FAULT-TOLERANT CONTROL POLICIES FOR MULTI-ROBOT SYSTEMS

BY

HYONGJU PARK

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Professor Seth A. Hutchinson, Director
Professor Geir E. Dullerud, Chair
Professor Nitin Vaidya
Assistant Professor Mohamed A. Belabbas

Abstract

Throughout the past decade, we have witnessed an active interest in distributed motion coordination algorithms for networked mobile autonomous robots. Often, in multi-robot systems, each robot executing a coordination task is a little cost, a disposable autonomous agent that has ad-hoc sensing or communication capability, and limited mobility. Coordination tasks that a group of multiple mobile robots might perform include formation control, rendezvous, distributed estimation, deployment, flocking, etc. Also, there are challenging tasks that are more suitable for a group of mobile robots than an individual robot, such as surveillance, exploration, or hazardous environmental monitoring. The field has been collectively investigated by many researchers in robotics, control, artificial intelligence, and distributed computing. However, relatively little work has been done on developing algorithms to provide resilience to failures that can occur. The problem is extremely difficult to handle in that any partial failure of a robot is not readily detectable. Some failures in robot resources can have an adverse effect on not only the performance of the robot itself, but also other robots, and the collective task performance as well.

This study presents the development of fault-tolerant distributed control policies for multi-robot systems. We consider two problems: rendezvous and coverage. For the former, the goal is to bring all robots to a common location, while for the latter the goal is to deploy robots to achieve optimal coverage of an environment. We consider the case in which each robot is an autonomous decision maker that is anonymous (*i.e.*, robots are indistinguishable to one another), memoryless (*i.e.*, each robot makes decisions based upon only its current information), and dimensionless (*i.e.*, collision checking is not considered). Each robot has a limited sensing range and can directly estimate the state of only those robots within that sensing range, which induces a network topology for the multi-robot system. We assume that it is not possible for the fault-free robots to identify the faulty robots (*e.g.*, due to the anonymous property of the robots). For each problem, we provide an efficient computational framework and analysis of algorithms, all of which converge in the face of faulty robots under a few assumptions

on the network topology and sensing abilities. A suite of experiments and simulations confirm our theoretical analysis and demonstrate that our proposed algorithms are useful in fault-prone multi-robot systems.

To my parents, for their love and support.

Acknowledgments

I must first and foremost thank my advisor, Seth Hutchinson, for being a great mentor during all these years at UIUC. Seth, thank you for your incredible help over all these years. Your understanding, your infinite patience and the dedication you show make it a pleasure to work with you. Throughout my time at UIUC, you have consistently shown faith and confidence in my abilities, for which I am eternally grateful.

I am also grateful to the other members of my thesis committee, Geir Dullerud, Nitin Vaidya, and Mohamed Ali Belabbas for taking the time to be an incredible thesis committee. I have been most lucky to count on their valuable advice, and career guidance. Especially, I would like to thank Nitin Vaidya who provided me insights to the fault-tolerant method used in this dissertation. I would also like to extend my most sincere appreciation to Scott Wyatt for making my research possible during the most difficult times. I also wish to thank people at EMS, School of Music at UIUC, in particular, Chad Wahls, Halim Beere, and Lucas Smith for their friendship and support. Finally, I would like to acknowledge Dan Block, Srinivasa Salapaka, Armand Beaudoin, Lance Hibbeler, and Robert Haber for their kindness and support. I should also mention that the administrative staff of the Department of Mechanical Engineering and Coordinate Science Lab, in particular, Kathyn Smith and Jana Lenz, has been incredibly helpful in numerous occasions.

My current and previous colleagues at the Beckman Institute have significantly contributed to making my experience at UIUC more comfortable and enjoyable. Thank you very much Han-Ul, A.J., Adrien, Usman, Jon, Jifei, Yinai, Felix, and Luke. Going through these years would have been quite harder without the support of my friends. I would like to thanks to all my friends, who supported me or influenced me along the way. Although it would be impossible to list every name, but some of those that come to mind include Seungho, Junho, and Youngmin, I am particularly grateful for your friendship all along these years.

Finally, and most importantly, I want to thank my family. My parents, Youngmok and Kumok,

thank you for your unconditional support and care, in the bad moments as much as in the good ones. I am proud to say that I would not be the person I am today without the teaching, and care they have always provided me. My sister, Yujong, thanks for your endless love and support. Without the love and patient endurance of my spouse, Jihae, my doctoral work would have been impossible.

Table of Contents

List of Tables

List of Figures

List of Symbols

$\mathbb{R}$	set of all real numbers
$\mathbb{R}_{\geq 0}$	set of non-negative real numbers
$\mathbb{Z}$	set of all integers
$\mathbb{Z}_{\geq 0}$	set of non-negative integers
$\|\cdot\|$	the Euclidean norm of a vector
$\|S\|$	cardinality of set S
$\mathcal{I}$	an index set $\{1, \ldots, n\}$
$\overline{\mathcal{I}}$	an index set of fault-free robots $\{1, \ldots, \overline{n}\}$
Q	a bounded workspace of a group of robots, usually a space in $\mathbb{R}^d$ with $d = 1, 2, 3$
$T^{\mathrm{cc}}_{\mathcal{G}} : Q^n \to Q^n$	a cicumcenter algorithm defined by a single-valued map where interconnection topology is a fixed graph $\mathcal{G}$
$\widetilde{T}^{\mathrm{cc}} : Q^n \to 2^{Q^n}$	a circumcenter algorithm defined by a set-valued map
$T^{\mathrm{dl}} : Q^n \to Q^n$	a discrete-time Lloyd's algorithm defined by a single-valued map
$\widetilde{T}^{\mathrm{rd}} : Q^n \times 2^{\mathcal{I}} \to 2^{Q^n}$	a distributed robust deployment algorithm defined by a single-valued map (with a set of robots that are not order$-k$ Voronoi neighbors to each other)
$T^{\mathrm{rd}} : Q^n \to 2^{Q^n}$	a distributed robust deployment algorithm (Algorithm 4) defined by a set-valued map (for all set of robots that do not guard a common region)
n	total number of robots
$\overline{n}$	total number of fault-free of robots
d	a dimension of a space
t	discrete time index
x	a vector, set, multiset, ordered set of positions of n robots
x_i	position vector for ith robot
$\widetilde{x}_i$	set of positions of ith robot plus the neighbors of ith robot
$\mathcal{X}_i$	a state space of robot i
$\mathcal{U}_i$	an input space of robot i

f	a discrete time state evolution map for a group of n robots
f_i	a discrete time state evolution map of robot i; a measure for probability of missed-detection of target by the ith robot in Chapter 3 and Chapter 7
t	discrete time index
r	common sensing, communication radius of all robots
r_i	sensing, communication radius of ith robot
$\emptyset$	an empty set
$\mathcal{G}$	a undirected/directed graph with n nodes
$\mathcal{A}(\mathcal{G})$	adjacency matrix for graph $\mathcal{G}$
$\mathcal{G}_{r-\text{disk}}$	$r-$disk graph
$\mathcal{G}_{\text{D}}$	a Delaunay graph
$\mathcal{V}$	vertex set
$\mathcal{E}$	edge set
$\mathcal{N}_i$	index set of neighbors of ith robot
$\widetilde{\mathcal{N}}_i$	index set of neighbors of ith robot plus the ith robot i
$\overline{\mathcal{N}}_i$	index set of fault-free neighbors of ith robot
$V_k(\mathcal{U})$	an order k Voronoi region associated with the set of generators $\mathcal{U}$
W_i	a region associated with ith robot in Chapter 6
F_i	a random set associated with ith robot in Chapter 5
$\mathscr{W}$	a partition of Q appears in Chapter 6
$\mathcal{R}$	a partition of Q into m disjoint regions appears in Chapter 7
$G : 2^{\mathcal{R}} \to 2^x$	a map that assign a set of regions to a set of sensors
$\mathscr{V}(x)$	a Voronoi partition of Q with the the set of generator positions x
$\mathscr{V}_k(x)$	an order$-k$ Voronoi partition of Q with the generator positions x
$\mathcal{F}$	index set of faulty robots
n_f	number of faulty robots among a group of n robots
n_{f_i}	number of faulty robots among $\mathcal{N}_i$, *i.e.*, $n_{f_i} = \lvert\mathcal{N}_i \cap \mathcal{F}\rvert$
$\widetilde{n}_{f_i}$	a number of faults in $\mathcal{N}_i$ determined by an algorithm by ith robot in Chapter 4
$n_{f_{\text{local,max}}}$	the maximum number of faults in neighbor each fault-free robot is desired to tolerate
$n_{\mathcal{F}}$	The common neighborhood size for each fault-free node in ADRC-III
$\overline{x}$	the multiset of positions of fault-free robots
$\overline{\mathcal{G}}$	a subgraph of $\mathcal{G}$ with only $\overline{n}$ fault-free robots

$\overline{\mathcal{V}}$	index set with only the fault-free robots
$\overline{\mathcal{E}}$	set of edges by removing from $\mathcal{E}$ every edge incident to faulty nodes in $\mathcal{F}$
$\text{conv}(S)$	convex hull of set S
$\text{ver}(S)$	set of vertice of a convex polytope S
$\text{diam}(S)$	diameter of a convex polytope S
$\mathcal{B}(p, r)$	the open ball of radius r centered at p
$\overline{\mathcal{B}}(p, r)$	the closed ball of radius r centered at p
$\text{CC}(S)$	circumcenter of set S
$\text{CR}(S)$	circumradius of set S
$\text{int}(S)$	interior of bounded set S
$\overline{p\,q}$	closed line segment connecting two vectors p and q
$\text{int}(\overline{p\,q})$	open line segment connecting two vectors p and q
∂S	boundary of set S
$H(p, q)$	closed halfspcae of points closer to vector p than to q
ϕ	a distribution (density) function over a bounded set
M_S	mass of set S
C_S	the mass centroid of set S
J_S	the 2nd moment of inertia of set S
$\mathcal{C}_{i,j}$	pairwise connectivity constraint set for robot i and j
$\mathcal{C}_i$	connectivity constraint set for robot i
σ_i	weight term associated with robot i for circumcenter algorithm
P	a probability measure
X	a random vector for location of a target point used in Chapter 3 and Chapter 7
D	an event that n robots detect some target
$\overline{D}$	a complement event that n robots fail to detect some target
D_i	an event that ith robot detects some target
$\overline{D_i}$	a complement event that ith robot fails to detect some target
ξ_i	weight term associated with robot i for Lloyd's algorithm
$\mathcal{H}(x, \mathscr{W})$	a cost function used in Chapter 3 that is an explicit function of vector x and partition $\mathscr{W}$ of Q
$\mathcal{H}(x, \mathcal{R}, G)$	a cost function used in Chapter 7 that is an explicit function of vector x, the partition $\mathcal{R}$ of Q, and the mapping G
k	a number used for order$-k$ (Voronoi) partition in Chapter 7

$\mathbf{M}$	state transition matrix of fault-free robots
$\det(\mathbf{A})$	determinant of $\mathbf{A}$
$\mathbf{I}_l$	$l \times l$ identity matrix
$\mathbf{1}_{l\times m}$	$l \times m$ matrix whose elements are all 1s.
γ	a lower bound used for matrix inequality in Chapter 4
$\lambda(\mathbf{P})$	coefficient of ergodicity of a row-stochastic matrix $\mathbf{P}$
$\delta(\mathbf{P})$	(2nd) coefficient of ergodicity of a row-stochastic matrix $\mathbf{P}$
$L(x,\ \lambda)$	Lagrandian with vector x and lagrange multiplier λ
∇g	gradient of a function g
$\mathcal{M}_i(x)$	a distributed counter part of a cost $\mathcal{H}(x,\ \mathcal{R}, G)$ associated with ith robot
$\mathbb{E}[W]$	expectation of random set W
$\mathbf{var}[W]$	variance of random set W
P_i	probability measure for ith robot in Chapter 5; probability measure for n robots when the k robots jointly detect target in Chapter 7]
Ω_i	a symbol used for sample space for ith robot in Chapter 5
$\mathcal{F}_i$	event space which is the collection of all subset of Ω_i
$p_{\mathrm{X}}(x)$	probability mass function for random variable X
$\mathcal{L}$	a symbol used for Lyapunov function

Chapter 1

Introduction

Throughout the past decade, we have witnessed an active interest in distributed motion coordination algorithms for networked mobile autonomous robots. These robots include not only Unmanned Ground Vehicles (UGVs), but Unmanned Aerial Vehicles (UAVs), and Autonomous Underwater Vehicles (AUVs) as well. Often, in multi-robot-systems, each robot executing a coordination task is a low cost, disposable autonomous agent that has ad-hoc sensing or communication capability, and limited mobility. Coordination tasks that a group of multiple mobile robots might perform include (pattern) formation control, rendezvous, distributed estimation, deployment, flocking, etc. Also, there are challenging tasks that are more suitable for a group of mobile robots than an individual robot, such as surveillance, exploration, or hazardous environmental monitoring. Examples for practical uses of networked multi-robots include warehouse automation, forest fire monitoring, oil-spill monitoring, and ocean sampling, to name a few. The related fields of research for this topic include autonomous mobile robots, swarm robotics, robotic networks, and mobile sensor networks. The field has been collectively investigated by many researchers in robotics, control, artificial intelligence, and distributed computing.

The *mobility* of robots, and robots' interactions with the environment make them prone to fail in many cases. Bernardine *et al.* [3], discussed three principal categories for possible failures that can occur in a networked multi-robot system.

- *Communication failures*: There are possibilities of occasional loss of messages, or loss of communication between robots.
- *Partial failure of a robot*: When a robot partially fails, it loses the ability to use some of its resources, but retains the ability to use other resources.
- *Complete robot failure*: When a robot completely fails, it will no longer function.

Relatively little work has been done for multi-robot coordination algorithms to provide resilience to failures that can occur. The problem is extremely difficult to handle in that any partial failure of a

robot is not easily detectable. Some failures in robot resources can have an adverse effect on not only the performance of the robot itself, but also other robots, and the collective task performance as well. If a distributed coordination algorithm is designed to work properly only under an ideal environment, a partial failure of some robot can result in failure of a cooperative mission. It is also of an independent interest to consider the worst-case scenario that can occur by faulty robots in a networked multi-robot system, *e.g.*, there can be malicious robots which aim to disrupt the performance of coordination tasks of functioning robots. A few of the related works for this example can be found in our previously generated studies [4, 5].

Some examples of a few practical failure scenarios that can happen in networked autonomous mobile robots include the following.

- *Sensor failure*: This includes incorrect sensor readings, and encompasses all scenarios in which robots are not able to efficiently collect sensor data. Since each robot makes decisions based upon its sensor data, this could result in collision with other robots, or task failure.

- *Battery depletion and power loss*: This failure is related to the complete robot failure described above. When this happens, robots will no longer operate.

There can be other hardware related failures, which could also hinder functioning robots from successfully performing their tasks.

In this study, we present distributed coordination algorithms for multi-robot systems that are resilient to possible robot failures. These include random failures experienced by individual robots, as well as the possibility of malicious behavior by some of the robots. We consider the two specific problems of rendezvous and deployment, *i.e.*, coverage control, among the coordination tasks, *e.g.*, formation control, rendezvous, distributed estimation, deployment, flocking, etc.

We note that the identities of the faulty robots in our problem are unknown, that is the faulty robots are members of the multi-robot system. Thus, our problem is different from *game theory* problems [6, 7] where each robot computes the best strategy knowing that other robots will do the same, and the typical objectives of these studies are to find the, *e.g.*, Nash equilibrium[1] [8]. Also, our problem is different from the objective of *robust control* where the main concern is robust design of a control system that is insensitive to modeling errors or uncertainty.

[1]For example, two strategies s, s' for two robots are said to be in Nash equilibrium if one robot is using s, then other robot can do no better than using s', and vice versa.

1.1 Thesis organization and highlights of chapters

This thesis is organized as follows.

- Chapter 2: In this chapter, we formally define our dynamical system composed of multiple robots, where robots have local sensing or communication capability. Next, we introduce the concept of malicious robots, and describe properties of multi-robot systems in the presence of malicious robots.

- Chapter 3: In this chapter, we briefly review recent state-of-the-art results related to rendezvous of multi-robot systems. We introduce the circumcenter algorithm with connectivity maintenance, and show that with the algorithm, all robots rendezvous to a point in finite time using LaSalle's invariance principle.

- Chapter 4: In this chapter, we propose a distributed control policy to achieve approximate rendezvous by a group of robots even when some robots in the system fail. These nonconforming robots correspond to faults in the multi-robot system, and our control policy is thus a fault-tolerant policy. Our main result is a practical distributed algorithm based upon discrete geometry, *e.g.*, Tverberg's theorem, which achieves approximate convergence in the face of faulty robots under weak assumptions on the interconnection topology. In simulation results, we show that our algorithm works better for the case of both stationary and dynamic faults, *i.e.*, when faulty robots remain stationary, than other contemporary convergence algorithms. Also, a suite of experiments are presented to confirm our theoretical results. We conclude the chapter by stating a number of future directions. The work in this chapter has appeared as two conference papers in [11, 12].

- Chapter 5: In this chapter, we consider the problem of designing distributed control algorithms to solve the rendezvous problem for multi–robot systems with limited sensing, for situations in which random robots may fail during execution. We first formulate a distributed solution based upon averaging algorithms that have been reported in the consensus literature. In this case, at each stage of execution, a 1–step sequential optimal control, *i.e.*, naïve greedy algorithm), is used. We then propose a distributed stochastic optimal control algorithm that minimizes a mean–variance cost function for each stage, given that the probability distribution for possible robot failure is known *a priori*. We show via simulation results that our algorithm provides statistically better rendezvous task performance in comparison to that of the classical circumcenter algorithm [9]

in cases of randomly occurring stationary robot failures. We conclude the chapter by stating a number of future directions. The work in this chapter has appeared as a conference paper in [13].

- Chapter 6: In this chapter, we introduce previous results on the multi-robot deployment problem. In particular, we review the Voronoi-based partitioning scheme used in mobile sensor networks, introduce Lloyd's algorithm, and provide theorems that show the convergence properties of a deployment algorithm that is based upon Lloyd's method.

- Chapter 7: In this chapter, we consider the case in which k sensors are assigned to each region in the partition, in order to obtain coverage that is robust to sensor failure. For this case, we prove that the optimal workspace partition is the order$-k$ Voronoi partition, with each sensor assigned to an order$-k$ Voronoi region for which it is a generator. The collection of associated regions for a given sensor defines its guarded region, and we prove that in the optimal configuration each sensor is located at the critical point of the distributed cost associated with its guarded region. Finally we introduce a class of distributed algorithms for our optimal sensor placement problem which require minimal inter-agent communications. The provided simulation results shows competitive coverage performance in the presence of individual node failures compared to classical Voronoi-based coverage method. The work in this chapter has appeared as a conference paper in [14].

- Chapter 8: In this chapter, we present a brief summary and possible future directions from this dissertation.

- Appendices: Appendix A briefly describes SQP, Appendix B discusses the ergodic theory, Appendix C provides several proofs for auxiliary lemmas that are used in the main proof of Chapter 4, and Appendix D introduces several concepts to represent algorithms a set-valued map and invariance principle tailored for set-valued mappings.

Chapter 2

Backgrounds and preliminaries

In this chapter, we formally define a multi-robot system to be a robotic network composed of a group of robots with local sensing or communication capabilities. In Section 2.1, we define each individual robot as a discrete-time continuous-space dynamical system. In Section 2.2, we define the interconnection topology of a group of robots in terms of the robots' sensing and/or communication capabilities. In Section 2.3, we introduce the concept of malicious robots. In Section 2.4, we describe properties of multi-robot systems that are of interested in our research. We conclude this chapter in Section 2.5 by defining terminologies that will be used in the subsequent chapters.

2.1 A multi-robot system

We consider a group of n autonomous mobile robots where each of them is contained in a bounded workspace $Q \subseteq \mathbb{R}^d$ where $d \in \{1, 2, 3\}$. We assume that there are n robots, each with an index $i \in \mathcal{I} = \{1, \ldots, n\}$ where $\mathcal{I}$ is called the *index set*. We model each robot as a discrete-time continuous-space dynamical system defined by the 3−turple $(\mathcal{X}_i, \mathcal{U}_i, f_i)$ where $\mathcal{X}_i$ is a state space, $\mathcal{U}_i$ is an input space, and $f_i : \mathcal{X}_i \times \mathcal{U}_i \to \mathcal{X}_i$ is a discrete time evolution map that defines the ith robot's motion. The state of the ith robot is defined by $x_i \in \mathcal{X}_i \subseteq Q \subset \mathbb{R}^d$. Each robot is assumed to be equipped with sensors that enable it to sense the environment, including the state of other robots. Robots may perform sensing via ad hoc communication [15] such that each robot communicates only with its local neighbors.

While the state of the multi-robot system is a vector $x = (x_1, \ldots, x_n) \in (\mathbb{R}^d)^n$, it will often be convenient to refer to the set of positions at which the robots are located. For this purpose, we use the notation $x = [x_1, \ldots, x_n]$ to denote the *multiset* of robot positions. We use multisets rather than simple sets for this purpose to allow the possibility of multiple robots occupying the same position. Note that we will use x to denote both the state, $x = (x_1, \ldots, x_n)$ and the multiset $x = [x_1, \ldots, x_n]$, relying on context to resolve any ambiguities. Also in Chapter 4, we use $n \times d$ matrix $\mathbf{x} = (x_1, \ldots, x_n)^\top$—where each point

$x_i \in \mathbb{R}^d$ is a $d \times 1$ column vector—to refer to the set of positions at which the robots are located. The ordered family of state evolution maps for the n robots, $f : (\mathcal{X}_1 \times \mathcal{U}_1) \times \cdots (\mathcal{X}_n \times \mathcal{U}_n) \to \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, is denoted by $f = (f_1, \ldots, f_n)$.

