient global results without global information even for large numbers of searchers.

4.6 Summary

We have presented a distributed pursuit-evasion algorithm for a team of mobile robots with limited sensing and communication capabilities, limited on-board memory, and access to only local mutual localization. Our algorithm can guarantee detection of moving evaders in an unknown, non-polygonal environment with holes, provided the team consists of a sufficient number of robots. A key contribution of this algorithm is a novel method for updating the global frontier between cleared and contaminated regions using only local information. We also validated the algorithm through both simulations and hardware experiments, and discussed some theoretical and numerical results on the algorithm's performance.

Chapter 5

Conclusions

In coordination algorithms for teams of mobile robots, interesting and challenging questions arise in determining how to synthesize information and choose actions given that the robots will be distributed around a space. The work on territory partitioning and coverage in this thesis demonstrated that the kind of communication available to the robotic team changes how a coordination algorithm can be designed. For pairwise gossip communication we showed that limited communication is not always a curse. In fact, our Discretized Gossip Coverage Algorithm can achieve better final territory partitions by performing a more exhaustive search for optimal pairwise configurations at each iteration. For the one-to-base communication model we explored how time delays in the propagation of information through a robotic network can introduce a need for a different kind of distributed decision making.

In Chapter 4 we tackled a different challenge in looking at how to sweep through and clear an unknown, multiply-connected planar environment. Our distributed algorithm in this case leveraged the physical locations of key guard robots posted along the frontier between cleared and contaminated areas. These guards served as both local coordinators for updating and expanding the frontier and also as localization beacons for nearby robots. In this way we managed to clear environments of evaders using only short-range communication and local mutual localization.

5.1 Summary

We began in Chapter 2 with a partitioning and coverage control algorithm for a team of robots with only range-limited gossip communication available. Here we detailed our approach to handling complex non-convex environments

by considering a discrete set of points connected by edges to form a graph. Our pairwise approach to territory partitioning allowed us to use iterative optimal two-partitioning instead of the more common Lloyd algorithm. As we showed, this change in tactics resulted in a notable improvement in final costs in complex environments. We also presented a motion protocol which ensures that the robots meet their neighbors frequently enough.

Chapter 3 presented a second algorithm for partitioning and coverage control for a different communication model: one-to-base station communication. To handle the time delays between contact from different robots to the central base we had to adapt partition-based approaches to instead evolve overlapping regions. The result was a variant of the Lloyd algorithm which split the normal centering and partitioning steps such that the communicating robot's centroid was updated while the territories of the other robots were adjusted. We showed that this approach produces convergence to a member of the set of centroidal Voronoi partitions in finite time.

Chapter 4 looked at a visibility-based pursuit-evasion problem in which a team of mobile robots with limited sensing and communication capabilities must coordinate to detect any evaders in an unknown, multiply-connected planar environment. Our distributed algorithm in this case defined four behaviors that individual agents would switch between as local conditions dictated. One important contribution was a method for distributing the storage and maintenance of the global frontier between cleared and contaminated areas among a series of local guards positioned along the frontier. We also presented experimental and numerical results demonstrating how the algorithm works and the number of agents required to clear certain kinds of environments.

5.2 Extensions & Future Directions

Coverage with Gossip Communication The Pairwise Partitioning Rule we proposed can be adapted to work for any separable cost function provided that there is some computationally simple way to search the space of options. One immediate possibility would be equitable partitioning. In the bigger picture, the improvement in final partition cost achieved by our gossip algorithm demonstrates the potential of gossip communication in distributed coordination algorithms. There appear to be many other problems where this realistic and minimal communication model could be fruitfully applied.

Coverage with One-to-Base Station Communication We have focused on the standard coverage control cost function with the one-to-base station communication model, but there are other cost functions which could be studied. In addition, some kind of randomization in the centroid update rule might allow the system to break out of high cost local minima.

Distributed Environment Clearing One useful extension for our approach to clearing would be to guarantee a connected communication graph for the searchers at all times, perhaps including a connection back to the initial starting point. On the theoretical side, the development of bounds on the number of $d-$searchers required to clear a general environment would be a significant contribution but also appears to be very difficult to determine. Finally, the frontier concept could also be applied to three-dimensional environments.