2.2 Networked robots

The communication and sensing occurring among the robots defines the interconnection topology of the multi-robot system. The topology is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which $\mathcal{V} = \mathcal{I}$ is the set of vertices, each of which corresponds to a robot, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of undirected edges. Given the set of edges, the index set of the neighbors for robot $i \in \mathcal{I}$ is defined by $\mathcal{N}_i = \{j \in \mathcal{I} \mid (i,j) \in \mathcal{E}\}$. For convenience, in the sequel we will use $\widetilde{\mathcal{N}}_i = \{j \in \mathcal{I} \mid (i,j) \in \mathcal{E}\} \cup \{i\}$ to denote the index set including both robot i and its neighbors. Similarly we will use $\widetilde{x}_i$ to denote the set of positions including both robot i and its neighbors.

If the n robots exchange messages, the interconnection topology is defined with a communication graph whose edge set is given by a set of ordered pairs $(i,\, j)$ with robots $i,\, j \in \mathcal{I}$ such that $(i,\, j) \in \mathcal{E}$ if and only if there is direct communication link from robot i to robot j. In a similar manner, based upon the sensing capabilities and configurations of the robots, the interconnection topology can be defined with a sensing graph whose edge set is the set of ordered pairs $(i,\, j)$ with robots $i,\, j \in \mathcal{I}$ such that $(i,\, j) \in \mathcal{E}$ if and only if the ith robot can detect the jth robot.

In our research, the interconnection topology of the system is represented by a *proximity graph*. A proximity graph is a graph whose vertex set is an index set of distinct points, and whose edge set is determined by the relative locations of the point set (see e.g., [16] for more details). Among proximity graphs, we are particularly interested in two types: the $r-$disk graph, and the Delaunay graph.

For an $r-$disk graph $\mathcal{G}_{r-\text{disk}} = (\mathcal{V},\, \mathcal{E})$, each robot $i \in \mathcal{I}$ has a corresponding position $x_i \in Q$. For robots i and j, $(i,\, j) \in \mathcal{E}$, if and only if $\|x_i - x_j\| < r_i$. Here, r_i denotes a sensing radius, and robot i is connected to robot j where robot j is within the sensing radius of robot i. If we assume that all robots have the same sensing radius, $r_i = r$, then $\mathcal{G}_{r-\text{disk}}$ is an undirected graph, i.e., $(i,\, j) \in \mathcal{E}$ if and only if $(j,\, i) \in \mathcal{E}$, and the connectivity relation is *symmetric*. The $r-$disk graph will be used extensively in both Chapter 4 and Chapter 5, which deal with the rendezvous problem.

The Delaunay graph $\mathcal{G}_{\text{D}} = (\mathcal{V},\, \mathcal{E})$ is the dual of the Voronoi diagram. The Voronoi diagram includes a region V_i for each $i \in \mathcal{I}$, and edge $(i,\, j) \in \mathcal{E}$ if and only if V_i is adjacent to V_j. The Voronoi diagram is described in more detail in Section 6.2.1, and in various texts on computational geometry, e.g., [17].

The Delaunay graph will be used in Chapter 7, which deals with deployment problems.

Based on context, we sometimes drop the subscripts "$r-$disk" and "D" from $\mathcal{G}$ and denote either of the graphs simply by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

2.3 Malicious agents

A Faulty Multi-robot System (F-MRS) consists of n robots, of which some are faulty. We denote by $\mathcal{F} \subseteq \mathcal{I}$ the index set of faulty robots, and by n_f the number of faulty robots, $n_f = |\mathcal{F}|$.

Because we will frequently refer to the set of fault-free robots, it is convenient to define the following notation. We denote by $\overline{\mathcal{I}}$ the index set of the fault-free robots, by $\overline{x}$ the set of their positions. The number of *fault-free* robots is denoted $\overline{n} = n - n_f$. Without loss of generality, in the sequel we will assume that the robots are indexed such that the fault-free robots have indices $\overline{\mathcal{I}} = \{1, \ldots, \overline{n}\}$ and the faulty robots have indices $\mathcal{F} = \{\overline{n}+1, \ldots, n\}$.

We define the interconnection topology of the fault-free robots by a directed graph $\overline{\mathcal{G}} = (\overline{\mathcal{V}}, \overline{\mathcal{E}})$ with $\overline{\mathcal{V}} = \overline{\mathcal{I}}$ and $\overline{\mathcal{E}} \subseteq \overline{\mathcal{V}} \times \overline{\mathcal{V}}$ obtained by removing from $\mathcal{E}$ all edges incident to faulty robot vertices. For the i^{th} robot, we denote by $\overline{\mathcal{N}}_i$ its index set of fault-free in-neighbors in the graph $\overline{\mathcal{G}}$, and by n_{f_i} the number of its faulty neighbors in the graph $\mathcal{G}$ (i.e., $n_{f_i} = |\mathcal{N}_i \cap \mathcal{F}|$).

Robots may fail in various ways, ranging from minor variations in performance to outright malicious behavior. In our research, we consider failures that lie on a spectrum ranging from so-called crash faults (in which the robot simply ceases to operate) to malicious behavior that attempts to undermine the performance of the entire multi-robot system. Faults of these types have long been a topic of research in the distributed computing literature [18]. In distributed computing, crash faults occur when a processor ceases sending messages to its neighbors, and malicious behavior is characterized as a Byzantine fault [19], in which a processor sends arbitrary, and possibly distinct, incorrect messages to its neighbors. Our problem is slightly different from that confronted in distributed computing problems. For us, in the case where robots can observe the behavior of their neighbors (using sensing capabilities), it is not possible for a malicious robot to send distinct signals to different neighbors, since all neighbors can directly observe the malicious robot's behavior. Thus, our characterization of malicious behavior includes only a subset of the Byzantine faults faced in distributed computing.

Using the terminology of Section 2.1 , we may characterize faults in terms of the state evolution map f_i and the control input u_i. If a fault-free robot has state evolution map f_i^*, then we say that the i-th robot is faulty if $f_i \neq f_i^*$ at any time during execution. In the case of crash faults, we would have

$f_i(x_i(t), u_i) = x_i(t)$, and in this case $f_i \neq f_i^*$ except for the situation in which $x_i(t)$ is an equilibrium point for the system. If we assume that robots share the same distributed control law $u^*(\cdot)$ [20], then a fault occurs any time when $u_i \neq u^*$. In the case of feedback control, u_i would generally be a function of $x_i(t)$, as well as the states of the neighbors of the ith robot, and possibly of t. Malicious behavior is thus characterized by a robot choosing to apply a control input that is different from the one prescribed by the distributed control law of the multi-robot system.

2.4 System properties

Similar to other studies (see e.g., [21]), the operation of the multi-robot system follows a cyclic behavior. The following three sequential operations create a single cycle: *Look, Compute, Move*. First in the *Look* state, each robot takes a snapshot of the current state of robots within its sensing region. Typically [22], a robot's sensing region is defined by a ball of radius r centered at its position. Next in the *Compute* state, based upon the information obtained from *Look* state each robot calculates its control input u_i. Then in the *Move* state, each robot executes its chosen control. Every robot is *memoryless* such that it generates a control input based upon only the information provided at the current time.

Robots are considered to be *dimensionless*; thus robots never collide[1] with each other, and multiple robots are allowed to be located at a same position. However, we do assume that each robot can distinguish if a point is occupied by multiple robots. This is sometimes called *multiplicity detection* [21] capability. Lastly, the robots are *anonymous*, i.e., each robot is *indistinguishable* from all other robots. Thus, if there are faulty robots, due to the anonymity property, it is not possible for any robot to identify which of its neighbors are faulty robots.

We consider synchronous systems, where every robot takes its snapshot of its local neighbors at times determined by a global clock. Thus, at all $t \in \mathbb{Z}_{\geq 0}$, for each functioning robot $i \in \overline{\mathcal{I}}$, the control system is given by

$$x_i(t+1) = f_i(x_i(t),\, u_i(t)), \quad t = 0, 1, 2, \ldots. \tag{2.1}$$

The motion control law for each robot is generated during the *Compute* state based on the information gained during the *Look* state.

[1] While we do not explicitly consider collision detection, all of the algorithms we present can be adapted to avoid collision by incorporating localized rules which do not violate the correctness or convergence properties of the algorithms.

2.5 A few other basic notations

We shall introduce a few notations we will use throughout the text. We use $\|\cdot\|$ to denote the Euclidean norm of a vector. We will use $\mathbb{R}_{\geq 0}$ to denote the non-negative real numbers, and $\mathbb{Z}_{\geq 0}$ to denote the non-negative integers. For a set $S \subset \mathbb{R}^d$ the convex hull $\mathrm{conv}(S)$ is defined as the smallest convex set containing S. We denote the boundary of the set by ∂S, and the interior of the set S as $\mathrm{int}(S)$. The cardinality of S is denoted by $|S|$. If S is a convex polytope in $\mathbb{R}^d$, then $\mathrm{Ver}(S)$ is the set of vertices of S, and $\mathrm{diam}(S)$ is the the diameter of S, which is defined by the maximum Euclidean distance between any two points in S, i.e., $\mathrm{diam}(S) = \max\{\|p - q\| \mid p, q \in S\}$. For two points $p, q \in \mathbb{R}^d$, we define $\mathrm{int}(\overline{p, q})$, $\overline{p, q}$ to be the *open* and *closed* line segment connecting points p and q respectively. Also we let $H(p, q) = \{s \in \mathbb{R}^d \mid \|s - p\| \leq \|s - q\|\}$ denote the *closed halfspace* of $\mathbb{R}^d$ of points closer to p than to q. If $d = 2$, we may use the term *halfplane* instead of *halfspace*. Given a bounded space $S \subset \mathbb{R}^d$, with a density function $\phi : S \to \mathbb{R}_{\geq 0}$ defined over S, and some point $p \in \mathbb{R}^d$, we define $M_S := \int_S \phi(q)\, dq$ to be the mass of S, $C_S := \frac{\int_S q\phi(q)\, dq}{\int_S \phi(q)\, dq}$ to be the mass centroid of S, and $J_{S,p} := \int_S \|q - p\|^2 \phi(q)\, dq$ to be the 2nd moment of inertia of S with respect to p. Given a square matrix $\mathbf{M} \in \mathbb{R}^{l \times l}$, $\mathbf{M} \succ 0$ means that $\mathbf{M}$ is positive definite. A matrix is *positive*, if every element in the matrix is positive. Given a square matrix $\mathbf{A}$, we denote by $[\mathbf{A}]_{ij}$ the (i, j)th element of $\mathbf{A}$, and by $[\mathbf{A}]_k$ the kth column of $\mathbf{A}$. We denote by $\mathbf{I}_n$ the $n \times n$ identity matrix, and by $\mathbf{1}_{n \times 1}$ the $n \times 1$ column vector whose elements are all 1s. For arbitrary $n \times n$ square matrices $\mathbf{B}$, $\mathbf{C}$, we write $\mathbf{B} \leq \mathbf{C}$ if $[\mathbf{B}]_{ij} \leq [\mathbf{C}]_{ij}$ for all pairs $i, j \in \{1, \ldots, n\}$. We use LHS as a shorthand for the left-hand side of an equation. Similarly, RHS as a shorthand for right-hand side. For $x \in \mathbb{R}^d$, and $S \subset \mathbb{R}^d$

$$\rho(x, S) = \inf_{y \in S} \|x - y\|$$

defines the distance of x from S. A sequence of points $(x(l))_{l=0}^{\infty}$ approaches a set S i.e., $x(l) \to S$ as $l \to \infty$ means

$$\rho(x(l), S) \to 0 \text{ as } l \to \infty.$$

The *closure* of a set $S \subset \mathbb{R}^d$ is defined by

$$\mathrm{cl}(S) = \{x \in \mathbb{R}^d \mid \rho(x, S) = 0\}.$$

Thus a set S is closed, if $S = \mathrm{cl}(S)$, and open if its complement is closed. An alternative definition of a closed set uses the definition of an open set. For $p \in \mathbb{R}^d$, we define $\mathcal{B}(p, r)$, $\overline{\mathcal{B}}(p, r)$ by the *open* and *closed* ball of radius r centered at p, i.e., $\mathcal{B}(p, r) = \{q \in \mathbb{R}^d \mid \|q - p\| < r\}$, and $\overline{\mathcal{B}}(p, r) = \{q \in \mathbb{R}^d \mid \|q - p\| \leq r\}$. A set S is open, if for each $x \in S$, there is $\epsilon > 0$ such that the $\epsilon-$neighborhood of x is contained in S, i.e., $\mathcal{B}(x, \epsilon) \subset S$. A set S is closed, if its complement is open. Given a real vector $v = [v_1, \ldots, v_l]^\top \in \mathbb{R}^l$, $v \geq 0$ means $v_i \geq 0$ for all $i = 1, \ldots, l$. We shall use both notations $(x(t))_{t=0}^{\infty}$ and $(x(t))$ (if the context is clear) to denote an infinite sequence $x(0)$, $x(1)$, $x(2), \ldots, x(t)$, $x(t+1), \ldots$.

Chapter 3

Related studies: Multi-robot rendezvous

One of the most well-studied topics in multi-robot coordination is the problem of bringing a distributed group of robots to the same physical location in $\mathbb{R}^2$ or $\mathbb{R}^3$. There are two versions of this problem. In the first version the goal is to arrive to an exact agreement on the physical location of robots. In robotics, this problem is referred to as *rendezvous* [1, 16, 20, 23, 24], or *gathering* [21, 25–29], and it is closely related to the distributed *consensus* problem, which studies agreement between locally communicating agents (not necessarily mobile agents arriving to a physical location) [30–32]. The second version of the problem is concerned with obtaining approximate solutions; every robot approaches the same physical location in an asymptotic manner. This type of problem is referred to as an $\epsilon-$*rendezvous* or *convergence* problem, or as *approximate consensus* in non-robotics communities.

Some of the earliest work in multi-robot rendezvous[1] problems is by Ando *et al.* [1,9], who proposed the *circumcenter algorithm*. The circumcenter algorithm utilizes a geometric center called the *circumcenter* from discrete geometry. Roughly speaking, given a point set, the circumcenter is the center of the smallest ball enclosing all the points in the set. According to the algorithm, each robot moves towards the circumcenter of the positions of its neighbors and the robot itself, while preserving the connectivity[2] with its neighbors. Lin and Morse [23, 24, 34] provided a convergence result for the circumcenter algorithm using a Lyapunov-like method given an initially connected graph for both synchronous [24] and asynchronous systems [34]. Later on, Cortes *et al.*, [35] provided a more general result by showing convergence of the circumcenter algorithm using LaSalle's Invariance principle under switching, *i.e.*, time-varying topology condition, and additionally showed the algorithm's robustness when link failures occur between nodes, under a few assumptions on interconnection topology.

Independently, those studies in [21, 25–29, 36] deal with the gathering problem, which usually considers finite-time convergence of a group of multiple robots with unlimited visibility[3]. These problems

[1] Texts such as [33] state in a rather different way that the rendezvous problem is a special case of the gathering problem where the system contains only two robots.

[2] Connectivity in the graph that represents the interconnection topology of the robots.

[3] Each robot can detect or see every other robot.

have been investigated in both synchronous and asynchronous systems under various assumptions on robot capability. Such capabilities of the robot include agreement on the coordinate system, consistent compass, multiplicity detection, and unlimited mobility. A detailed analysis and description of the state of the art on this subject are contained in Chapter 3 of [33] and the references therein.

The rendezvous and gathering problems are closely related to the *consensus* problem, which is a popular topic in both computer science [18] and control [30, 31, 37–43] literature. In particular, those studies in control communities concern multi-agent coordination that aims at convergence of the states of the agents to a common value. These studies use system theoretic approaches, and have various names, *e.g.*, consensus [37–40], flocking[4] [31], and formation control [41, 42] in which the communication links between agents are either time-dependent or time-independent. There are also studies [30, 43] that are not limited to a specific task, but can be applied for a general coordination tasks for multi-agent systems.

The remainder of this chapter is organized as follows. Section 3.1 formally defines the rendezvous task. Then we provide details of the circumcenter algorithm in Section 3.2. In Section 3.3, we provide a brief analysis of the algorithm, and introduces LaSalle's Invariance principle for non-deterministic algorithms. Section 3.3.2 provides a theorem which states finite-time convergence of a group of robots executing the circumcenter algorithm. Section 3.4 concludes this chapter by providing discussions of the approaches used in the chapter.

3.1 Rendezvous of a group of robots

We formally define both exact and approximate rendezvous of a group of robots as follows.

Definition 3.1.1 (An exact rendezvous)**.** *Let $x(t) = \{x_1(t), \ldots, x_n(t)\}$ be the set of positions of n robots at time $t = 0, 1, 2 \ldots$, where $x_i(t) \in Q \subset \mathbb{R}^d$ with $d = 1, 2, 3$. Suppose that the interconnection topology of the n robots is represented by an $r-$disk graph $\mathcal{G}_{r-\mathrm{disk}}(t) = (\mathcal{V}, \mathcal{E}(t))$. Then, the n robots are said to (exactly) rendezvous at time t, if the following relation holds:*

$$x_i(t) = x_j(t), \;\; (i,\, j) \in \mathcal{E}(0). \tag{3.1}$$

Recall that $\mathcal{E}(0)$ is the set of edges of the graph $\mathcal{G}_{r-\mathrm{disk}}(0)$ at initial time 0. Thus, if the graph of n robots is initially connected, and the robots exactly rendezvous at some time $t > 0$, then $x_1(t) =$

[4] Flocking is a special case of the distributed consensus problem for multi-agent systems or swarms in which the agents must agree upon a common heading value. If applied to animals, this is also called schooling and herding.

$\cdots = x_n(t)$. Note that according to the definition, when robots rendezvous, they do not need to be *stationary*[5].

Definition 3.1.2 (An approximate rendezvous)**.** *Let $x(t) = \{x_1(t), \ldots, x_n(t)\}$ be the set of positions of n robots at time $t = 0, 1, 2 \ldots$, where $x_i(t) \in Q \subset \mathbb{R}^d$ with $d = 1, 2, 3$. Suppose that the interconnection topology of n robots is represented by an $r-$disk graph $\mathcal{G}_{r-\mathrm{disk}}(t) = (\mathcal{V}, \mathcal{E}(t))$. Then, the n robots are said to approximately rendezvous, if for each $\epsilon > 0$, there exists $T \in \mathbb{Z}_{\geq 0}$ such that $t > T$ implies*

$$\|x_i(t) - x_j(t)\| < \epsilon, \ \ (i, j) \in \mathcal{E}(0). \tag{3.2}$$

An alternative definition of $\epsilon-$rendezvous can be found in Chapter 4 of [20], which uses the condition that the Euclidean distance between the position of each robot and the average of positions of all robots must be less than some distance $\epsilon > 0$.

3.2 A rendezvous algorithm: the circumcenter algorithm

In this section, we review the circumcenter and circumradius of a bounded set, and introduce the circumcenter algorithm [1, 16, 23], which is one of the well-studied rendezvous algorithms. We also introduce a connectivity constraint set that provides sufficient conditions for preserving pairwise connectivity between robots, which is used for the circumcenter algorithm.

As its name implies, the circumcenter algorithm requires the computation of the circumcenter of a point set. This is defined formally as follows.

Definition 3.2.1 (Circumcenter of a bounded set [9])**.** *The circumcenter of a bounded set $S \subset \mathbb{R}^d$, denoted by* $\mathrm{CC}(S)$ *is the* center *of the closed ball of minimum radius that contains S. The circumradius of S denoted by* $\mathrm{CR}(S)$ *is the* radius *of this smallest ball.*

It is known that the circumcenter is *unique* [44]. The problem of obtaining the circumcenter and circumradius of a bounded set $S \subset \mathbb{R}^d$ can be formulated as the following convex programming problem [44] in $r \in \mathbb{R}$ and $p \in \mathbb{R}^d$:

$$\begin{aligned} \underset{r,\, p}{\text{minimize}} \quad & r \\ \text{subject to} \quad & \|q - p\| \leq r, \ \ \forall q \in S, \end{aligned}$$

[5] A robot i is stationary at time $t \geq 0$ if $x_i(t) = x_i(t+1)$

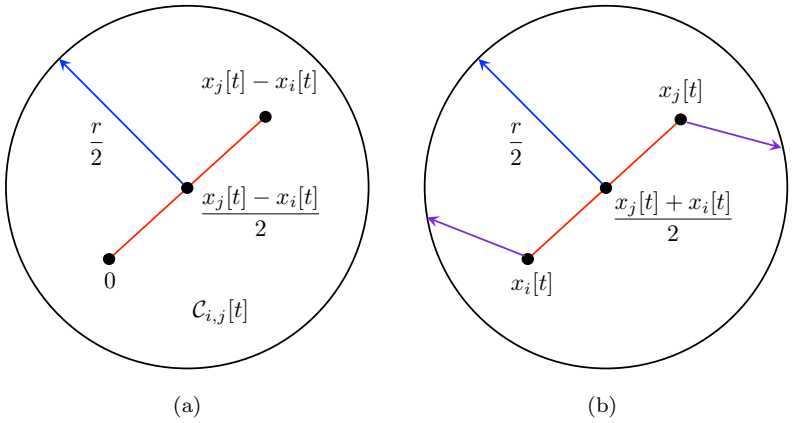


Figure 3.1: (a) Pairwise connectivity constraint $\mathcal{C}_{i,j}(t)$, (b) Proof for Proposition 3.3.3 (Inspired by the figure of Ando *et al.*, [1]).

where r and p which solve the problem are the circumradius of S, and the circumcenter of S respectively. This is also known as *minimum covering sphere problem* in some literature [44, 45].

During the execution of the circumcenter algorithm, it is necessary to ensure that network connectivity is not compromised. For each robot, the connectivity constraint [1, 9, 23, 24, 34, 35] provides a *sufficient condition* for maintaining connectivity with its neighbors at the next iteration, *i.e.*, stage, step, using only *local information*, such as the positions of its own neighbors. Thus, such a connectivity constraint can be used for distributed algorithms. We define this constraint as follows. Let $\mathcal{C}_{i,j}(t)$ be the *pairwise* connectivity constraint set for two distinct robots, robot i and j, which are neighbors to each other, *i.e.*, $\|x_i(t) - x_j(t)\| \leq r$ at time t where r is the sensing radius. The pairwise connectivity constraint is defined by

$$\mathcal{C}_{i,j}(t) = \overline{\mathcal{B}}\left(\frac{x_j(t) - x_i(t)}{2}, \frac{r}{2}\right). \tag{3.3}$$

Fig. 3.1(a) shows the pairwise connectivity constraint with robots i and j. Suppose that at time $t = 0, 1, 2\ldots$, robot $l = 1, \ldots, n$ evolves with a discrete time system $x_l(t+1) = x_l(t) + u_l(t)$. Then if at time t, ith and jth robots that are neighbors, *i.e.*, $\|x_i(t) - x_j(t)\| \leq r$, and they choose their control to be $u_i(t) \in \mathcal{C}_{i,j}(t)$, and $u_j(t) \in \mathcal{C}_{j,i}(t)$ respectively, then at time $t+1$, the two nodes i and j remain neighbors, *i.e.*, $\|x_i(t+1) - x_j(t+1)\| \leq r$.

If there are neighbors of robot i other than robot j, the connectivity constraint set with all the neighbors is exactly the intersection of all pairwise constraint sets associated with robot i. We denote by $\mathcal{C}_i(t)$ the intersection:

$$\mathcal{C}_i(t) = \bigcap_{j \in \mathcal{N}_i(t)} \mathcal{C}_{i,j}(t). \tag{3.4}$$

The circumcenter algorithm is a distributed rendezvous algorithm that was first introduced by Ando *et al.* [1]. The convergence properties of the algorithm were further studied by a few other researchers including Lin and Morse [23, 24], and Cortes *et al.* [16, 20]. As was discussed in Section 2.4, at every stage, each robot executes the following operations:

- *Look*: Each robot determines the positions of its neighbors.
- *Compute*: Each robot calculates its circumcenter, and its connectivity constraint set.
- *Move*: Each robot moves towards the circumcenter while maintaining connectivity with its neighbors, and respecting its maximum travel distance per stage.

We note that the circumcenter of a point set $\{x_j(t)\}_{j\in\widetilde{\mathcal{N}}_i(t)}$ is equivalent to the circumcenter of the convex hull of of the point set $\{x_j(t)\}_{j\in\widetilde{\mathcal{N}}_i(t)}$, *i.e.*,

$$\mathrm{CC}\left(\{x_j(t)\}_{j\in\widetilde{\mathcal{N}}_i(t)}\right) = \mathrm{CC}\left(\mathrm{conv}\left(\{x_j(t)\}_{j\in\widetilde{\mathcal{N}}_i(t)}\right)\right). \tag{3.5}$$

To be used in the sequel, we define a new symbol $\mathrm{CC}_i(t)$ to be the circumcenter of node i and its neighbors at stage t, *i.e.*,

$$\mathrm{CC}_i(t) := \mathrm{CC}\left(\{x_j(t)\}_{j\in\widetilde{\mathcal{N}}_i(t)}\right). \tag{3.6}$$

The circumcenter algorithm can be applied to a simple discrete time dynamical system by

$$x_i(t+1) = x_i(t) + u_i(t), \qquad t = 0,\, 1,\, 2, \ldots, \tag{3.7}$$

where $u_i(t)$ is defined by

$$u_i(t) = \sigma_i^*(t)(\mathrm{CC}_i(t) - x_i(t)), \quad \sigma_i^*(t) \in [0,\ \ 1], \quad u_i(t) \in \mathcal{C}_i(t) \cap \overline{\mathcal{B}}(\mathbf{0},\, v_{\max}) \tag{3.8}$$

where $\sigma_i^*(t)$ is a weight between 0 and 1 that appropriately assigns the desired position $x_i(t+1)$ along a closed line segment $\overline{x_i(t), \mathrm{CC}_i(t)}$, $v_{\max} > 0$ is the maximum travel distance for every robot during one stage, and $\mathbf{0}$ is a $d \times 1$ zero vector. In particular, if we are interested in the control vector $u_i(t)$ with

the largest magnitude[6], we obtain $\sigma_i^*(t)$ from the following problem:

$$\begin{aligned} &\underset{\sigma_i(t)\in[0,\ 1]}{\text{maximize}} && \sigma_i(t) \\ &\text{subject to} && \sigma_i(t)(\mathrm{CC}_i(t) - x_i(t)) \in \mathcal{C}_i(t) \cap \overline{\mathcal{B}}(\mathbf{0},\, v_{\max}). \end{aligned}$$

This representation of the circumcenter law for a simple discrete-time dynamical system (3.7)-(3.2) is similar to that given in [35] of Cortes *et al.* The circumcenter algorithm is guaranteed to achieve exact rendezvous. We will discuss this in the following section.

3.3 Analysis of the circumcenter algorithm

In this section we present an analysis of the correctness properties of the circumcenter algorithm. Our approach follows closely that presented in [24, 35], but we include this here because many techniques used here will be applied in Chapter 7 as well. For those who are familiar with multi-robot rendezvous may prefer to skip this section.

First, we state propositions that define two important properties of the circumcenter of a convex polytope, which will be used to show convergence property of the circumcenter algorithm.

Proposition 3.3.1 (Blumenthal [45])**.** *Given a closed convex polytope $S \subset \mathbb{R}^d$, the following relation holds.*

$$\mathrm{CC}(S) \in S \setminus \mathrm{ver}(S). \tag{3.9}$$

The proof is contained [45].

Proposition 3.3.2 (Proposition 2.1 [35], Proposition 1, [24])**.** *Let S be a non-empty set in $\mathbb{R}^d$. If $p \in S \setminus \mathrm{CC}(S)$, and $S \subset \overline{\mathcal{B}}(p, r')$, then*

$$\left(\mathrm{int}(\overline{p, \mathrm{CC}(S)}) \cap \overline{\mathcal{B}}\left(\frac{p+q}{2}, \frac{r'}{2}\right)\right) \neq \emptyset \tag{3.10}$$

holds for all $q \in S$.

Proof. First we note that the open line segment $\mathrm{int}(\overline{p, \mathrm{CC}(S)})$ is non-empty because $p \neq \mathrm{CC(S)}$. We consider two cases:

[6]We assume that each robot moves toward its circumcenter while maintaining connectivity with its neighbors under its limitations on mobility as closely as it can.

Case 1: $\|q-p\| < r'$: In this case, it is sufficient to show that p is an interior point of the closed ball $\overline{\mathcal{B}}\left(\frac{p+q}{2}, \frac{r'}{2}\right)$. Since $\|q-p\| < r'$, $p \in \mathcal{B}\left(\frac{p+q}{2}, \frac{r'}{2}\right)$, which implies

$$\left(\text{int}(\overline{p, \text{CC}(S)}) \cap \overline{\mathcal{B}}\left(\frac{p+q}{2}, \frac{r'}{2}\right)\right) \neq \emptyset$$

as required.

Case 2: $\|q-p\| = r'$: Since p is not the circumcenter of S *e.g.*, $p \neq \text{CC}(S)$, and the circumcenter is unique, it can be deduced that $\text{CR}(S) < r'$. In other words, there is a circumcenter $\text{CC}(S) \neq p$ of the set S that has the corresponding circumcenter $\text{CR}(S)$ smaller than r'. By this, both $\text{CC}(S) \in \overline{\mathcal{B}}(p, \text{CR}(S))$ and $\text{CC}(S) \in \overline{\mathcal{B}}(q, \text{CR}(S))$ hold, and this implies that $\text{CC}(S) \in \overline{\mathcal{B}}(p, \text{CR}(S)) \cap \overline{\mathcal{B}}(q, \text{CR}(S))$. By using the fact that $\text{CR}(S) < r'$, and $\text{CC}(S) \in \overline{\mathcal{B}}(p, r') \cap \overline{\mathcal{B}}(q, r')$, it can be deduced that $\left(\text{int}(\overline{p, \text{CC}(S)}) \cap \overline{\mathcal{B}}\left(\frac{p+q}{2}, \frac{r'}{2}\right)\right) \neq \emptyset$ as required.

A proof similar to this is contained in pg. 2 [35]. ■

It is formally stated in the following proposition that the connectivity constraint can be used to maintain connectivity between a group of robots.

Proposition 3.3.3 (Lemma 4.2, Chapter 4 [20])**.** *Let $x(t) = \{x_1(t), \ldots, x_n(t)\}$ be the set of positions of n robots at some time $t \in \mathbb{Z}_{\geq 0}$, $x_j \in Q \subset \mathbb{R}^d$. We assume that the following are true.*

- *The interconnection topology at time t is defined with an $r-$disk graph $\mathcal{G}_{r-\text{disk}}(t) = (\mathcal{V}, \mathcal{E}(t))$*
- *At time $t = 0, 1, 2, \ldots$, each robot $i = 1, \ldots, n$ evolves by $x_i(t+1) = x_i(t) + u_i(t)$, and chooses its control $u_i(t)$ to be in its connectivity constraint set $\mathcal{C}_i(t)$ (3.3-3.4),* e.g., *$u_i(t) \in \mathcal{C}_i(r)(t)$.*

Then, $\|x_i(t) - x_j(t)\| \leq r$ holds for each $(i, j) \in \mathcal{E}(0)$, at all $t = 0, 1, 2, \ldots$.

Proof. Let us first show that given two robots i, and j which are neighbors to each other at time instant t, *e.g.*, $\|x_i(t) - x_j(t)\| \leq r$, if $u_i(t) \in \mathcal{C}_{i,j}(t)$ and $u_j(t) \in \mathcal{C}_{j,i}(t)$, then

$$\|x_i(t+1) - x_j(t+1)\| \leq r.$$

Before we proceed, we define *sum*, and *difference* between set, and vector. Given a bounded set $S \subset \mathbb{R}^d$ and a vector $s \in \mathbb{R}^d$, the sum is $s + S = \{s + s' \,|\, s' \in S\}$. It can be shown that

$$x_i(t+1) \in x_i(t) + \mathcal{C}_{i,j}(t) = \overline{\mathcal{B}}\left(\frac{x_i(t) + x_j(t)}{2}, \frac{r}{2}\right), \tag{3.11}$$

and

$$x_j(t+1) \in x_j(t) + \mathcal{C}_{j,i}(t) = \overline{\mathcal{B}}\left(\frac{x_i(t)+x_j(t)}{2},\, \frac{r}{2}\right). \tag{3.12}$$

Both $x_i(t+1)$, $x_j(t+1)$ being in the same closed ball with radius $\frac{r}{2}$ implies

$$\|x_i(t+1) - x_j(t+1)\| \leq r$$

(see Fig. 3.1(b)). By repeating this process with the rest of the neighbors in $\mathcal{N}_i(t)$, it is routine to show that the connectivity is maintained with all neighbors of robot i, and the result follows. ■

We note that Proposition 3.3.3 implies that an infinite sequence of $r-$disk graphs $(\mathcal{G}_{r-\text{disk}}(t))_{t=0}^{\infty}$ that represents the interconnection topology of $(x(t))_{t=0}^{\infty}$ satisfies the following relations

$$\mathcal{G}_{r-\text{disk}}(0) \subseteq \mathcal{G}_{r-\text{disk}}(1) \subseteq \cdots \subseteq \mathcal{G}_{r-\text{disk}}(t) \subseteq \cdots, \quad t = 2,\, 3,\, \ldots \tag{3.13}$$

3.3.1 The circumcenter algorithm as a map

In this subsection, we represent the circumcenter algorithm with connected topologies as a set-valued mapping $\widetilde{T}^{\text{cc}} : Q^n \to 2^{Q^n}$. For convenience of analysis, using (3.7-3.2), we define a singleton-valued map $T^{\text{cc}}_{\mathcal{G}_{r-\text{disk}}(t)} : Q^n \to Q^n$ whose ith component map $T^{\text{cc}}_{\mathcal{G}_{r-\text{disk}}(t),i} : Q^n \to Q$ is given by

$$T^{\text{cc}}_{\mathcal{G}_{r-\text{disk}}(t),i}(x_1(t), \ldots, x_n(t)) = x_i(t) + \sigma_i^*(t)(\text{CC}_i(t) - x_i(t)) \tag{3.14}$$

where $\sigma_i^*(t)$ is obtained from (3.2). Note for a fixed topology represented by the graph $\mathcal{G}_{r-\text{disk}}(t)$, $T^{\text{cc}}_{\mathcal{G}_{r-\text{disk}}(t),i}$ depends continuously on the locations of the ith robot and its neighbors, *e.g.*, $\{x_j(t)\}_{j\in\widetilde{\mathcal{N}}_i(t)}$ at $t = 0,\, 1,\, 2\ldots$. We formally put this as a proposition.

Proposition 3.3.4. *Given a fixed topology $\mathcal{G}_{r-\text{disk}}(t)$ at $t = 0,\, 1,\, 2, \ldots$, the singleton-valued map $T^{\text{cc}}_{\mathcal{G}_{r-\text{disk}}(t)}$ is continuous on Q^n.*

A set-valued map can be used to take into account the time-varying topology of a group of robots along the trajectory of the circumcenter algorithm. Set-valued maps can be used to model non-deterministic algorithms [30, 35, 46]. We define the set valued map $\widetilde{T}^{\text{cc}} : Q^n \to 2^{Q^n}$ by

$$\widetilde{T}^{\text{cc}}(x(t)) = \{T^{\text{cc}}_{\mathcal{G}_{r-\text{disk}}}(x(t)) \in Q^n \mid \mathcal{G}_{r-\text{disk}} = (\mathcal{V},\, \mathcal{E}) \text{ is connected}\}, \tag{3.15}$$

i.e., the set valued map $\widetilde{T}^{\text{cc}}(x(t))$ is collection of all possible maps $T^{\text{cc}}_{\mathcal{G}_{r-\text{disk}}}(x(t))$ at different interconnection topologies that are connected. The set $\widetilde{T}^{\text{cc}}(x(t))$ is *finite* because the number connected graph with a vertex set $\mathcal{V}$ is *finite*. We note from (3.15) that $T^{\text{cc}}_{\mathcal{G}_{r-\text{disk}}(t)}(x(t)) \in \widetilde{T}^{\text{cc}}(x(t))$ holds for each $t = 0, 1, 2, \ldots$.

A set-valued map $\widetilde{T}^{\text{cc}}$ is said to be *closed* at y^*, if for any convergent sequences $y^m \to y^*$, $z^m \to z^*$, such that $z^m \in \widetilde{T}^{\text{cc}}(y^m)$ holds for each $m = 0, 1, 2, \ldots$, then $z^* \in \widetilde{T}^{\text{cc}}(y^*)$. The map $\widetilde{T}^{\text{cc}}$ is said to be closed on Q^n, if it is closed for all $y \in Q^n$. The closedness of set-valued map is analogous to the continuity of an ordinary map. The following proposition states that the map T is closed on Q^n.

Proposition 3.3.5 (Proposition 4.3 [35])**.** *The set-valued map $\widetilde{T}^{\text{cc}}$ is closed on $Q^n \subset (\mathbb{R}^d)^n$.*

A proof[7] for proposition is contained in [35]. The proof is based upon the definition of the closed set-valued map.

Proposition 3.3.6 (Proposition 4.3 [35])**.** *For each $t = 0, 1, 2, \ldots$, $x(t+1) \in \widetilde{T}^{\text{cc}}(x(t))$ implies* $\text{conv}(x(t+1)) \subset \text{conv}(x(t))$.

Proof. By Proposition 3.3.1 and (3.5), for each $x_i(t)$,

$$\text{CC}_i(t) \in \text{conv}\left(\{x_j(t)\}_{j \in \widetilde{\mathcal{N}}_i(t)}\right)$$

holds for all $i = 1, \ldots, n$. By the definition of convex-hull, for each $i = 1, \ldots, n$,

$$\text{conv}\left(\{x_j(t)\}_{j \in \widetilde{\mathcal{N}}_i(t)}\right) \subset \text{conv}(x(t))$$

holds and this implies that for each $i = 1, \ldots, n$, $\text{CC}_i(t) \in \text{conv}(x(t))$. Hence, $\overline{x_i(t), \text{CC}_i(t)} \in \text{conv}(x(t))$ holds. Thus, from (3.7)-(3.2) it can be deduced that $x_i(t+1) \in \text{conv}(x(t))$ holds for each $i = 1, \ldots, n$, and again by the definition of convex hull of a point set, $\text{conv}(x(t+1)) \subseteq \text{conv}(x(t))$ holds for $t = 0, 1, 2, \ldots$. ∎

3.3.2 Exact rendezvous with the circumcenter algorithm

In this subsection, we prove that a set of robots with initially connected topology executing the circumcenter law will converge to a point in finite time using LaSalle's Invariance Principle for closed

[7] In [35], the set-valued map that was shown to be closed is the *cirumcenter algorithm at all strongly connected topologies*, and this is more general than our map $\widetilde{T}^{\text{cc}}$ (3.15). Thus their proposition 4.3 [35] can be used.

set-valued maps. Suppose that there is a set $Q \in \mathbb{R}^d$, and let there be an algorithm that is represented by a set valued map $\widetilde{T}^{\text{cc}} : Q^n \to 2^{Q^n}$. Given n point set $q^0 \in Q^n$, a *trajectory* of the algorithm defined by $\widetilde{T}^{\text{cc}}$ from q^0 is a sequence $(q^k)_{k=0}^{\infty}$ which satisfies

$$q^{k+1} \in \widetilde{T}^{\text{cc}}(q^k), \quad k = 0,\, 1,\, 2, \ldots \tag{3.16}$$

A trajectory $(q^k)_{k=0}^{\infty}$ *approaches* a set U if, for every neighborhood[8] L of U, there exists $k' > 0$ such that $q^k \in L$ for $k \geq k'$. We write $q^k \to U$, as $k \to \infty$. A set M is *weakly positively invariant* w.r.t. $\widetilde{T}^{\text{cc}}$, if for each $q \in M$ there exists $q' \in \widetilde{T}^{\text{cc}}(p)$ such that $q' \in M$. The continuous function $\mathcal{L} : (\mathbb{R}^d)^n \to \mathbb{R}$ is *non-increasing along* $\widetilde{T}^{\text{cc}}$ on $Q^n \subset \mathbb{R}^d$, if $\mathcal{L}(q') \leq \mathcal{L}(q)$ for all $q \in Q^n$, and all $q' \in \widetilde{T}^{\text{cc}}(q)$. We note that the function V with such properties is often referred to as *Lyapunov* function.

Remark 3.3.1. *The Theorem D.4.1 in the Appendix D (LaSalle's Invariance Priciple for closed set-valued maps) can be used to show the a convergence of non-deterministic discrete-time dynamical system whose trajectory $\widetilde{T}^{\text{cc}}$ is computed by recursively setting q^{k+1} to an arbitrary element in the set $\widetilde{T}^{\text{cc}}(q^k)$ for each $k = 0, 1, \ldots$, where $\widetilde{T}^{\text{cc}}$ is a closed set-valued map [16, 47]. Note that the theorem requires that there must be a continuous function $\mathcal{L}$ non-increasing along every trajectory of $\widetilde{T}^{\text{cc}}$. In the following subsection, we chose the Lypunov function $\mathcal{L}$ to be the diameter of the group of n robots, and use Theorem D.4.1 to show that a multi-robot system evolving with circumcenter algorithm beginning with initially connected interconnection topology, will converge to a point.*

Next, to be used in the sequel, we define the term: *consensus* of points.

Definition 3.3.1 (Consensus of n points)**.** *An ordered set of points $x = (x_1, \ldots, x_n) \in (\mathbb{R}^d)^n$ is called a* consensus *if $x_1 = x_2 = \cdots = x_n$.*

Next, we show that a group of robots whose interconnection topology is initially connected, executing the circumcenter algorithm defined by the set-valued map $\widetilde{T}^{\text{cc}}$ will rendezvous in finite time. For the proof, we use LaSalle's Invariance Principle stated in Theorem D.4.1, and choose Lyapunov candidate function to be the diameter of the position set of the n robots.

$$\mathcal{L}(x(t)) := \operatorname{diam}(x(t)) = \max_{i,\, j \in \mathcal{I}} \|x_j(t) - x_i(t)\| \tag{3.17}$$

The method of using the diameter of a point set as a Lyapunov function has been previously used

[8] A neighborhood of a set $U \subseteq S$ is a subset of S that, for each point $u \in U$, contains an open ball centered at u.

in a number of papers [24, 30, 35]. In order to qualify as a Lyapunov function, the function $\mathcal{L}$ must be non-increasing along $\widetilde{T}^{\text{cc}}$ in Q^n. In the next proposition, we show that the function $\mathcal{L}$ is indeed non-increasing along the algorithm $\widetilde{T}^{\text{cc}}$.