Bibliography

[1] A. Pereira, H. Heidarsson, C. Oberg, D. Caron, B. Jones, and G. Sukhatme, "A communication framework for cost-effective operation of AUVs in coastal regions," in *Field and Service Robotics* (A. Howard, K. Iagnemma, and A. Kelly, eds.), vol. 62 of *Tracts in Advanced Robotics*, pp. 433–442, Springer, 2010.

[2] R. Smith, J. Das, H. Heidarsson, A. Pereira, F. Arrichiello, I. Cetnic, L. Darjany, M.-E. Garneau, M. Howard, C. Oberg, M. Ragan, E. Seubert, E. Smith, B. Stauffer, A. Schnetzer, G. Toro-Farmer, D. Caron, B. Jones, and G. Sukhatme, "USC CINAPS builds bridges," *IEEE Robotics & Automation Magazine*, vol. 17, no. 1, pp. 20 –30, 2010.

[3] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Magazine*, vol. 29, no. 1, pp. 9–20, 2008.

[4] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*. Applied Mathematics Series, Princeton University Press, 2009. Available at http://www.coordinationbook.info.

[5] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982. Presented as Bell Laboratory Technical Memorandum at a 1957 Institute for Mathematical Statistics meeting.

[6] 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.

[7] M. Zhong and C. G. Cassandras, "Distributed coverage control in sensor network environments with polygonal obstacles," in *IFAC World Congress*, (Seoul, Korea), pp. 4162–4167, July 2008.

[8] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *IEEE Conf. on Decision and Control*, (Cancún, México), pp. 3947–3952, Dec. 2008.

[9] M. Schwager, D. Rus, and J. J. Slotine, "Decentralized, adaptive coverage control for networked robots," *International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.

[10] R. Cortez, H. Tanner, and R. Lumia, "Distributed robotic radiation mapping," in *Experimental Robotics* (O. Khatib, V. Kumar, and G. Pappas, eds.), vol. 54 of *Springer Tracts in Advanced Robotics*, pp. 147–156, Springer, 2009.

[11] O. Baron, O. Berman, D. Krass, and Q. Wang, "The equitable location problem on the plane," *European Journal of Operational Research*, vol. 183, no. 2, pp. 578–590, 2007.

[12] S. Yun, M. Schwager, and D. Rus, "Coordinating construction of truss structures using distributed equal-mass partitioning," in *International Symposium on Robotics Research*, (Lucerne, Switzerland), Aug. 2009.

[13] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[14] F. Bullo, R. Carli, and P. Frasca, "Gossip coverage control for robotic networks: Dynamical systems on the the space of partitions," *SIAM Journal on Control and Optimization*, Aug. 2010. Submitted. Available at http://motion.me.ucsb.edu/pdf/2008u-bcf.pdf.

[15] C. Gao, J. Cortés, and F. Bullo, "Notes on averaging over acyclic digraphs and discrete coverage control," *Automatica*, vol. 44, no. 8, pp. 2120–2127, 2008.

[16] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete partitioning and coverage control with gossip communication," in *ASME Dynamic Systems and Control Conference*, (Hollywood, CA, USA), Oct. 2009.

[17] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE*, May 2010. To appear.

[18] J. MacQueen, "Some methods for the classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability* (L. M. Le Cam and J. Neyman, eds.), vol. I, pp. 281–297, University of California Press, 1965-1966.

[19] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[20] V. V. Vazirani, *Approximation Algorithms*. Springer, 2001.

[21] P. O. Fjällström, "Algorithms for graph partitioning: A survey," *Linköping Electronic Articles in Computer and Information Science*, vol. 3, no. 10, 1998.

[22] F. R. Adler and D. M. Gordon, "Optimization, conflict, and nonoverlapping foraging ranges in ants," *American Naturalist*, vol. 162, no. 5, pp. 529–543, 2003.

[23] A. Mosser and C. Packer, "Group territoriality and the benefits of sociality in the African lion, *Panthera leo*," *Animal Behaviour*, vol. 78, no. 2, pp. 359–370, 2009.

[24] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM Journal on Computing*, vol. 21, no. 5, pp. 863–888, 1992.