Proposition 3.3.7. *For all $t = 0, 1, 2, \ldots$, $\mathcal{L}(x(t+1)) \leq \mathcal{L}(x(t))$ holds, where $x(t+1) \in T(x(t))$.*

Proof. This is an immediate consequence of Proposition 3.3.6. In other words, for each $t = 0, 1, 2, \ldots$, $\text{conv}(x(t+1)) \subset \text{conv}(x(t))$ implies $\mathcal{L}(x(t+1)) \leq \mathcal{L}(x(t))$ where $x(t+1) \in T(x(t))$ [24, 30, 35]. ■

We note that the Proposition 3.3.6 implies that for every $x(t)$ in the trajectory $(x(t))_{t=0}^{\infty}$ we have $x_i(t) \in \text{conv}(x(0))$ holds for $i = 1, \ldots, n$, with all $t = 0, 1, 2, \ldots$. To be used in the sequel, we define a new symbol D to denote the compact set $\text{conv}(x(0))$, *e.g.*, $D := \text{conv}(x(0))$. We are ready to state the main result in the following theorem, which is similar to that provided in [20, 35].

Theorem 3.3.1 (An exact rendezvous with the circumcenter law [24,35])**.** *Let $x(t) = \{x_1(t), \ldots, x_n(t)\}$ be the set of positions of n robots at time $t = 0, 1, 2, \ldots$, where $x_i(t) \in Q \subset \mathbb{R}^d$. Suppose that the interconnection topology of n robots are represented by an $r-$disk graph $\mathcal{G}_{r-\text{disk}}(t) = (\mathcal{V}, \mathcal{E}(t))$, and initially at $t = 0$, the graph is connected. If at $t = 0, 1, 2, \ldots$, each robot executes the circumcenter algorithm, then all n robots will exactly rendezvous at a point in finite time.*

Proof. The proof below follows closely that given in Theorem 4.6 of [35]. We provide a brief summary of the proof. We show via LaSalle's invariance principle (Theorem D.4.1) that robots executing circumcenter algorithm with connected topologies defined by a set-valued map (3.15) at each stage approaches an invariant set that is a consensus. Since it is given that the initial topology is connected and the connectivity is maintained by the algorithm, the theorem holds as a special case. Then we show the finite time convergence by the definition of the circumcenter and circumcenter algorithm.

We note that the followings are true:

- According to Proposition 3.3.5, (3.14), and (3.15), a group of robots executing circumcenter algorithm that is denoted by a set-valued map $\widetilde{T}^{\text{cc}}$ is *closed* on D^n.
- The function $\mathcal{L}$ defined in (3.17) is *continuous*, and by Proposition 3.3.7, $\mathcal{L}$ is *non-increasing* along $\widetilde{T}^{\text{cc}}$. Thus $\mathcal{L}$ is a Lyapunov function w.r.t. $\widetilde{T}^{\text{cc}}$.
- For every $x(t)$ in the trajectory $(x(t))_{t=0}^{\infty}$ of $\widetilde{T}^{\text{cc}}$, $x_i(t) \in D$ for $i = 1, \ldots, n$ at each $t = 0, 1, 2, \ldots$, which implies that the trajectory is *bounded*.

Let M be the weakly positively invariant set defined by

$$M := \{q \in Q^n \mid \exists q' \in \widetilde{T}^{\mathrm{cc}}(q) \text{ such that } \mathcal{L}(q') = \mathcal{L}(q)\}.$$

We are ready to apply LaSalle's Invariance Principle for non-deterministic algorithms (Theorem D.4.1). Using the theorem, we can show that every bounded trajectory generated by $\widetilde{T}^{\mathrm{cc}}$ under time-varying topologies approaches M, *e.g.*,

$$x(t) \to M, \quad \text{as } t \to \infty. \tag{3.18}$$

We claim that M is $n-$tuple of points in consensus. Let Γ[9] be the ordered set of n points in consensus

$$\Gamma = \{(q, \ldots, q) \in D^n \mid q \in D\}. \tag{3.19}$$

Then our claim becomes $M = \Gamma$. We show that both inclusions $M \subset \Gamma$, and $\Gamma \subset M$ hold. First note that for every $q' \in \Gamma$, $q' \in M$ holds, which implies that $\Gamma \subset M$.

Next we show the inclusion $M \subset \Gamma$ by a contradiction. Suppose that $M \nsubseteq \Gamma$ such that $M \setminus \Gamma \neq \emptyset$. We claim that the set $M \setminus \Gamma$ is a weakly positively invariant set with the desired property as defined. We consider a point set $q'' = \{q_1'', \ldots, q_n''\} \in M \setminus \Gamma$, then $\mathrm{diam}(q'') > 0$ holds. In what follows, we show that by our algorithm T, for each $p' \in \mathrm{ver}(\mathrm{conv}(q''))$, $p'' \in \mathrm{conv}(q'') \setminus \mathrm{ver}(\mathrm{conv}(q''))$ holds for all $p'' \in \widetilde{T}^{\mathrm{cc}}(p')$. We can show this by Proposition 3.3.1, Proposition 3.3.2 , and Proposition 3.3.3 which states that interconnection topology of n robots from initially connected topology, will say connected by the connectivity constraint imposed on the circumcenter algorithm. First note that by Proposition 3.3.3, unless n robots are in consensus, the connected topology implies that for each $q_i'' \in q'' \subset M \setminus \Gamma$, there is at least one point $q_j'' \in q''$, with $q_i'' \neq q_j''$, such that $\left\|q_i'' - q_j''\right\| \leq r$. Since $\mathrm{diam}(\mathrm{conv}(q''))$ is determined by points in $\mathrm{ver}(\mathrm{conv}(q''))$, given each vertex $q_m'' \in \mathrm{ver}(\mathrm{conv}(q''))$, there is at least one point in $q_l'' \in q''$ with $q_m'' \neq q_l''$ such that $\|q_m'' - q_l''\| \leq r$. Let $\mathcal{G}^\star$ be some connected graph. Then, according to Proposition 3.3.1 and Proposition 3.3.2, for any such $\mathcal{G}^\star$, $T^{\mathrm{cc}}_{\mathcal{G}^\star}(q'') \subseteq \mathrm{conv}(q'') \setminus \mathrm{ver}(\mathrm{conv}(q''))$, holds. In other words, the relation holds for each circumcenter algorithm defined by a point-to-point map $T^{\mathrm{cc}}_{\mathcal{G}^\star}$. By the definition of the set-valued map $\widetilde{T}^{\mathrm{cc}}$ from (3.15), $q''' \subseteq \mathrm{conv}(q'') \setminus \mathrm{ver}(\mathrm{conv}(q''))$ holds for all $q''' \in \widetilde{T}^{\mathrm{cc}}(q'')$, and this implies that $\mathrm{diam}(q''') < \mathrm{diam}(q'')$ holds for all $q''' \in \widetilde{T}^{\mathrm{cc}}(q'')$, *e.g.*, $V(q''') < V(q'')$ holds for all $q''' \in \widetilde{T}^{\mathrm{cc}}(q'')$. Thus, the set M does not satisfy the condition $V(q''') = V(q'')$ where $q''' \in \widetilde{T}^{\mathrm{cc}}(q'')$, which is a contradiction. Thus, $M = \Gamma$ holds as required. Hence, by the Lasalle's Invariance principle

[9] In [35], Γ is termed as the *diagonal set of* D^n such that $\mathrm{diag}(D^n) := \{(q, \ldots, q) \in D^n \mid q \in D\}$.

stated in Theorem D.4.1, the trajectory $(x(t))_{t=0}^{\infty}$ of the algorithm $\widetilde{T}^{\mathrm{cc}}$ will approach the set Γ.

Next, we show that the sequence converge to a point instead of the set Γ. Note that since every trajectory of $\widetilde{T}^{\mathrm{cc}}$, *e.g.*, $(x(t))_{t=0}^{\infty}$ where $x(t+1) \in \widetilde{T}^{\mathrm{cc}}(x(t))$, is bounded, there must be a convergent subsequence which converges to a consensus.

Then, using the properties of Lyapunov function $\mathcal{L}$ (continuity and monotonicity), we show that entire sequences converge to the limit of the convergent subsequence, *e.g.*, consensus. Let $x^\star \in \Gamma$. We note that the following are true:

- For $t = 0, 1, 2, \ldots, \mathcal{L}(x(t+1)) \leq \mathcal{L}(x(t))$ holds where $x(t+1) \in \widetilde{T}^{\mathrm{cc}}(x(t))$.
- There exists a convergent subsequence with infinite set of non-negative integers in its natural order[10] $\mathcal{K}$ such that

$$x(t) \to x^\star \tag{3.20}$$

  as $t \to \infty$ with $t \in \mathcal{K}$, where $x^\star \in \Gamma$.

We show that

$$\lim_{t\to\infty} \mathcal{L}(x(t)) = \lim_{t\in\mathcal{K}} \mathcal{L}(x(t)) = \mathcal{L}(x^\star) = 0 \tag{3.21}$$

holds for some $x^\star \in \Gamma$. By continuity of $\mathcal{L}$, we have

$$\lim_{t\in\mathcal{K}} \mathcal{L}(x(t)) = \mathcal{L}(x^\star). \tag{3.22}$$

From the 1st bullet, the sequence $(\mathcal{L}(x(t)))_{t=0}^{\infty}$ is non-increasing such that for any $l \in \mathbb{Z}_{\geq 0}$,

$$\mathcal{L}(x^\star) \leq \mathcal{L}(x(l)). \tag{3.23}$$

holds. Using (3.22), and definition of limit, for each $\epsilon > 0$, there is $t_\epsilon \in \mathcal{K}$ such that for $t \geq t_\epsilon$, and $t \in \mathcal{K}$,

$$\mathcal{L}(x(t)) - \mathcal{L}(x^\star) < \epsilon \tag{3.24}$$

By the monotonic property of the sequence $(\mathcal{L}(x(l)))_{l=0}^{\infty}$, for all $l \geq t_\epsilon$,

$$\mathcal{L}(x(l)) \leq \mathcal{L}(x(t_\epsilon)). \tag{3.25}$$

[10] An example of such sequence is $\mathcal{K} = 0, 1, 7, 9, \ldots$.

In sum, we have $\mathcal{L}(x(l)) - \mathcal{L}(x^\star) < \epsilon$ for all $l \geq t_\epsilon$, and this implies that $x(l) \to x^\star$ as $l \to \infty$, *e.g.*, the entire sequence converges to $x^\star$.

Lastly, we show that the convergence is in *finite* time. The above asymptotic convergence result implies that for each $\epsilon' > 0$ there exists some time $t_{\epsilon'} \geq 0$ such that for $t' > t_{\epsilon'}$, $\|x_i(t) - x_i^\star\| < \epsilon'$ holds for all $i = 1, \ldots, n$. Let us consider $\epsilon' = \min(v_{\max}, \frac{r}{2})$, then at time $t'' = t_{\epsilon'} + 1 > t_{\epsilon'}$, $\mathrm{CC}_i(t'') = \mathrm{CC}_j(t'')$ for all i, $j \in \mathcal{I}$ which implies an exact rendezvous $x_1(t'') = \cdots = x_n(t'')$ at time t'', as required. ■

3.4 Discussion

All the propositions, and theorems contained in this chapter have been proven previously by other researchers [24, 30, 35, 47, 48]. A similar versions of Theorem 3.3.1 can be found in [20, 24, 35]. For the convergence proof of Theorem 3.3.1, we used the discrete-time version of LaSalle's Invariance principle (See Theorem D.4.1 in the Appendix D) [49] extended to set-valued maps[11]. Our proof closely follows that given in [35]. Independently, Lin and Morse [24] used a standard Lyapunov argument to prove Theorem 3.3.1.

A similar convergence result of Theorem 3.3.1, can be also obtained using the set-valued Lyapunov theory discussed in [30, 43]. Independent of the above mentioned research, Lorenz [50] proposed a theorem using the proper, equi-proper convex-hull averaging map that bears similarity with Moreau's set-valued Lyapunov theorem [30, 43], but requires weaker conditions than Moreau's theorem. More details on the difference between the two approaches are stated in [51].

[11] The complete proof of the LaSalle's Invariance principle for non-determinmistic algorithm is contained in the Appendix D.

Chapter 4

Multi-robot robust rendezvous algorithm: a combinatorial approach

In this chapter, we consider a fault-tolerant version of the multi-robot rendezvous problem. In most all approaches that have been reported for the rendezvous problem, it is implicitly assumed that all of the individual robots will faithfully and accurately execute an agreed upon, decentralized control policy. Here, we consider the case in which some of the robots, termed *faulty* robots, fail to follow the policy. For multi-robot systems, this could result from physical failure due to component malfunction (*e.g.*, sensor error), or to depletion of energy (*e.g.*, dead batteries). More generally, faults might even include adversarial scenarios, such as the presence of malicious robots which try to maximally degrade the performance of coordination tasks [4, 5].

We present what we believe to be the first fault-tolerant, distributed consensus algorithm for the case of robots operating in a multi-dimensional workspace. The contributions of the chapter are twofold. First, we present a distributed algorithm for convergence of fault-free robots in the presence of malicious robots. The algorithm relies on the concept of a Tverbeg partition of a pointset in $\mathbb{R}^d$. The key property of a Tverberg partition is that there is a non-empty intersection of the convex hulls of the elements in partition. This property allows the construction of fault-tolerant control policies that do not rely on coordinate averaging schemes. Second, we give a rigorous analysis of the algorithm's performance, combining elements of the methods given in the decentralized control community [31, 32], in which faulty robots are not considered, and from the fault-tolerant computing community [52, 53], in which network topology is independent of system state. Our analysis also provides a bound on the number of iterations required to achieve approximate convergence within a specified error bound. We demonstrate our algorithm for three different sensing models, evaluating performance via extensive simulation.

Fault tolerance has been a concern in digital computing since its earliest days [54], and our fault-tolerant rendezvous problem has a close relationship to the classical *Byzantine Generals* problem in distributed computing [19]. The two problems have several key differences, *e.g.*, for the Byzantine Generals problem the communication links are fixed, while in the multi-robot scenario communication

links depend on the distances between pairs of robots; however, the overall goal for the Byzantine Generals problem (achieving agreement among the fault-free processors, in spite of the presence of faulty processors in the network) is analogous to the goal of achieving rendezvous. The approximate Byzantine consensus problem is an extension of the original Byzantine generals problem for which the goal is to allow the fault-free processors to agree on a value asymptotically [55–57]. Recently, there have been studies of approximate Byzantine consensus problems where the consensus value is a d-dimensional vector in Euclidean space. This problem is termed as Byzantine *vector* consensus [58]. Because of the similarities between the Byzantine Generals problem and the rendezvous problem, we are able to exploit a number of results from the area of fault-tolerant distributed computing, particularly the results found in [52, 53].

Independent from those studies on fault-tolerant consensus in digital computing communities, there is also a body of research on fault-tolerant consensus [59–71] and fault-tolerant rendezvous or gathering [28, 35, 72–76, 76–78] in the control and robotics communities respectively. We will review here a few of those studies that are closely related to the problem of interest in this chapter.

In the control community, a number of studies have focused on the consensus problem for the special case of a scalar consensus variable. A median-based consensus algorithm is described in [61, 79], in which each node uses only its own value and the median of its neighbors' values for the state update. In a similar manner, Leblanc *et al.*, [56, 80] consider a protocol where each node removes neighbors' values that are extreme with respect to its own value. Their study is an extension of *resilient consensus protocol* previously investigated in [81]. There are also several studies that use control theoretic approaches to provide provably correct distributed consensus algorithms for cases in which malicious or unreliable links are present *e.g.*, [63–66, 68, 69]. Again, in all of these approaches, the consensus variables are scalars, taking their values on the real-line.

A number of approaches have focused on robustness to changes in network topology, caused, for example, by adversarial nodes that can cause loss of edges or even nodes from the connectivity graph, *e.g.*, [56, 63, 82]. In [82], the authors prove robustness of a special class of random graphs to such adversarial behaviors, and also show that the problem of determining robustness is coNP-complete. In [62], Khanafer *et al.*, proposed a zero-sum game between groups of nodes and an adversary where the group of nodes executes robust distributed averaging while the adversary strategically disconnects a set of links to prevent the nodes from converging. They formulate two versions of the problem which are competition between two players whose action is reversed *i.e.*, min-max and max-min problems.

The Maximum principle was used to obtain optimal strategies for both problems and also to provide sufficient conditions for existence of saddle points. We note that these approaches consider only a limited class of possible faults - those that result in changes to network connectivity. In contrast, our approach is able to deal a more general class of faults in which nodes may exhibit malicious behavior while remaining anonymous, and without altering network topology.

Finally, we note the work of Zhu and Martinez [59, 60] who consider the special case of adversarial nodes launching replay attacks[1]. They have proposed a novel distributed resilient algorithm for multi-vehicle systems based upon a receding-horizon control method that converges to a desired formation regardless of any replay attacks by adversarial nodes. Their model requires each node to have a memory. Again, our approach is able to handle this type of fault, as well as much more general classes of faults, and our method does not require memory.

There have been a number of attempts to solve the fault-tolerant gathering/rendezvous problem in the robotics community [28,35,72–78]. However, all of these rely on the assumption that each fault-free robot can see all other robots in the workspace, *i.e.*, each fault-free robot has unlimited visibility, which implies a fully connected communication graph. Under this condition, Agmon and Peleg [72] presented a correct algorithm to gather all functioning robots when one of the robots crashes permanently. Defago *et al.*, [28] extended Agmon and Peleg's previous work and showed feasibility of probabilistic gathering under various assumptions related to synchrony and crash/Byzantine faults. Bouzid *et al.*, [74] proposed an algorithm to gather all fault-free robots in the presence of multiple crash faults. In their algorithm, Weber points[2], which have the key property of remaining unchanged under straight line movements of any of the points towards or away from it, were used. The same authors also proposed Byzantine tolerant gathering algorithm [75–77] for multi-robots moving in a line. Their approach is a combination of that of Leblanc [56] and Zhang [61], in that in their algorithm each robot uses a trimming method to remove up to f largest and f smallest values from their neighbors, and takes the median of the values that are left.

In the remainder of the chapter, we present our approach to solving the fault-tolerant distributed consensus problem. In Section 4.1 we present the concept of a Tverberg partition, which is the key concept underlying our approach. In Section 4.2 we present our fault-tolerant algorithm, which we refer to as ADRC. The main theoretical result of the chapter, a theorem that guarantees convergence under mild connectivity assumptions, is presented in Section 4.3. Section 4.4 presents three different

[1] Adversarial nodes consecutively repeating the control commands for a period of time

[2] The geometric median of set of points in a Euclidean space is the point minimizing the sum of distances to the points [83].

instantiations of the ADRC algorithm based on different sensing models and their performance is analyzed via number of simulation results in Section 4.5 and a suite of experiments using a multi-robot testbed in Section 4.6. Finally, Section 4.7 concludes the chapter with a few remarks.

4.1 Tverberg Partitions and Safe Points

Our fault-tolerant algorithms rely on the ability to construct a partition of n points into $n_f + 1$ disjoint subsets whose convex hulls have a non-empty intersection. As we describe below, this implies that the non-empty intersection will consist of points that lie in the convex hull of the set of fault-free nodes. In this section we review results from discrete geometry that establish when and how such a partition can be constructed.

Definition 4.1.1 (An r-divisible point set [84])**.** *A set of n points is r-divisible if it can be partitioned into r pairwise disjoint subsets such that the intersection of the convex hulls of these r subsets is non-empty.*

Using the definition, we state the classical Tverberg's theorem, which provides conditions that guarantee a given point set to be r-divisible.

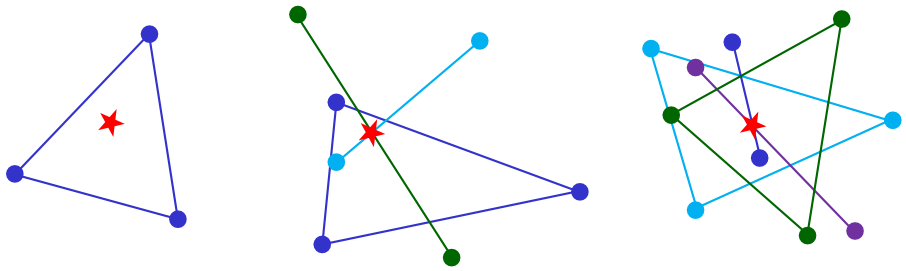
Theorem 4.1.1 (Tverberg's Theorem [84])**.** *Any set of n points in $\mathbb{R}^d$ is r-divisible if $n \geq (d+1)(r-1)+1$.*

Corollary 4.1.1 (Maximum Tverberg Partition)**.** *The maximum value of r for which a set of n points in $\mathbb{R}^d$ is guaranteed to be r-divisible using Theorem 4.1.1 is $r = \lceil n/(d+1) \rceil$.*

The result follows from straightforward computations using the bound in Theorem 4.1.1.

Fig. 4.1 shows examples of point sets in $\mathbb{R}^2$ that are r-divisible, for $n = 4, 7, 10$ and $r = 2, 3, 4$ respectively. Note in Fig. 4.1(a), the four points are partitioned into a set containing the three vertices of the triangle and a set containing only the point lying inside the triangle, the latter also being the only point in the intersection of the convex hulls of the two elements of the partition.

A partition $\Pi = \{P_1 \dots P_r\}$ such that $\cap \text{conv}(P_i) \neq \emptyset$ is called a *Tverberg partition*, and the size of the partition $|\Pi| = r$ is called the *Tverberg depth* or merely *depth* when the context is clear. Note that for a given point set of size n, the Tverberg partition of depth r is not necessarily unique. A point $p \in \cap_i \text{conv}(P_i)$ is called a *Tverberg point of depth r*. A Tverberg point for a point set in $\mathbb{R}^d$ is analogous to the concept of the median for a point set in $\mathbb{R}$.



(a) 4 points, $r = 2$ (b) 7 points, $r = 3$ (c) 10 points, $r = 4$

Figure 4.1: Examples showing Tverberg points obtained with different number of points and division in $\mathbb{R}^2$ (circle: point, star: a Tverberg point).

Suppose $x = \{x_1 \dots x_n\}$ specifies the set of states of n robots, of which n_f are faulty. If x is $(n_f + 1)$ divisible, *i.e.*, if $n \geq (d+1)n_f + 1$, then we can partition x into subsets $P_1, \dots P_{n_f+1}$, and at least one of these sets will contain only fault-free robots (since there are $n_f + 1$ sets but only n_f faulty robots). Furthermore, since $\cap_i \text{conv}(P_i) \subseteq \text{conv}(P_j)$ for all $j \in \{1, \dots, n_f + 1\}$, any Tverberg point of depth $n_f + 1$ is contained in the convex hull of a set of fault-free nodes. This motivates the following definition of safe point.

Definition 4.1.2 (Safe point)**.** *For a set of n points in $\mathbb{R}^d$, of which at most n_f correspond to the positions of faulty nodes, a point p is n_f-safe (referred to as an n_f-safe point) if it has a neighborhood that is guaranteed to lie in the convex hull of the $n - n_f$ fault-free nodes.*

There are at least two ways to ensure that a point p is n_f-safe: (*i*) It is a Tverberg point of depth $n_f + 1$ that lies in the relative interior of the non-empty intersection of the convex hulls of $(n_f + 1)$-disjoint subsets. These subsets constitute the associated Tverberg partition. (*ii*) For every subset of size $\overline{n}$, p has a neighborhood that lies in the convex hull of the subset.

Our algorithm for fault-tolerant rendezvous explicitly constructs d-dimensional neighborhoods of safe points at each iteration. The following proposition provides a method to construct such a neighborhood.

Proposition 4.1.1. *For a set of n points in $\mathbb{R}^d$ that is $(n_f + 1)$- divisible, if $z_1, \dots, z_{d+1}$ are Tverberg points of depth $n_f + 1$ in general position, each $q \in \text{ri}(\text{conv}(\{z_1, \dots, z_{d+1}\}))$ is n_f-safe.*

Proof. By the properties of Tverberg point, for each choice of the subset Q with size $\overline{n}$,

$$z_i \in \text{conv}(Q), \ i = 1, \dots, d+1. \tag{4.1}$$

Under the condition that $z_1, \dots, z_{d+1}$ is in general condition $\text{ri}(\text{conv}(z_1, \dots, z_{d+1})) \neq \emptyset$. By the def-

inition of convex sets, (4.1) implies $\text{conv}(z_1, \ldots, z_{d+1}) \subset \text{conv}(Q)$. Hence for each choice of $q \in \text{ri}(\text{conv}(z_1, \ldots, z_{d+1}))$, and Q with size $\overline{n}$ $q \in \text{ri}(\text{conv}(Q))$. By Definition 4.1.2 q is n_f-safe, and the proof is complete. ■

Unfortunately, even if a point set is $(n_f + 1)$-divisible, it is not always the case that there exist $d + 1$ Tverberg points of depth $n_f + 1$ in general position. This can be seen in the example of Fig. 4.1(a), in which the set of Tverberg points is a 0-dimensional subset of $\mathbb{R}^2$ (*i.e.*, a single point). In such cases, the relative interior of the Tverberg points is empty, and a d-dimensional neighborhood of safe points does not exist. In such cases, Tverberg points may lie on the boundary, rather than in the interior, of the convex hull of fault free nodes. For example, in Fig. 4.1(a), if any vertex of the triangle corresponds to a faulty node, then the Tverberg point will be a vertex of the convex hull of the fault-free nodes, and not an interior point. This motivates the following definition.

Definition 4.1.3 ((r, k)-divisible point set [85])**.** *A set of n points in $\mathbb{R}^d$ is (r, k)-divisible if it can be partitioned into r pairwise disjoint subsets such that the intersection of the convex hulls of these r subsets is at least k-dimensional ($0 \leq k \leq d$).*

If a point set is $(n_f + 1, d)$-divisible, then there exists a set of $d + 1$ Tverberg points of depth $n_f + 1$ in general position, which allows the application of Proposition 4.1.1. Reay [85] and Roudneff [86, 87], have given conditions under which a point set is (r, k)-divisible.

Conjecture 4.1.1 (Reay's conjecture [85])**.** *A set of n points in general position in $\mathbb{R}^d$ (with $0 \leq k \leq d$) is (r, k)-divisible if $n \geq (d + 1)(r - 1) + k + 1$.*

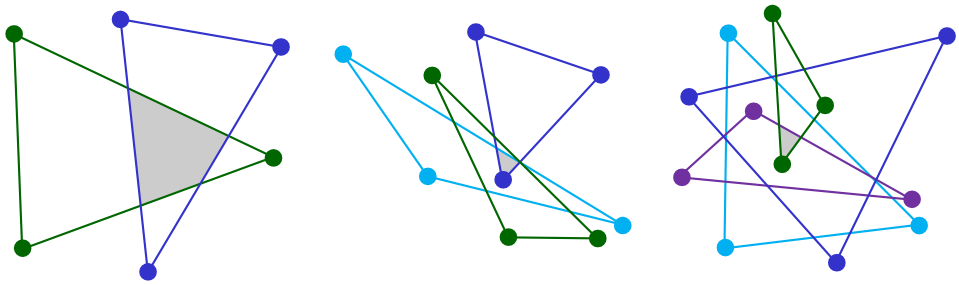
For the case of $k = d$, Reay's conjecture has been shown to be true for $2 \leq d \leq 8$ [85–88].

Proposition 4.1.2 (Birch [88] and Roudneff [86,87])**.** *For $d = 2, 3, \ldots, 8$, any set of n points in general position in $\mathbb{R}^d$ is (r, d)-divisible if $n \geq r(d + 1)$.*

This result allows us to apply our algorithm to robots with state spaces $\mathcal{X} \subseteq \mathbb{R}^d$ for $d \leq 8$, and provides the following sufficient condition for constructing a neighborhood of n_f-safe points.

Corollary 4.1.2 (Sufficient condition for n_f-safe neighborhood)**.** *For $d = 2, 3 \ldots, 8$, any set of $n \geq (n_f + 1)(d + 1)$ points in general position will have a Tverberg partition of depth $n_f + 1$, along with a set of $d + 1$ Tverberg points $z_1, \ldots z_{d+1}$ such that every $q \in \text{ri}(\text{conv}(\{z_1, \ldots, z_{d+1}\}))$ is n_f-safe.*

Three examples are shown in Fig. 4.2.



(a) 6 points, $r = 2$ (b) 9 points, $r = 3$ (c) 12 points, $r = 4$

Figure 4.2: Examples of $(r, 2)$-divisible points set in $\mathbb{R}^2$ (circle: position of nodes, shaded area: 2-dimensional intersection).

4.1.1 Computational Complexity and Approximations

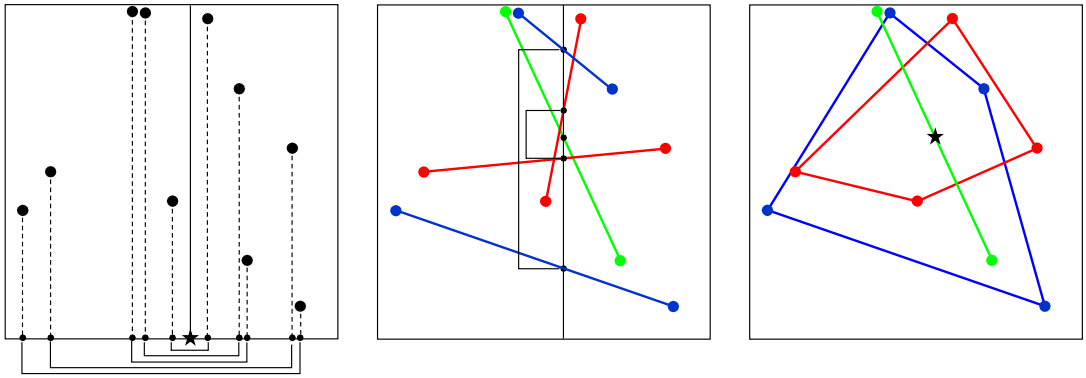
In general, the problem of computing Tverberg partitions is NP-Hard. Given n points in d dimensional space, the best known algorithm to obtain a Tverberg partition of depth $r = \lceil n/(d+1) \rceil$ requires up to $O(n^d)$ computation time.

A recent study [2] reports a *Lifting Algorithm* that computes an approximate[3] Tverberg partition of size $\lceil n/2^d \rceil$ and a sample Tverberg point in linear time in n and quasi-polynomial time in d, *i.e.*, $d^{O(1)}n$, under the condition that the n points are in general position[4]. The algorithm is a recursive projection-lifting algorithm. First, the point set in $\mathbb{R}^d$ is projected onto hyperplanes, H_m, of successively lower dimension, m, eventually onto the real line, $H_1 = \mathbb{R}^1$, and a partition (not a Tverberg partition) is constructed for this projection onto H_1. Then, for $m = 2, \ldots, d$ a partition for H_m is computed using a lifting method applied to the partition of H_{m-1}. The algorithm terminates when $m = d$, and the resulting partition is a Tverberg partition of depth $\lceil n/2^d \rceil$.

While a general description of the algorithm is beyond the scope of this chapter, it can easily be illustrated for the case of a point set in $\mathbb{R}^2$. Fig. 4.3 illustrates the procedure. In this case, there is only one step of projection (onto hyperplane $H_1 = \mathbb{R}^1$) and one lifting step. First, the n points are projected onto $\mathbb{R}^1$, and a partition is formed by creating 2-tuples of points that have successively increasing distance to the left and the right of the median. For the example shown in Fig. 4.3(a) the partition consists of five 2-tuples of points. Next, the line l through the median and perpendicular to $\mathbb{R}^1$ is constructed. For each 2-tuple in the partition of $\mathbb{R}^1$, the points are lifted back into $\mathbb{R}^2$, and

[3] *Approximate* in the sense that, given the same number of points, the algorithm obtains a Tverberg partition with decreased depth.

[4] The set of non-general configurations is measure zero, and thus any randomly drawn set of n points will be in general position with probability 1. Nevertheless, since our algorithms run on computers with finite precision arithmetic, the general position assumption must be verified in practice. When the general position assumption does not hold, small random perturbations may be applied to produce a point set in general configuration.



(a) (b) (c)

Figure 4.3: A procedure to obtain a Tverberg point by lifting method (the figure is inspired by that contained in [2]).

the intersection of l with the convex hull of the lifted points is computed, as shown in Fig. 4.3(b). The median of these intersection points is computed, and 2-tuples of convex hulls are constructed by establishing symmetric correspondences about this median (analogous to the process for H_1). In this example, since there were an odd number (five) of elements in the partition of H_1, we construct two 2-tuples, and one 1-tuple. These points that are included in corresponding 1- or 2-tuples define the Tverberg partition of the original point set, which is shown in Fig. 4.3(c). The depth of the resulting Tverberg partition is $\lceil n/2^d \rceil = 3$. Complete details of the lifting method can be found in [2].

4.2 A Fault-Tolerant Algorithm for Distributed Consensus

In this section, we introduce our *Approximate Distributed Robust Convergence* algorithm, which we call ADRC. The basic idea behind our algorithm is that each fault-free robot constructs $d+1$ maximal depth Tverberg points, uses these to define a safe point, and then executes a motion toward the safe point. The basic procedure is shown in Algorithm 1. We now present the details of the algorithm.

For each $i \in \overline{\mathcal{I}}$, define $\widetilde{n}_{f_i}(t)$ as the maximal value of n_{f_i} for which we can ensure the existence of an n_{f_i}-safe point with respect to the neighborhood $\mathcal{N}_i(t)$. From Corollary 4.1.2 we have

$$\widetilde{n}_{f_i}(t) \leq \frac{|\mathcal{N}_i(t)|}{d+1} - 1 \tag{4.2}$$

We use the *Lifting Algorithm* of Section 4.1.1 to construct a Tverberg partition of $\mathcal{N}_i(t)$ of depth

Algorithm 1: ADRC

Require: $\delta > 0$, $\{\{\mathcal{N}_i(t),\, \alpha_i(t)\}_{i\in\overline{\mathcal{I}}}\}_{t\in\mathbb{Z}_{\geq 0}}$
foreach *iteration* t **do**
  for *each fault-free robot* i **do**
    `// Look`
    Compute $\widetilde{n}_{f_i}(t)$
    `// Compute`
    Compute an $\widetilde{n}_{f_i}(t)$-safe point, $s_i(t)$
    $u_i(t) \leftarrow \alpha_i(t)(s_i(t) - x_i(t))$
    `// Move`
    if $\|u_i(t)\| < \delta$ **then**
      Halt
    else
      Execute control $u_i(t)$
    end
  end
end

$r = \lceil |\mathcal{N}_i(t)|/2^d \rceil$. This imposes the constraint that

$$\widetilde{n}_{f_i}(t) \leq \left\lceil \frac{|\mathcal{N}_i(t)|}{2^d} \right\rceil - 1 \tag{4.3}$$

Combining (4.2) and (4.3) we obtain:

$$\widetilde{n}_{f_i}(t) \leq \min\left\{ \left\lceil \frac{|\mathcal{N}_i(t)|}{2^d} \right\rceil, \frac{|\mathcal{N}_i(t)|}{d+1} \right\} - 1 \tag{4.4}$$

Proposition 4.2.1 (Existence of a safe point)**.** *For a node with neighbors $\mathcal{N}_i(t)$, the Lifting Algorithm will construct a Tverberg partition of depth $\widetilde{n}_{f_i}(t)+1$ and an $\widetilde{n}_{f_i}(t)$-safe point for*

$$\widetilde{n}_{f_i}(t) = \min\left\{ \left\lceil \frac{|\mathcal{N}_i(t)|}{2^d} \right\rceil, \left\lfloor \frac{|\mathcal{N}_i(t)|}{d+1} \right\rfloor \right\} - 1 \tag{4.5}$$

For the ith robot, our algorithm generates $d+1$ Tverberg points of depth $\widetilde{n}_{f_i}(t)+1$. This is done by invoking the lifting algorithm $d+1$ times, each time using a randomly chosen direction for the hyperplane projection steps. The center of mass of these $d+1$ points, $s_i(t)$, is an $\widetilde{n}_{f_i}(t)$-safe point. The basic procedure is shown in Algorithm 2. Fig. 4.4 illustrates the process for $d = 2$ and $|\mathcal{N}_i(t)| = 20$. Figs. 4.4(a)–(c) show the results of three applications of the lifting algorithm, each of which produces a Tverberg point of depth 5. Fig. 4.4(d) shows the resulting safe point s_i.

The action taken by the ith robot at time t is simply to move toward the safe point $s_i(t)$. This leads

Algorithm 2: Calculate $\widetilde{n}_{f_i}(t)$-Safe point

Require: $\mathcal{N}_i(t)$, d
$\widetilde{n}_{f_i}(t) \leftarrow \min\left\{\left\lceil \frac{|\mathcal{N}_i(t)|}{2^d} \right\rceil, \frac{|\mathcal{N}_i(t)|}{d+1}\right\} - 1$
$j \leftarrow 0$, $z_i[\,] \leftarrow \emptyset$
while $j \leq d+1$ **do**
  $\theta \leftarrow$`rand(`$1, d$`)`$\times 2\pi$
  $z_{ic} \leftarrow$ `tverbergPnt(`$\mathcal{N}_i(t), \widetilde{n}_{f_i}(t)+1, \theta$`)`
  `/* Calculate a Tverberg point of depth` $\widetilde{n}_{f_i}(t)+1$ `using the lifting algorithm with hyperplane at random angle` θ `*/`
  if `isGeneralPosition(`$z_i[\,], z_{ic}$`)` **then**
    $z_i[j] \leftarrow z_{ic}$
    $j \leftarrow j+1$
  end
end
$s_i(t) \leftarrow$ `mean` $(z_i[\,])$
return $s_i(t)$

to the following state update equation

$$x_i(t+1) = x_i(t) + u_i(t) \tag{4.6}$$

where

$$u_i(t) = \alpha_i(t)(s_i(t) - x_i(t)), \tag{4.7}$$

and $\alpha_i(t)$ is dynamically chosen parameter in the range, $0 < \alpha_{\min} \leq \alpha_i(t) \leq \alpha_{\max} < 1$, such that $u_i(t)$ does not violate constraints, *e.g.*, maximum allowable displacement per stage. It is possible to consider systems with more complex dynamics than those of (4.6), but we do not do so in this chapter.

Our algorithm is *aggressive*, in that it computes the maximal number of partitions possible, thus providing maximal fault-tolerance relative to the neighborhood size of each robot. This aggressiveness comes at a computational cost. We address this, and other computational issues, in Section 4.4.

Without loss of generality, throughout the remainder of the chapter we assume that robots evolving under ADRC are always in general position. If this is not the case, general position can be achieved by adding small perturbations to the positions of the robots.

4.3 Analysis of ADRC

In this section, we analyze the convergence of ADRC. The main result is given in Theorem 4.3.1, which establishes conditions under which ADRC will converge. Our analysis builds on previous work in distributed control of fault-free systems [31, 89] and in fault-tolerant computing [52, 53]. The former

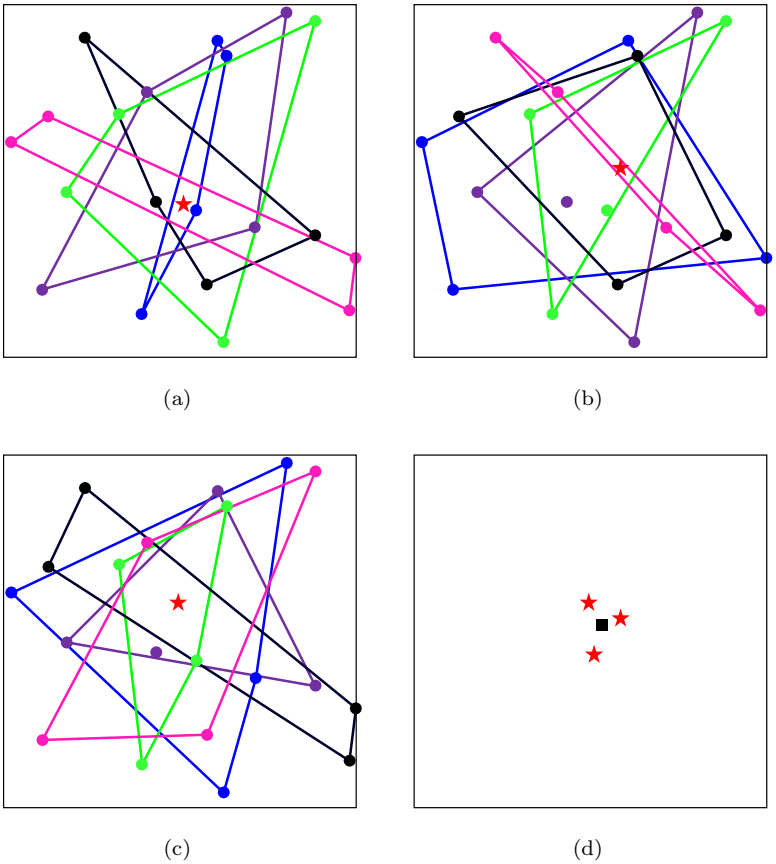


Figure 4.4: A safe point calculated by lifting method with 20 points. In (a)-(c), stars are Tverberg points and circles are positions of the nodes, and in (d) square symbol is the safe point, stars are Tverberg points obtained from (a)-(c).

is not concerned with cases in which individual nodes may fail, while the latter does not consider the case of time-varying network topology.

We begin by showing that under ADRC the evolution of the fault-free nodes can be described as a time-varying linear system that depends *only on the fault-free nodes*, and deriving certain properties of the corresponding time-varying system matrix. We then establish an appropriate concept of joint connectivity [31,89], which we call *repeated reachability*. The concept will be used to provide a minimally restrictive condition on the connectivity graph of the fault-free nodes for convergence under ADRC. Finally, we employ properties of stochastic matrices [90–93] to demonstrate convergence of ADRC.

4.3.1 Evolution of the fault-free nodes as an LTV system

The behavior of the fault-free robots executing ADRC can be described as a discrete-time linear time-varying system. In particular, the system evolution can be simply expressed as a backward product of non-homogeneous system matrices. Our approach closely follows that given in [52].

Proposition 4.3.1. *For an F-MRS in which the fault-free nodes execute ADRC, if $n_{f_i}(t) \leq \widetilde{n}_{f_i}(t)$ for each $i \in \overline{\mathcal{I}}$, then the evolution of the fault-free nodes, $\overline{\mathbf{x}}(t)$, can be represented by an LTV system of the form*

$$\overline{\mathbf{x}}(t+1) = \mathbf{M}(t)\overline{\mathbf{x}}(t), \;\; t = 0,\, 1,\, 2, \ldots, \tag{4.8}$$

in which $\overline{\mathbf{x}}(t)$ is an $\overline{n} \times d$ matrix, and $\mathbf{M}(t)$ is an $\overline{n} \times \overline{n}$ row-stochastic matrix with $[\mathbf{M}]_{ij}(t) > 0$ for $i = j$, or $j \in \overline{\mathcal{N}}_i(t)$.