[25] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view," *International Journal of Robotics Research*, vol. 25, no. 4, pp. 299–315, 2006.

[26] S. Sachs, S. M. LaValle, and S. Rajko, "Visibility-based pursuit-evasion in an unknown planar environment," *International Journal of Robotics Research*, vol. 23, no. 1, pp. 3–26, 2004.

[27] A. Kolling and S. Carpin, "Multi-robot pursuit-evasion without maps," in *2010 IEEE Int. Conf. on Robotics and Automation*, (Anchorage, Alaska), pp. 3045–3051, May 2010.

[28] T. D. Parsons, "Pursuit-evasion in a graph," in *Theory and Applications of Graphs* (Y. Alavi and D. Lick, eds.), vol. 642 of *Lecture Notes in Mathematics*, pp. 426–441, Springer, 1978.

[29] M. Adler, H. Räcke, N. Sivadasan, C. Sohler, and B. Vöcking, "Randomized pursuit-evasion in graphs," *Combinatorics, Probability and Computing*, vol. 12, pp. 225–244, 2003.

[30] A. Kolling and S. Carpin, "Multi-robot surveillance: an improved algorithm for the graph-clear problem," in *2008 IEEE Int. Conf. on Robotics and Automation*, (Pasadena, CA), pp. 2360–2365, May 2008.

[31] G. Hollinger, S. Singh, and A. Kehagias, "Improving efficiency of clearing with multi-agent teams," *International Journal of Robotics Research*, vol. 29, no. 8, pp. 1088–1105, 2010.

[32] B. Jung and G. S. Sukhatme, "Tracking targets using multiple robots: The effect of environment occlusion," *Autonomous Robots*, vol. 13, no. 3, pp. 191–205, 2002.

[33] S. D. Bopardikar, F. Bullo, and J. P. Hespanha, "On discrete-time pursuit-evasion games with sensing limitations," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1429–1439, 2008.

[34] B. Yamauchi, "Frontier-based exploration using multiple robots," in *2nd Int. Conf. on Autonomous Agents*, (Minneapolis, MN), pp. 47–53, May 1998.

[35] A. Howard, M. J. Matarić, and G. S. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Autonomous Robots*, vol. 13, no. 2, pp. 113–126, 2002.

[36] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, "The sensor-based random graph method for cooperative robot exploration," *IEEE/ASME Trans. on Mechatronics*, vol. 14, no. 2, pp. 163–175, 2009.

[37] H. Minc, *Nonnegative Matrices*. Wiley, 1988.

[38] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. Springer, 2 ed., 2000.

[39] B. Gerkey, R. T. Vaughan, and A. Howard, "The Player/Stage Project: Tools for multi-robot and distributed sensor systems," in *Int. Conference on Advanced Robotics*, (Coimbra, Portugal), pp. 317–323, June 2003.

[40] J. G. Siek, L.-Q. Lee, and A. Lumsdaine, "Boost Graph Library." `http://www.boost.org`, July 2007. Version 1.34.1.

[41] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[42] J. W. Durham and F. Bullo, "Smooth nearness-diagram navigation," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, (Nice, France), pp. 690–695, Sept. 2008.

[43] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems*. Springer, 2005.

[44] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: modeling and analysis of a three-tier architecture for sparse sensor networks," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 215–233, 2003.

[45] J. W. Durham, R. Carli, and F. Bullo, "Pairwise optimal coverage control for robotic networks in discretized environments," in *IEEE Conf. on Decision and Control*, (Atlanta, GA, USA), pp. 7286–7291, Dec. 2010.

[46] A. Franchi, G. Oriolo, and P. Stegagno, "Mutual localization in a multi-robot system with anonymous relative position measures," in *2009 IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, (St. Louis, MO), pp. 3974–3980, Oct. 2009.

[47] A. Censi, "An ICP variant using a point-to-line metric," in *2008 IEEE Int. Conf. on Robotics and Automation*, (Pasadena, CA), pp. 19–25, May 2008.

[48] A. Franchi and P. Stegagno, "Multirobot Integrated Platform." `http://www.dis.uniroma1.it/~labrob/software/MIP/`, Aug. 2009.