Proof. Let $\overline{\mathcal{Y}}_i := \{x_j\}_{j \in \overline{\mathcal{N}}_i}$ denote the set of positions of the fault-free neighbors of node i. By Proposition 4.2.1, ADRC constructs an $n_{f_i}(t)$-safe point $s_i(t)$ for each $i \in \overline{\mathcal{I}}$. Thus, there exists some set of fault-free neighbors with positions $P \subseteq \overline{\mathcal{Y}}_i$ such that

$$s_i(t) \in \mathrm{ri}(\mathrm{conv}(P)) \subseteq \mathrm{ri}(\mathrm{conv}(\overline{\mathcal{Y}}_i(t)),$$

Thus (see Proposition C.0.1), for each $i \in \overline{\mathcal{I}}$, there is a set of non-zero weights $\{\lambda_{ij}\}_{j \in \overline{\mathcal{N}}_i(t)}$ such that $s_i(t)$ can be represented by

$$s_i(t) = \sum_{j \in \overline{\mathcal{N}}_i(t)} \lambda_{ij}(t) x_j(t) \tag{4.9}$$

with $\sum_{j \in \overline{\mathcal{N}}_i(t)} \lambda_{ij}(t) = 1$, and $\lambda_{ij}(t) > 0$ for all $j \in \overline{\mathcal{N}}_i(t)$.

Plugging (4.9) into (4.6) and (4.7), for each $i \in \overline{\mathcal{I}}$, we obtain

$$x_i(t+1) = (1-\alpha_i(t))x_i(t) + \alpha_i(t)\left(\sum_{j \in \overline{\mathcal{N}}_i(t)} \lambda_{ij}(t)x_j(t)\}\right).$$

There is such an equation for each $i \in \overline{\mathcal{I}}$, and these may be combined to obtain (4.8) by defining the $\overline{n} \times \overline{n}$ matrix $\mathbf{M}(t)$ as follows. For $i, j \in \overline{\mathcal{I}}$

$$\mathbf{M}(t) = \begin{cases} 1-\alpha_i(t) & \text{if } j = i \\ \alpha_i(t)\lambda_{ij}(t) & \text{if } j \neq i \text{ and } j \in \overline{\mathcal{N}}_i(t) \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\mathbf{M}(t)$ is row-stochastic, with diagonal elements $[\mathbf{M}]_{ii}(t) = 1-\alpha_i(t) \geq 1-\alpha_{\max}$, and $[\mathbf{M}]_{ij}(t) = \alpha_i(t)\lambda_{ij}(t)$ for $i \neq j$ if $j \in \overline{\mathcal{N}}_i(t)$.

■

Using (4.8), we may define state transition matrix $\Phi(t_F, t_I)$, for t_I, $t_F \in \mathbb{Z}_{\geq 0}$ where $t_I \leq t_F$ using a *backward product* of system matrices

$$\begin{aligned} \Phi(t_F, t_I) &= \mathbf{M}(t_F)\mathbf{M}(t_F-1)\ldots\mathbf{M}(t_I) \\ &= \prod_{t=t_I}^{t_F} \mathbf{M}(t). \end{aligned} \tag{4.10}$$

4.3.2 Jointly Reachable Graphs

In this section, we define the concept of *joint reachability*, which is analogous to the concept of joint connectivity introduced in [31, 89]. For a jointly reachable sequence of graphs, we give a relationship between the adjacency matrix for the union of the graphs and the adjacency matrices for the individual graphs in the sequence. We then extend the concept of joint reachability to infinite sequences of graphs by defining the concept of *repeated reachability*, which will play a key role to establish a minimally restrictive condition in Theorem 4.3.1.

We denote by $\overline{\mathcal{G}}(t)$ the connectivity graph of the fault-free nodes at time t, and by $\overline{\mathcal{A}}(t)$ the corresponding adjacency matrix.

Definition 4.3.1 (Jointly reachable sequence of graphs)**.** *For $j \in \mathbb{N}$, consider a sequence of graphs*

$$\overline{\mathcal{G}}(T_j), \ldots, \overline{\mathcal{G}}(T_{j+1} - 1)$$

of length $L_j = T_{j+1} - T_j$, and with common vertex set $\overline{\mathcal{V}}$, such that $\overline{\mathcal{G}}(t) = (\overline{\mathcal{V}}, \overline{\mathcal{E}}(t))$ for $t = T_j, \ldots, T_{j+1} - 1$. The union of these graphs is defined by

$$\widetilde{\mathcal{G}}_{T_j, T_{j+1}-1} = \bigcup_{t=T_j}^{T_{j+1}-1} \overline{\mathcal{G}}(t) = \left(\overline{\mathcal{V}}, \bigcup_{t=T_j}^{T_{j+1}-1} \overline{\mathcal{E}}(t) \right).$$

We say that the sequence is jointly reachable if there exists some $v \in \overline{\mathcal{V}}$ such that for each $v' \neq v$ in $\overline{\mathcal{V}}$ there exists a path from v' to v in $\widetilde{\mathcal{G}}_{T_j, T_{j+1}-1}$.

For a jointly connected sequence of graphs $\overline{\mathcal{G}}(T_j), \ldots, \overline{\mathcal{G}}(T_{j+1} - 1)$, we denote by $\widetilde{\mathcal{A}}_j$ the adjacency matrix for $\widetilde{\mathcal{G}}_{T_j, T_{j+1}-1}$. The following lemma provides a useful relationship between $\widetilde{\mathcal{A}}_j$ and the individual adjacency matrices $\overline{\mathcal{A}}(t)$, $T_j \leq t \leq T_{j+1} - 1$.

Lemma 4.3.1. *For $\overline{\mathcal{A}}(T_j), \ldots, \overline{\mathcal{A}}(T_{j+1}-1)$ adjacency matrices for a sequence of graphs $\overline{\mathcal{G}}(T_j), \ldots, \overline{\mathcal{G}}(T_{j+1}-1)$, with the adjacency matrix $\widetilde{\mathcal{A}}_j$ for $\widetilde{\mathcal{G}}_{T_j, T_{j+1}-1}$, the following inequality holds:*

$$\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_j \leq \prod_{t=T_j}^{T_{j+1}-1} (\mathbf{I}_{\overline{n}} + \overline{\mathcal{A}}(t)). \tag{4.11}$$

Proof. Since $\widetilde{\mathcal{G}}_{T_j, T_{j+1}-1}$ is the *union* graph of the finite graph sequence $\{\overline{\mathcal{G}}(t)\}_{T_j}^{T_{j+1}-1}$,

$$\widetilde{\mathcal{E}}_{T_j, T_{j+1}-1} = \bigcup_{t \in [T_j,\, T_{j+1}-1]} \overline{\mathcal{E}}(t),$$

which implies

$$\widetilde{\mathcal{A}}_j = \widetilde{\mathcal{A}}_{T_j, T_{j+1}-1} \leq \sum_{t=T_j}^{T_{j+1}-1} \overline{\mathcal{A}}(t). \tag{4.12}$$

Simple calculation yields

$$\mathbf{I}_{\overline{n}} + \sum_{t=T_j}^{T_{j+1}-1} \overline{\mathcal{A}}(t) \leq \prod_{t=T_j}^{T_{j+1}-1} (\mathbf{I}_{\overline{n}} + \overline{\mathcal{A}}(t)). \tag{4.13}$$

Now, we can combine the two inequalities (4.12) and (4.13) to obtain:

$$\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_j \leq \prod_{t=T_j}^{T_{j+1}-1} (\mathbf{I}_{\overline{n}} + \overline{\mathcal{A}}(t)).$$

■

The following related result holds for any sequence of adjacency matrices. This includes $\overline{\mathcal{A}}(t), \widetilde{\mathcal{A}}_j$ or a general adjacency matrix $\mathcal{A}_j$.

Lemma 4.3.2. *Let $\mathcal{A}_0, \mathcal{A}_1, \ldots$ be a sequence of $\overline{n} \times \overline{n}$ adjacency matrices and let $\eta = (\overline{n}-1)2^{\overline{n}(\overline{n}-1)}$. Then for any $l = 0, 1, \ldots$ we have*

$$(\mathbf{I}_{\overline{n}} + \mathcal{A}_{l^*})^{\overline{n}-1} \leq \prod_{j=l}^{l+\eta-1} (\mathbf{I}_{\overline{n}} + \mathcal{A}_j)$$

for $l^* \in \mathbb{Z}_{\geq 0}$ such that the term $(\mathbf{I}_{\overline{n}} + \mathcal{A}_{l^*})$ appears at least $\overline{n}-1$ times in the right hand side.

Proof. Since there are at most $2^{\overline{n}(\overline{n}-1)}$ possible adjacency matrices, in any set of $\eta = (\overline{n}-1)2^{\overline{n}(\overline{n}-1)}$ adjacency matrices, at least one matrix, $\mathcal{A}_{l^*}$ must appear at least $\overline{n}-1$ times. The inequality then follows from the fact that $\mathcal{A}_j$ consists of nonnegative entries. ■

Joint reachability is a property of finite-length sequences of graphs. To establish conditions for convergence of ADRC, we extend this notion to infinite graph sequences with the following definition, which is similar to the concept of *repeatedly jointly rooted graph sequence* previously defined in [89].

Definition 4.3.2 (Repeatedly reachable graph sequence)**.** *An infinite sequence of graphs $\overline{\mathcal{G}}(0), \overline{\mathcal{G}}(1), \overline{\mathcal{G}}(2) \ldots$ is said to be* repeatedly reachable *if there exists a sequence of times $0 = T_1 < T_2 \ldots$ such that $T_{j+1} - T_j = L_j < \infty$ and the subsequence $\overline{\mathcal{G}}(T_j),\ \overline{\mathcal{G}}(T_j+1),\ \ldots,\ \overline{\mathcal{G}}(T_{j+1}-1)$ is jointly reachable for all j.*

We denote by $L_{\max}$ the least uniform upper bound for all L_j, i.e.*, $L_j \leq L_{\max}$, for all $j \in \mathbb{N}$.*

In other words, the sequence $\overline{\mathcal{G}}(0), \overline{\mathcal{G}}(1), \overline{\mathcal{G}}(2) \ldots$ is repeatedly reachable if it can be partitioned into contiguous finite length subsequences of lengths $L_j \leq L_{\max}$ that are themselves jointly reachable. Note that the condition "the sequence $\overline{\mathcal{G}}(0), \overline{\mathcal{G}}(1), \overline{\mathcal{G}}(2) \ldots$ is repeatedly reachable" is significantly less restrictive than the condition " $\overline{\mathcal{G}}(t)$ strongly connected for all $t \in \mathbb{Z}_{\geq 0}$".

4.3.3 Convergence of ADRC

In this section, we provide the main theoretical result of the chapter. We begin by using properties of jointly reachable graphs to infer a lower bound on the backward product of system matrices (Proposition 4.3.2 whose proof relies on Lemma 4.3.3). Then, using Proposition 4.3.2, we derive a bound on the coefficient of ergodicity for a finite backward product of system matrices (Lemma 4.3.4). Finally, we present the main result in Theorem 4.3.1.

Lemma 4.3.3. *For an F-MRS in which the fault-free nodes execute ADRC, and for a jointly reachable sequence of graphs $\overline{\mathcal{G}}(T_j), \ldots, \overline{\mathcal{G}}(T_{j+1}-1)$ with $\widetilde{\mathcal{A}}_j$ the adjacency graph for $\widetilde{\mathcal{G}}_{T_j,T_{j+1}-1}$, if $n_{f_i}(t) \leq \widetilde{n}_{f_i}(t)$ for all $i \in \overline{\mathcal{I}}$, for each $j \in \mathbb{N}$ there exists $\gamma_j \in (0,\, 1)$ such that*

$$\gamma_j^{L_j}(\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_j) \leq \prod_{t=T_j}^{T_{j+1}-1} \mathbf{M}(t) \tag{4.14}$$

where $L_j = T_{j+1} - T_j$.

Proof. By Proposition 4.3.1, $[\mathbf{M}(t)]_{ij} > 0$, for $i = j$ or $[\overline{\mathcal{A}}(t)]_{ij} \neq 0$, and thus, for each t there exists $\gamma(t) \in (0,\, 1)$ such that

$$\gamma(t)(\mathbf{I}_{\overline{n}} + \overline{\mathcal{A}}(t)) \leq \mathbf{M}(t).$$

Taking the product for each side over the sequence we obtain

$$\prod_{t=T_j}^{T_{j+1}-1} \gamma(t)(\mathbf{I}_{\overline{n}} + \overline{\mathcal{A}}(t)) \leq \prod_{t=T_j}^{T_{j+1}-1} \mathbf{M}(t). \tag{4.15}$$

Let γ_j be a lower bound on $\gamma(t)$ for the interval T_j to $T_{j+1}-1$. Then

$$\gamma_j^{L_j} \prod_{t=T_j}^{T_{j+1}-1} (\mathbf{I}_{\overline{n}} + \overline{\mathcal{A}}(t)) \leq \prod_{t=T_j}^{T_{j+1}-1} \gamma(t)(\mathbf{I}_{\overline{n}} + \overline{\mathcal{A}}(t)). \tag{4.16}$$

Finally, Combining (4.15) and (4.16) and applying Lemma 4.3.1 we obtain

$$\gamma_j^{L_j}(\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_j) \leq \prod_{t=T_j}^{T_{j+1}-1} \mathbf{M}(t).$$

■

Proposition 4.3.2. *For an F-MRS in which the fault-free nodes execute ADRC, if*

(a) $n_{f_i}(t) \leq \widetilde{n}_{f_i}(t)$ *for each* $i \in \overline{\mathcal{I}}$ *and* $t \in \mathbb{Z}_{\geq 0}$*, and*

(b) the sequence of connectivity graphs for the fault-free nodes, $\overline{\mathcal{G}}(0), \overline{\mathcal{G}}(1), \overline{\mathcal{G}}(2), \ldots$*, is repeatedly reachable*

then for each $l \in \mathbb{Z}_{\geq 1}$*, there exists* $\gamma \in (0,\, 1)$*, and* $i \in \overline{\mathcal{I}}$ *such that*

$$\gamma^{T_{l+\eta}-T_l} \mathbf{1}_{\overline{n}\times 1} \leq \left[\prod_{t=T_l}^{T_{l+\eta}-1} \mathbf{M}(t) \right]_i \tag{4.17}$$

in which $0 = T_0 < T_1 < T_2 \ldots$ *is a sequence of times such that* $T_{j+1} - T_j = L_j < \infty$ *and* $\overline{\mathcal{G}}(T_j), \ldots, \overline{\mathcal{G}}(T_{j+1}-1)$ *is jointly reachable for all* j*; and* $\eta = (\overline{n}-1)2^{\overline{n}(\overline{n}-1)}$*.*

Proof. By Lemma 4.3.3 there exists $\gamma_j \in (0,\, 1)$ such that

$$\gamma_j^{L_j}(\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_j) \leq \prod_{t=T_j}^{T_{j+1}-1} \mathbf{M}(t), \tag{4.18}$$

for $L_j = T_{j+1} - T_j$.

We may compute the product of each side over the interval from $j = l$ to $j = l + \eta - 1$,

$$\prod_{j=l}^{l+\eta-1} \gamma_j^{L_j}(\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_j) \leq \prod_{j=l}^{l+\eta-1} \prod_{t=T_j}^{T_{j+1}-1} \mathbf{M}(t). \tag{4.19}$$

Now, let γ be a uniform lower bound for the γ_j, then

$$\prod_{j=l}^{l+\eta-1} \gamma^{L_j}(\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_j) \leq \prod_{j=l}^{l+\eta-1} \gamma_j^{L_j}(\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_j). \tag{4.20}$$

Since $L_j = T_{j+1} - T_j$, simple calculations yield

$$\sum_{j=l}^{l+\eta-1} L_j = \sum_{j=l}^{l+\eta-1} T_{j+1} - T_j = T_{l+\eta} - T_l.$$

Combining this result with (4.19) and (4.20) we obtain

$$\gamma^{T_{l+\eta}-T_l} \prod_{j=l}^{l+\eta-1} (\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_j) \leq \prod_{t=T_l}^{T_{l+\eta}-1} \mathbf{M}(t). \tag{4.21}$$

We now apply Lemma 4.3.2. Let l^* be such that the term $(\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_{l^*})$ appears at least $\overline{n} - 1$ times in

the product on the left hand side of (4.21). Then (by Lemma 4.3.2)

$$\left(\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_{l^*}\right)^{\overline{n}-1} \leq \prod_{j=l}^{l+\eta-1} \left(\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_j\right),$$

and thus

$$\gamma^{T_{l+\eta}-T_l}\left(\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_{l^*}\right)^{\overline{n}-1} \leq \gamma^{T_{l+\eta}-T_l} \prod_{j=l}^{l+\eta-1} (\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_j). \tag{4.22}$$

But the matrix $\widetilde{\mathcal{A}}_{l^*}$ is an adjacency matrix for a graph with a globally reachable node, and thus, using Lemma C.0.3, there is some i such that

$$\gamma^{T_{l+\eta}-T_l}\mathbf{1}_{\overline{n}\times 1} \quad \leq \quad \gamma^{T_{l+\eta}-T_l}\left[\left(\mathbf{I}_{\overline{n}} + \widetilde{\mathcal{A}}_{l^*}\right)^{\overline{n}-1}\right]_i. \tag{4.23}$$

Finally, combining (4.21), (4.22), and (4.23), we obtain

$$\gamma^{T_{l+\eta}-T_l}\mathbf{1}_{\overline{n}\times 1} \leq \left[\prod_{t=T_l}^{T_{l+\eta}-1} \mathbf{M}(t)\right]_i.$$

■

The following lemma provides a bound on the coefficient of ergodicity, τ_1, for a finite backward product of system matrices evolving under ADRC. A brief review of stochastic matrices and ergodicity is provided in Appendix B. A more comprehensive review can be found in [90, 91].

Lemma 4.3.4. *Let $\tau_1(\mathbf{S})$ denote the coefficient of ergodicity for a stochastic matrix $\mathbf{S}$. For an F-MRS in which the fault-free nodes execute ADRC, if*

(a) $n_{f_i}(t) \leq \widetilde{n}_{f_i}(t)$ for each $i \in \overline{\mathcal{I}}$ and $t \in \mathbb{Z}_{\geq 0}$, and

(b) the sequence of connectivity graphs for the fault-free nodes, $\overline{\mathcal{G}}(0), \overline{\mathcal{G}}(1), \overline{\mathcal{G}}(2), \ldots$, is repeatedly reachable

then the following inequality holds:

$$\prod_{h=0}^{v} \tau_1\left(\prod_{t=T_{h\eta}}^{T_{(h+1)\eta}-1} \mathbf{M}(t)\right) \leq (1-\gamma^{L_{\max}\eta})^v, \tag{4.24}$$

in which $0 = T_0 < T_1 < T_2 \ldots$ is a sequence of times such that $T_{j+1} - T_j = L_j < \infty$ and $\overline{\mathcal{G}}(T_j), \ldots, \overline{\mathcal{G}}(T_{j+1}-1)$ is jointly reachable for all j; and $\eta = (\overline{n}-1)2^{\overline{n}(\overline{n}-1)}$.

Proof. Proposition B.0.1 together with Proposition 4.3.2 imply

$$\tau_1\left(\prod_{t=T_{h\eta}}^{T_{(h+1)\eta}-1}\mathbf{M}(t)\right)\leq 1-\gamma^{T_{(h+1)\eta}-T_{h\eta}} \tag{4.25}$$

which holds for all $h\in\mathbb{Z}_{\geq 0}$. Let $L_{\max}\in\mathbb{N}$ be a uniform upper bound for the L_j. Thus, for all $h\in\mathbb{Z}_{\geq 0}$

$$1-\gamma^{T_{(h+1)\eta}-T_{h\eta}}\leq 1-\gamma^{L_{\max}\eta}. \tag{4.26}$$

Combining (4.25) and (4.26), and computing the product of each side for $h=0$ to $h=v$ for some v, we obtain,

$$\prod_{h=0}^{v}\tau_1\left(\prod_{t=T_{h\eta}}^{T_{(h+1)\eta}-1}\mathbf{M}(t)\right)\leq(1-\gamma^{L_{\max}\eta})^v.$$

Note that the values taken by t in the left hand side range from $t=0$ when $h=0$ to $t=T_{(v+1)\eta}-1$ for $h=v$. ■

We now state the main theorem of the chapter, which guarantees that under certain connectivity conditions, the fault-free robots executing ADRC will converge to within any desired ϵ bound, regardless of the behavior of the faulty robots. The proof of the theorem provides a bound on the time required to achieve ϵ-convergence.

Theorem 4.3.1 (Convergence of ADRC)**.** *For an F-MRS in which the fault-free nodes execute ADRC, if*

(a) $n_{f_i}(t)\leq\widetilde{n}_{f_i}(t)$ *for each* $i\in\overline{\mathcal{I}}$ *and* $t\in\mathbb{Z}_{\geq 0}$*, and*

(b) the sequence of connectivity graphs for the fault-free nodes, $\overline{\mathcal{G}}(0),\overline{\mathcal{G}}(1),\overline{\mathcal{G}}(2),\ldots$*, is repeatedly reachable*

then for every $t\geq 0$*, fault-free pair* $i,j\in\overline{\mathcal{I}}$*, and* $\epsilon>0$*, there is some* $t_\epsilon>0$ *such that* $\|x_i(t)-x_j(t)\|<\epsilon$ *for all* $t>t_\epsilon$*.*

Proof. Our proof, which uses the ergodicity of a backward product of stochastic matrices, is directly inspired by the results given in [32, 52], and relies on classical results from [94].

Let $0=T_0<T_1<T_2\ldots$ denote a sequence of times such that $T_{j+1}-T_j=L_j<\infty$, and $\overline{\mathcal{G}}(T_j),\ldots,\overline{\mathcal{G}}(T_{j+1}-1)$ is jointly reachable for all j. Such a sequence exists, since $\overline{\mathcal{G}}(0),\overline{\mathcal{G}}(1),\overline{\mathcal{G}}(2),\ldots$ is repeatedly reachable. For an arbitrary time $t\geq 0$, let $v^\star$ be the largest non-negative integer which satisfies $T_{(v^\star+1)\eta}-1\leq t$, and $t^\star=T_{(v^\star+1)\eta}-1$, with $\eta=(\overline{n}-1)2^{\overline{n}(\overline{n}-1)}$.

We can express the system evolution of fault-free robots using state transition matrix Φ given in (4.10) as

$$\overline{\mathbf{x}}(t) = \Phi(t,\, 0)\overline{\mathbf{x}}(0) \tag{4.27}$$

$$= \Phi(t,\, T_{(v^\star+1)\eta})\Phi(T_{(v^\star+1)\eta} - 1,\, 0)\overline{\mathbf{x}}(0) \tag{4.28}$$

$$= \Phi(t,\, t^\star)\underbrace{\Phi(t^\star,\, 0)\overline{\mathbf{x}}(0)}_{\overline{\mathbf{x}}(t^\star)}. \tag{4.29}$$

Now, for each time $t \geq 0$, there are two possibilities:

1. $t = t^\star = T_{(v^\star+1)\eta} - 1$ for some $v^\star$, or
2. $T_{(v^\star+1)\eta} < t < T_{(v^\star+1)\eta} - 1$, for some $v^\star$.

We will first consider the case when $t = t^\star$, and evaluate the maximum difference of the rows of $\Phi(t^\star,\, 0)$ to provide a uniform upper bound for the Euclidean distance between the positions of any fault-free pair at $t^\star$. Then using the contracting property of the map Φ, we will show that the maximum Euclidean distance between fault-free pairs is non-increasing for $T_{(v^\star+1)\eta} < t < T_{(v^\star+1)\eta} - 1$, for any $v^\star$.

Case 1: $t = t^\star$

For notational convenience, we denote by x_i^l the lth coordinate of x_i, and we define $q_{ij}(t) := [\Phi(t,\, 0)]_{ij}$ as the (i, j)th entry of the matrix product $\Phi(t, 0)$. We denote by α_l the absolute value of the largest lth component of the initial position of all robots

$$\alpha_l = \max_{m \in \overline{\mathcal{I}}} |x_m^l(0)|$$

Consider the difference in lth coordinate of the positions of any two fault-free robots $i,\, j \in \overline{\mathcal{I}}$ at time $t^\star > 0$

$$
\begin{aligned}
|x_i^l(t^\star) - x_j^l(t^\star)| = & \left| \sum_{g=1}^{\overline{n}} (q_{ig}(t^\star) - q_{jg}(t^\star)) x_g^l(0) \right| \\
\leq & \sum_{g=1}^{\overline{n}} \left| (q_{ig}(t^\star) - q_{jg}(t^\star)) x_g^l(0) \right| && (4.30) \\
\leq & \sum_{g=1}^{\overline{n}} |(q_{ig}(t^\star) - q_{jg}(t^\star))| \left| x_g^l(0) \right| && (4.31) \\
\leq & \alpha_l \sum_{g=1}^{\overline{n}} |q_{ig}(t^\star) - q_{jg}(t^\star)| && (4.32) \\
\leq & \alpha_l \overline{n}\, \delta \left(\prod_{t=0}^{t^\star} \mathbf{M}(t) \right) && (4.33) \\
= & \alpha_l \overline{n}\, \delta \left(\prod_{t=0}^{T_{(v^\star+1)\eta}-1} \mathbf{M}(t) \right) && (4.34) \\
= & \alpha_l \overline{n}\, \delta \left(\prod_{h=0}^{v^\star} \prod_{t=T_{h\eta}}^{T_{(h+1)\eta}-1} \mathbf{M}(t) \right) && (4.35) \\
\leq & \alpha_l \overline{n} \prod_{h=0}^{v^\star} \tau_1 \left(\prod_{t=T_{h\eta}}^{T_{(h+1)\eta}-1} \mathbf{M}(t) \right) && (4.36) \\
\leq & \alpha_l \overline{n} (1 - \gamma^{L_{\max}\eta})^{v^\star} && (4.37)
\end{aligned}
$$

In the steps above (4.30) follows from the triangle inequality, (4.31) follows from the Cauchy-Schwartz inequality; (4.32) uses the definition of α_l; (4.33) uses the definition of maximum range given by (B.3); (4.34) is obtained using $t^\star = T_{(v^\star+1)\eta} - 1$; (4.36) follows from Lemma B.0.2; (4.37) follows from (6.7) of lemma 4.3.4.

Using (4.37), we can compute a bound on the distance between the positions of any two robots at time $t^\star$:

$$
\begin{aligned}
\|x_i(t^\star) - x_j(t^\star)\|^2 &= \sum_{l=1}^{d} \left| x_i^l(t^\star) - x_j^l(t^\star) \right|^2 && (4.38) \\
&\leq \sum_{l=1}^{d} \alpha_l^2 \overline{n}^2 (1 - \gamma^{L_{\max}\eta})^{2v^\star}. && (4.39)
\end{aligned}
$$

Setting the upper bound to ϵ, *i.e.*, setting

$$\sum_{l=1}^{d} \alpha_l^2 \overline{n}^2 (1-\gamma^{L_{\max}\eta})^{2v^\star} \leq \epsilon^2$$

and solving for $v^\star$ yields

$$v^* \leq \frac{\log \epsilon - \log \overline{n} - \frac{1}{2}\log \sum \alpha_l^2}{\log\left(1-\gamma^{L_{\max}\eta}\right)}. \tag{4.40}$$

And we can obtain the actual time $t^\star$ at the switching step $v^\star$ using

$$t^\star = T_{(v^\star+1)\eta} - 1.$$

We note in (4.39) that as the switching time $v^\star \to \infty$ the left hand side will tend to 0. We now consider the case when t is *not* a switching time.

Case 2: $T_{(v^\star+1)\eta} < t < T_{(v^\star+1)\eta} - 1$

For a given configuration $\overline{x}(t)$ of the fault-free nodes, we define the *diameter* of $\overline{x}(t)$ as

$$\operatorname{diam}(\overline{x}(t)) := \max_{i,j\in\overline{\mathcal{I}}} \|x_i(t) - x_j(t)\|. \tag{4.41}$$

For a switching time $t = t^\star$, (4.39) provides a uniform upper bound on the diameter of the positions of the fault-free nodes:

$$\operatorname{diam}(\overline{x}(t^\star)) \leq \overline{n}(1-\gamma^{L_{\max}\eta})^{v^\star} \left(\sum_{l=1}^{d} \alpha_l^2\right)^{\frac{1}{2}}.$$

For $t^\star = T_{(v^\star+1)\eta} < t < T_{(v^\star+1)\eta} - 1$, if the fault-free robots execute ADRC, we can apply (4.29) to obtain

$$\operatorname{conv}(\overline{x}(t)) = \operatorname{conv}(\Phi(t,\, t^\star)\overline{x}(t^\star)) \subset \operatorname{conv}(\overline{x}(t^\star))$$

and this implies

$$\operatorname{diam}(\overline{x}(t)) \leq \operatorname{diam}(\overline{x}(t^\star)).$$

Thus, the uniform upper-bound obtained for Euclidean distance between all fault-free pairs at switching time $t^\star$ is also a valid for all $t \geq t^\star$. However it is not known whether for some arbitrary pair $i,\, j \in \overline{\mathcal{I}}$, $\|x_i(t) - x_j(t)\| \leq \|x_i(t^\star) - x_j(t^\star)\|$ will hold for $t > t^\star$.

Combining the results for Case 1 and Case 2 above, we have shown that for every $\epsilon > 0$, and for all pairs i, $j \in \overline{\mathcal{I}}$, there is $t^\star > 0$ such that $t > t^\star$ implies $\|x_i(t) - x_j(t)\| < \epsilon$. ■

Remark: To this point, our proof only demonstrates ergodicity in the *weak* sense, which, on its own, does not imply that the positions of the fault-free robots will converge to a point that is stationary. However, it has been shown by [92] that backward products of row-stochastic matrices have a nice property that is summarized in the following theorem:

Theorem 4.3.2 (Chatterjee and Seneta [93])**.** *For backward product of stochastic matrices, weak and strong ergodicity are equivalent.*

Using Theorem 4.3.2, we can deduce the following corollary, which implies convergence to a fixed point.

Corollary 4.3.1 (Convergence of ADRC to a fixed point)**.** *Consider the assumptions and settings given by Theorem 4.3.1. Then there is $p \in Q \subseteq \mathbb{R}^d$ such that for all $i \in \overline{\mathcal{I}}$, $x_i(t) \to p$ as $t \to \infty$.*

4.3.4 Comments on the weak ergodicity of ADRC

The paper [90], discusses various classes of matrices that show ergodic properties. Stochastic, Indecomposable, Aperiodic (SIA) matrices comprise the largest of these classes. Roughly speaking, SIA matrices are *regular* matrices[5], and an infinite product of SIA matrices tends to a matrix with identical rows. The set of *scrambling matrice*[6] is a subset of the SIA matrices, with the nice feature that multiplying any two scrambling matrices produces a matrix that is also scrambling. This is contrast to SIA matrices: multiplying two SIA matrices does not necessarily produce an SIA matrix. The following relationships hold between matrix classes that are used in this chapter:

$$\{\text{Positive matrices}\} \subset \{\text{Matrices with a positive column}\} \subset \{\text{Scrambling matrices}\} \subset \{\text{SIA}\}.$$

Recall that Proposition 4.3.2 states that the infinite backward product of system matrices generated by ADRC can be expressed as an infinite backward product of matrices with positive columns. By the inclusion above, these are in fact scrambling matrices. Thus, the weak ergodicity of ADRC follows

[5] A row-stochastic matrix is *regular* if it has a unit eigenvalue, i.e, the eigenvalue $\lambda = 1$ is simple. The powers of every regular matrix converge to a rank one row-stochastic matrix.

[6] A row-stochastic matrix is *scrambling* if and only if any two rows have at least one positive element in a coincident position.

by the classical results contained in, *e.g.*, [90, 91] which asserts that *the infinite product of scrambling matrices is weakly ergodic.*

4.4 A Family of ADRC Algorithms

Like all distributed control algorithms, convergence of ADRC relies on maintaining appropriate connectivity conditions; in the case of ADRC, repeated reachability of the connectivity graphs of the fault-free nodes is a condition of Theorem 4.3.1. In this section, we propose three versions of ADRC, each with its own strategy for maintaining adequate connectivity.

In the case of fault-free networks in which connectivity is determined by the sensing capabilities of each robot, it is often possible to enforce connectivity constraints by limiting the range of motion of each robot, based on the locations of its neighbors (*e.g.*, [23, 35]). The *circumcenter algorithm* [1, 23] is one such approach. Unfortunately, these approaches cannot be applied in cases when some of the robots are faulty; if a fault-free robot constrains its motion in order to enforce connectivity with a faulty neighbor, then the faulty neighbor has the possibility to impede, or even prevent, convergence of the fault-free nodes by its actions. For example, if connectivity is enforced, and if there is a faulty robot that moves far away, fault-free robots that are initially connected to the faulty robot may have no choice but to follow the faulty robot (in order to maintain connectivity), which may result in a *partitioned connectivity graph* or *divergence* of robot positions.

Rather than constraining the motions of the robots, we have opted to design a class of algorithms that employ variable-range sensing to maintain connectivity. This approach is motivated by work in wireless sensor networks (WSNs) [95, 96], where sensor nodes are capable of adjusting their sensing ranges to conserve energy. In particular, we assume that each fault-free robot can dynamically adjust its sensing range, resulting in a trade-off between cost of energy for sensing and connectivity maintenance.

Our three algorithms, ADRC-I, ADRC-II, ADRC-III, are defined as follows:

ADRC-I Each robot has a fixed sensing range, r_i. This is the typical case that is considered in distributed control algorithms for fault-free networks.

ADRC-II At time $t+1$, the ith robot chooses its sensing range $r_i(t+1)$ so that $\mathcal{N}_i(t) \subseteq \mathcal{N}_i(t+1)$, *i.e.*, the sensing range is chosen so that its set of neighbors is monotonically non-decreasing.

ADRC-III At time $t+1$, the ith robot chooses its sensing range $r_i(t+1)$ so that it has $n_{\mathcal{N}}$ neighbors, where

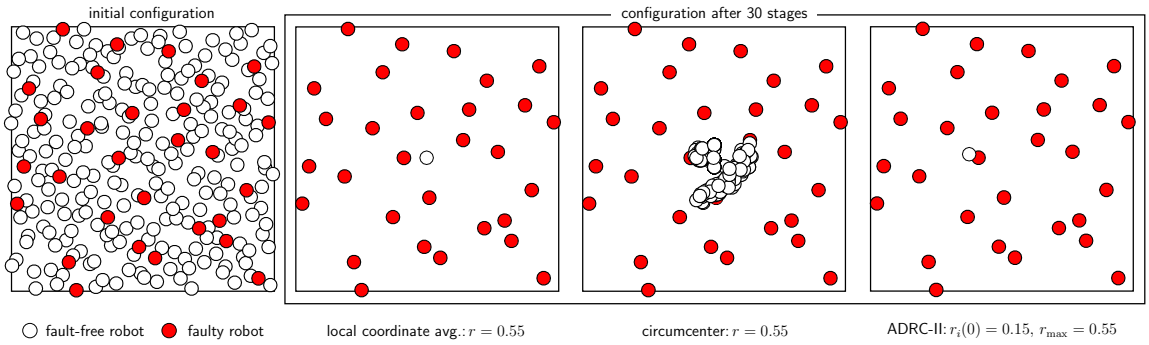


Figure 4.5: Initial configuration and the configuration after 30 stages with stationary faults.

$n_{\mathcal{N}} \triangleq \max\{(n_{f_{\text{local,max}}} + 1)(d+1),\, n_{f_{\text{local,max}}} 2^d + 1\}$

and $n_{f_{\text{local,max}}}$ is the maximum number of faults every robot is desired to tolerate.

Note that ADRC-III imposes a fixed value for the neighborhood size as a function of the maximum number of faulty neighbors, while the ADRC-II allows the number of neighbors to grow to $\overline{n}$ as the robots converge. As a result, the computational cost for ADRC-III may be significantly less than that for ADRC-II, particularly during the final stages of convergence (when the presence of faulty neighbors will have less impact on performance). Furthermore, since ADRC-II enforces continued connectivity with all of its initial neighbors, sensing cost can be made arbitrarily high by a malicious neighbor. For these reasons, if a reasonable estimate is available for the number of faulty neighbors, ADRC-III is a more attractive algorithm.

4.5 Numerical simulation

This section presents a suite of numerical simulation results to demonstrate the performance of our proposed algorithms. We will work with an F-MRS where the intercommunication topology is characterized by the proximity of the robot positions. In particular,the interconnection topology of the robots at time t is defined by a disk graph $\mathcal{G}_{\text{disk}}(t) = (\mathcal{V}, \mathcal{E}(t))$, where $\mathcal{E}(t) \in \mathcal{V} \times \mathcal{V}$ and $(j, i) \in \mathcal{E}(t)$ if and only if $i, j \in \mathcal{V}$, $i \neq j$, and $\|x_j(t) - x_i(t)\| \leq \min\{r_i(t), r_{\max}\}$.

The workspace for our simulations is $Q = [0,\, 1] \times [0,\, 1] \in \mathbb{R}^2$, in which 300 robots are initially deployed in general position. Comparisons of rendezvous performance in the presence of faulty robots are made between three algorithms: a local coordinate averaging algorithm [37], the circumcenter algorithm [1], and ADRC (in its three versions). For local coordinate averaging and the circumcenter algorithm, the sensing radius is fixed and uniform with $r_i(t) = 0.55$. For fair comparison, we apply controllable sensing

radius for our algorithm where the initial values are $r_i(0) = 0.55$, and the upper bound is $r_{\max} = 0.55$. The maximum displacement per stage is $v_{\max} = 0.05$, the convergence error bound is $\epsilon = 1 \times 10^{-7}$, and the number of faults is $n_f = 30$. For ADRC-III, we uniformly set the number of tolerable faulty neighbors for every fault-free robot as $n_{f_{\text{local,max}}} = 40$ such that $n_{\mathcal{N}} = 161$, and $\widetilde{n}_{f_i} = n_{f_{\text{local,max}}} = 40$ for all $i \in \overline{\mathcal{I}}$.

The first simulation considers the case of *stationary faults*. Fig. 4.5 shows the initial configuration, and configuration at stage 30 for the three algorithms. Fig. 4.6 shows position change over the evolutions of each of the three algorithms. As can be seen from the figures, the circumcenter law does not converge in the presence of stationary faults, while both local-averaging and ADRC-II do converge. The value $\widetilde{n}_{f_i} - n_{f_i}$ for ADRC-II is shown in Fig. 4.7(a), and algebraic connectivity[7] over 30 stages is shown in Fig. 4.7(b). The two plots in Fig. 4.7 show that the current example satisfies two connectivity conditions found in Theorem 4.3.1. Fig. 4.8 shows $r_i(t)$ for the fault-free robots (a) for ADRC-II when there are no faults, (b) for ADRC-II when there are 30 faults, (c) for ADRC-III when there are 30 faults. Compared to Fig. 4.8(a) where there are no faults, Fig. 4.8(b) shows that overall the $r_i(t)$ for fault-free robots do not decrease as fault-free robots positions converge to a point. On the other hands, Fig. 4.8(c) shows that if ADRC-III is used under the identical settings, the sensing radii converge. The example shows one advantage of ADRC-III over ADRC-II.

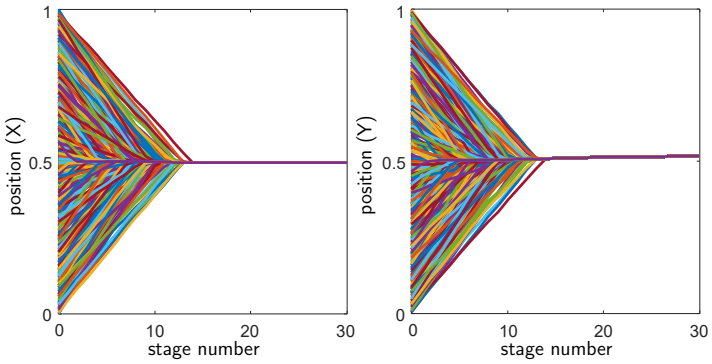
The second simulation results correspond to the case of *dynamic faults*. For this simulation, each faulty robot merely traces out a square pattern (each side of length $2 \times v_{\max}$). The initial configuration, and the faulty robots motion pattern is shown in Fig. 4.9. Fig. 4.10 shows positions change over the evolutions of the three algorithms. As can be seen, ADRC-II converges to a consensus, while both local averaging and the circumcenter algorithm fail to converge. The value $\widetilde{n}_{f_i} - n_{f_i}$ for ADRC-II is shown in Fig. 4.11(a), and algebraic connectivity over 30 stages is shown in Fig. 4.11(b).

Table 4.1: Details of the points of convergence (POC) in Fig. 4.12(b)

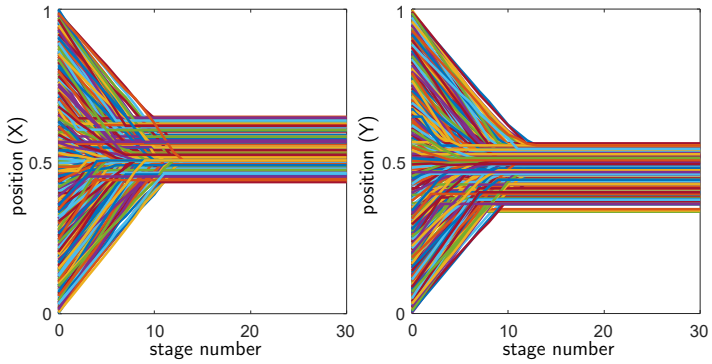
POC	number of nodes at the POC	for each node at the POC, the number of neighbors from				
		A	B	C	D	$\mathcal{F}$
A	81	80	1	0	0	0
B	70	12	69	0	0	0
C	29	0	0	28	25	28
D	90	0	0	0	81	0

In our third and final simulation, we provide a few examples that depict shortcomings of ADRC-III

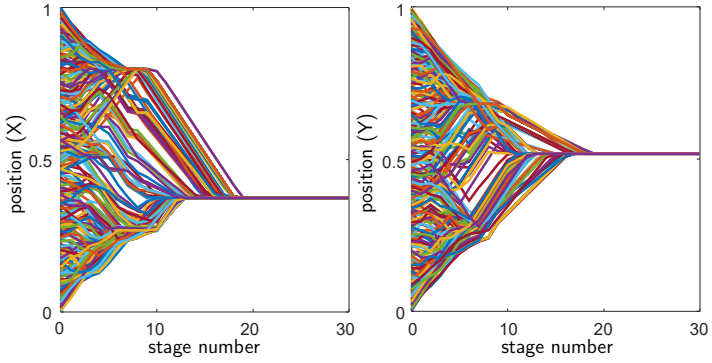
[7]It was shown in [97] that if the algebraic connectivity of a digraph $\mathcal{G}$ is positive, *i.e.*, $\lambda_2(L(\mathcal{G})) > 0$, then $\mathcal{G}$ has a globally reachable node.



(a) local averaging

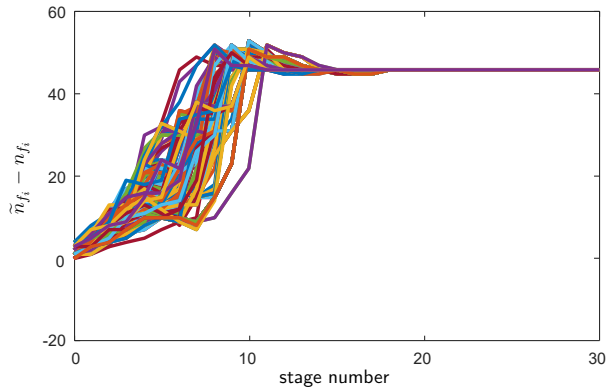


(b) circumcenter

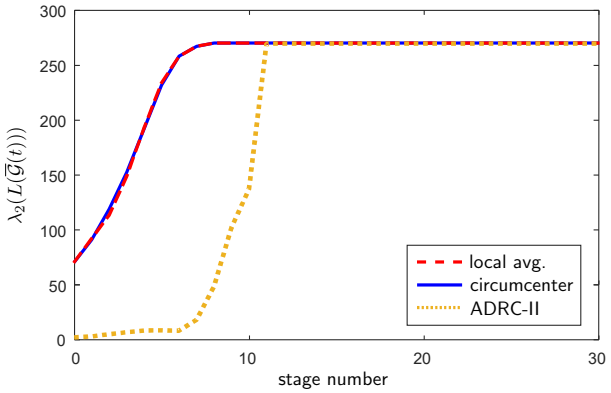


(c) ADRC-II

Figure 4.6: Positions change of fault-free robots during 30 stages.

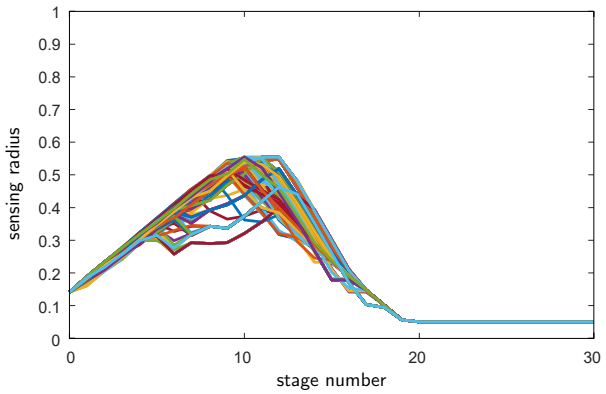


(a) $\widetilde{n}_{f_i} - n_{f_i}$

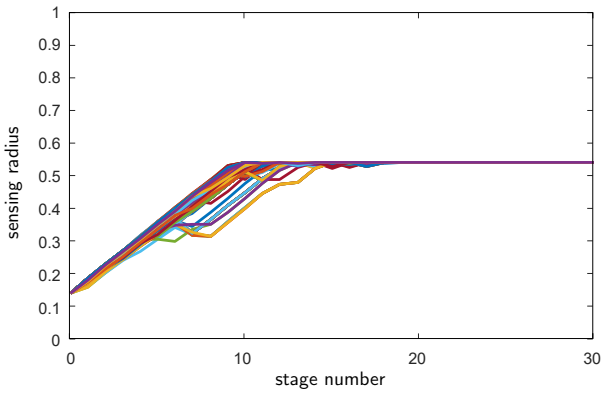


(b) algebraic connectivity

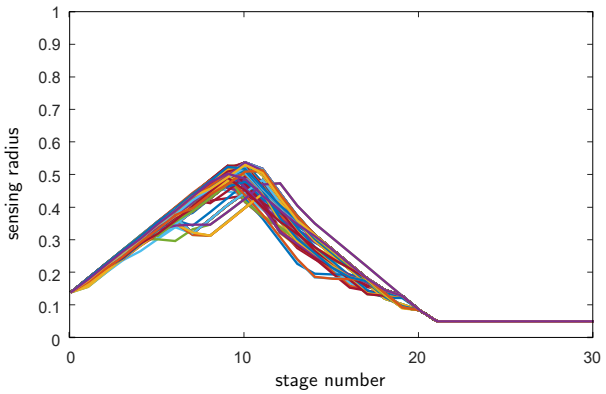
Figure 4.7: Connectivity changes over the evolution.



(a) $n_f = 0$ (ADRC-II)



(b) $n_f = 30$ (ADRC-II)



(c) $n_f = 30$ (ADRC-III)

Figure 4.8: Radii change during the evolution with ADRC-II, III with 0 and 30 stationary faults.

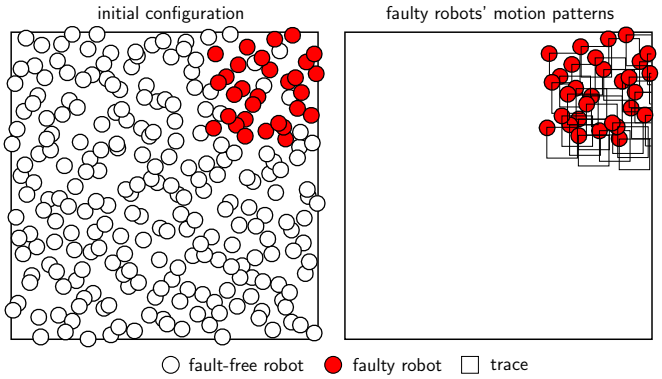


Figure 4.9: Initial configuration and faulty robots' motion pattern.

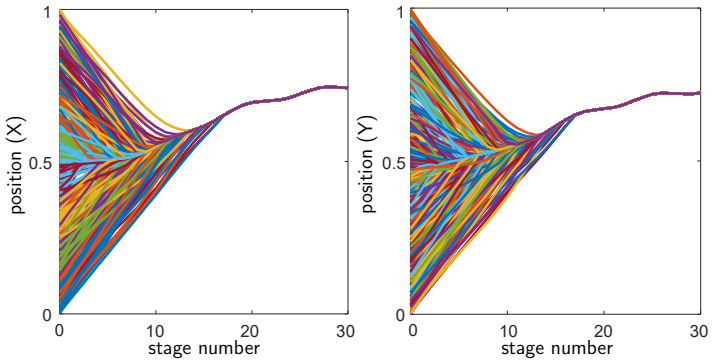
due to limiting the neighborhood size by the value $n_{f_{\text{local,max}}}$. Fig. 4.12 shows the configuration after 30 stages when applying the value for $n_{f_{\text{local,max}}} = 30$ and 20 respectively. The symbols A, B in Fig. 4.12(a) and A–D in Fig. 4.12(b) are the points of convergence (POC), and disks indicate the common sensing ranges of the points. In Fig. 4.12(a), the neighborhood size for every fault-free robot is set to 121. The number of robots at A[8] is 160. Since each robot at A is connected to 121 robots from A, it will not move because its safe point should be found near A. The number of robots at B is 110, and every robot at B is connected to 9 robots from A, 106 robots from B, and 6 faulty robots. Since each safe point for a robot at B is necessarily contained in the convex hull of fault-free neighbors' positions and the value $n_{f_{\text{local,max}}} = 30$ is greater than the number of robots connected to A plus the number of faulty robots (9+6=15), the safe point must be at B. Thus, all the robots at B will not move. Fig. 4.12(b) shows convergence to 4 groups of points after 30 stages, and the result can be analyzed in a similar manner. Refer to Table 7.1 for the details.

4.6 Experimental results

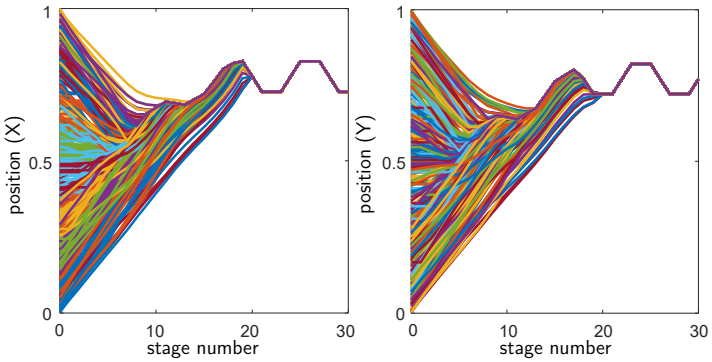
This section presents a series of Robotarium experiments to verify our fault-tolerant rendezvous algorithm. The Robotarium [98,99] is a multi-robot testbed developed at the Georgia Institute of Technology. The testbed consists of custom-designed robots which are called the GRITSBots [99]. An image of the Robotarium testbed is shown in Fig. 4.13(a), and that of the GRITSBot is shown in Fig. 4.13(b).

After our algorithm—being provided as a script—is uploaded to each GRITSBot, robots are initially deployed at a randomly chosen configuration. Then, each robot is commanded to execute the script for a specified number of discrete time steps, which in our case, is set to 1000. Total 10 experiments

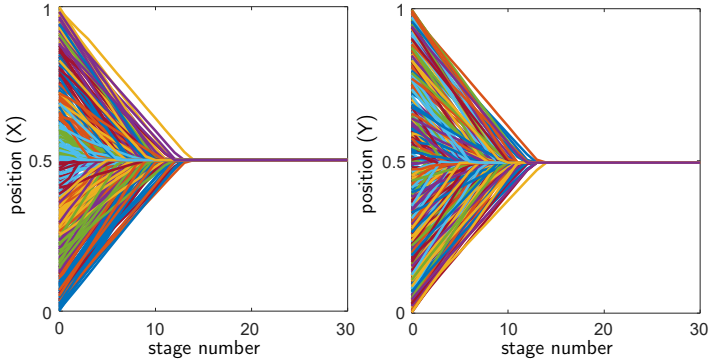
[8]In this context, a robot is at A, if it is sufficiently close to A.



(a) local averaging

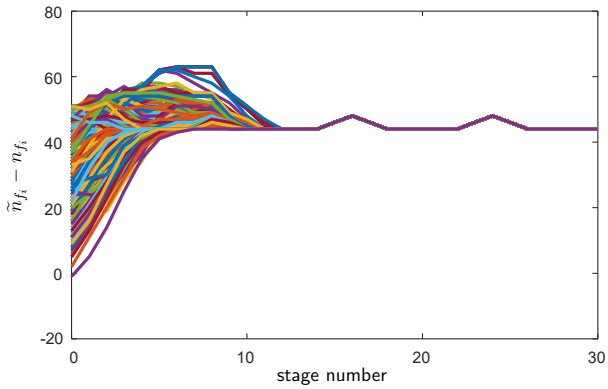


(b) circumcenter

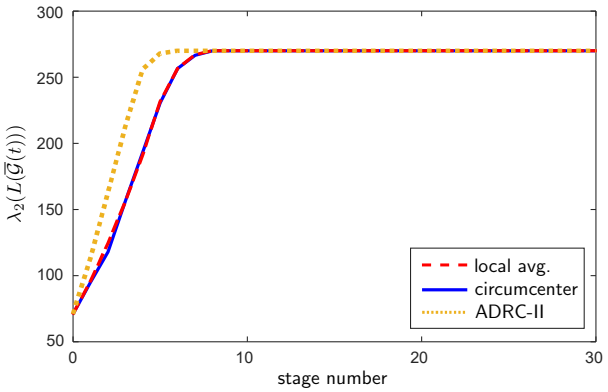


(c) ADRC-II

Figure 4.10: Positions change of fault-free robots during 30 stages.

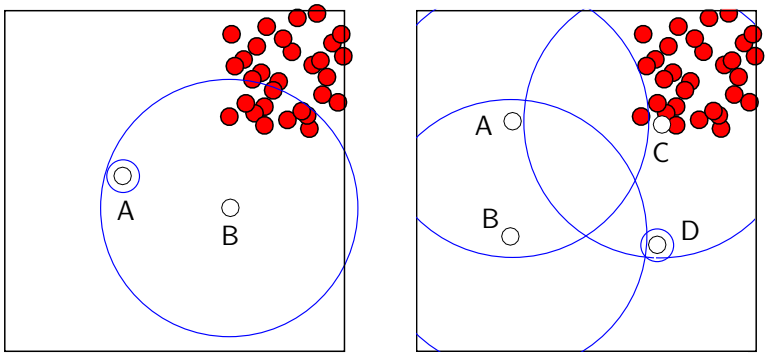


(a) $\widetilde{n}_{f_i} - n_{f_i}$



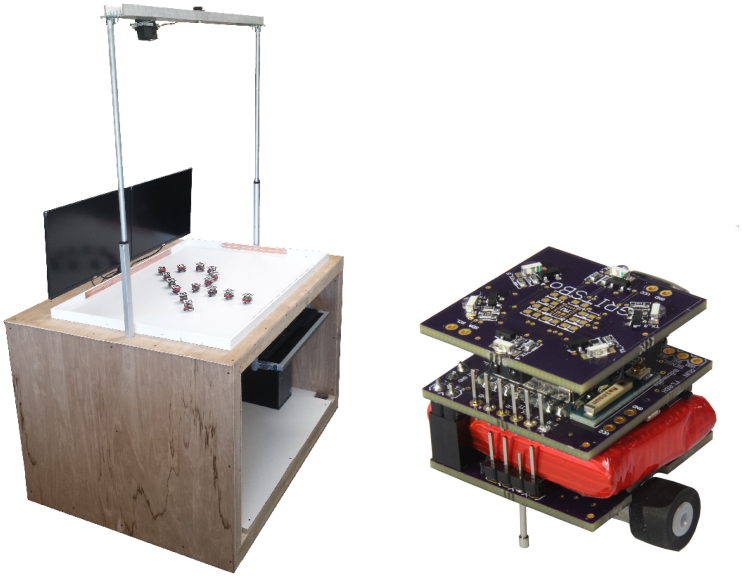
(b) algebraic connectivity

Figure 4.11: Connectivity changes over the evolution.



(a) $n_{f_{\text{local,max}}} = 30$ (b) $n_{f_{\text{local,max}}} = 20$

Figure 4.12: Convergences to multiple points due to the neighborhood size limit.



(a) Robotarium test-bed (b) GRITSBot

Figure 4.13: Photo courtesy of [98].

are carried out to test our theoretical results on the real multi-robot platform. The experiments from #1 through #5 are performed with 8 robots from which one of them are faulty and the experiments from #6 through #10 are performed with 11 robots from which two of them are faulty. Those faulty robots are randomly sampled and simply wander around the workspace periodically based on some random nonlinear sinusoidal function. Due to the small workspace size and the limited availability on the number of robots, we assumed *complete graph* for the interconnection topology throughout the experiments[9].

Fig. 4.14, 4.15, and 4.16 show positions change of all robots during the 1000 iterations. The solid lines show the change in the positions of fault-free robots, and the dashed lines show the change in the positions of faulty robots. As can bee seen from the figures, after 1000 iterations, all the fault-free robots successfully gathered together at a fixed location, regardless of the actions of faulty robots. In particular, Fig. 4.17 shows a few web-cam snapshot images of both initial and final configurations and the motions of all robots obtained from the two experiments: #5 and #10.

[9]Recall that each robot executing ADRC algorithms should have sufficiently large neighborhood size to tolerate certain number of faults.

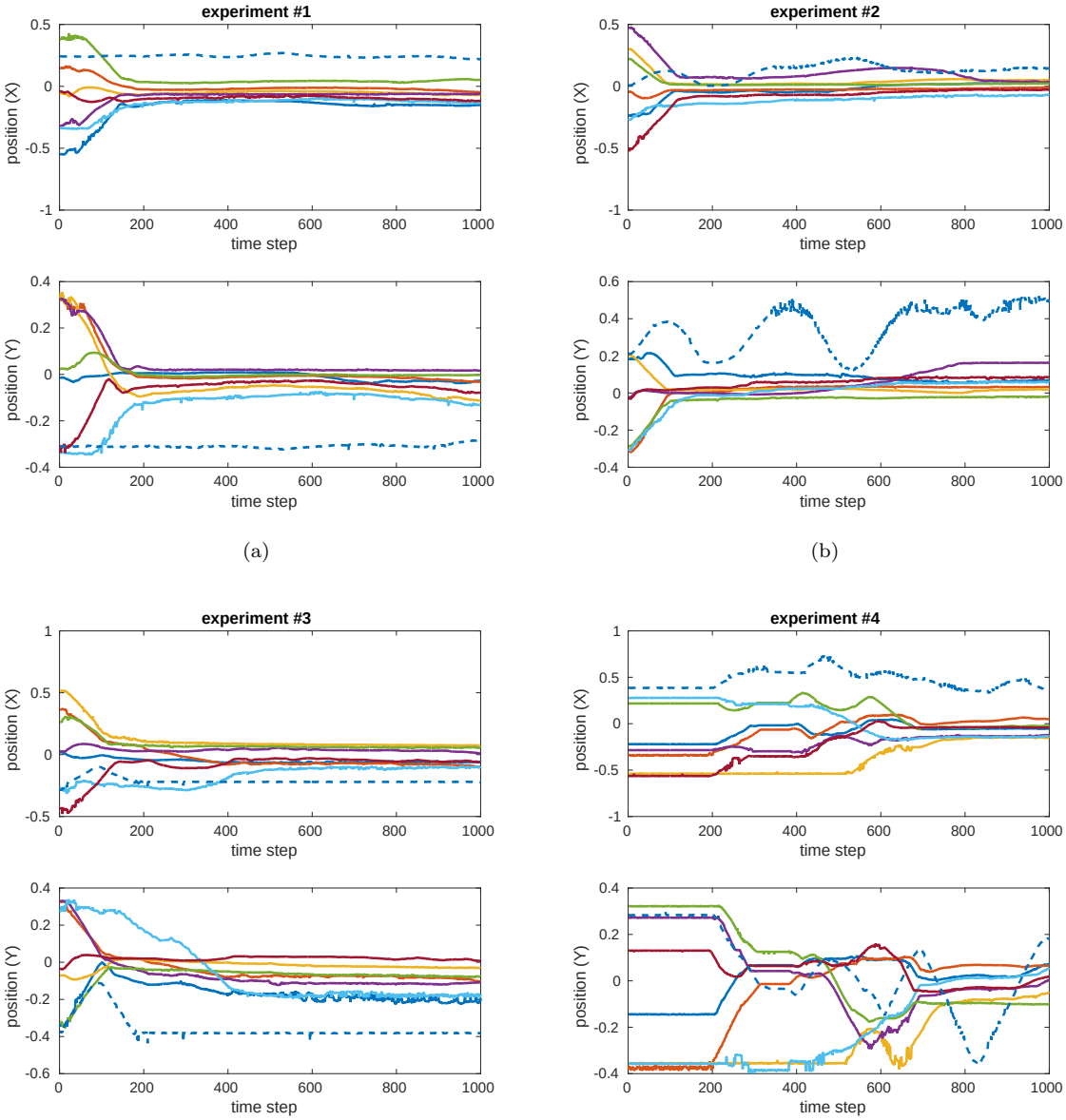
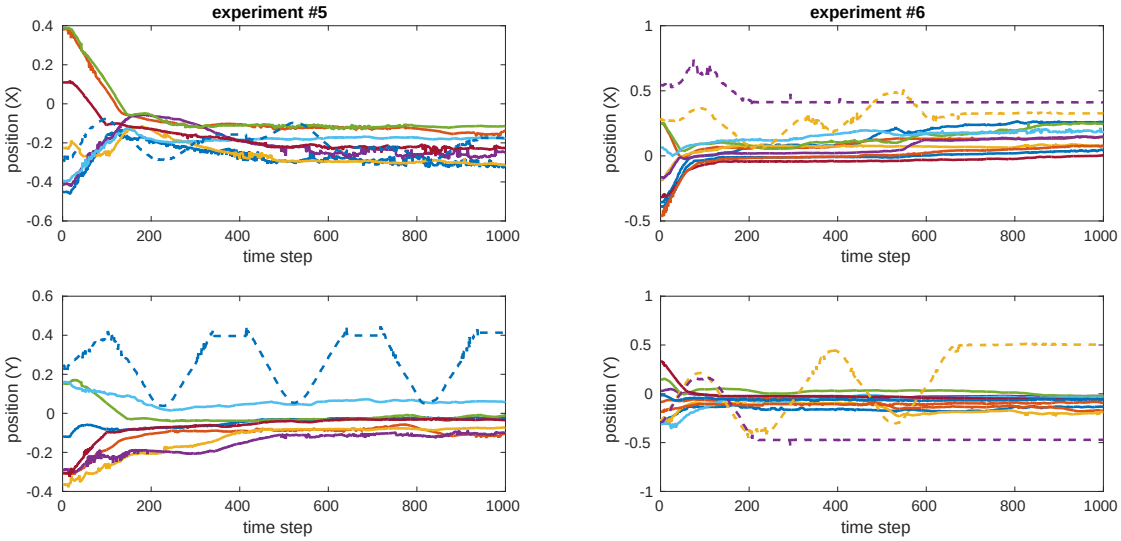
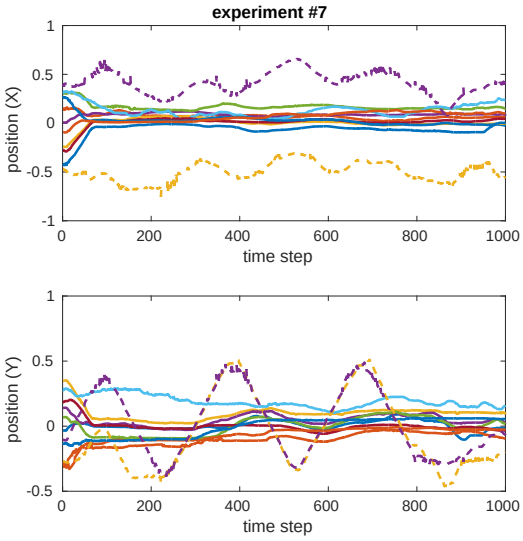


Figure 4.14: Positions change of fault-free robots during 1000 iterations (experiment #1-#4).

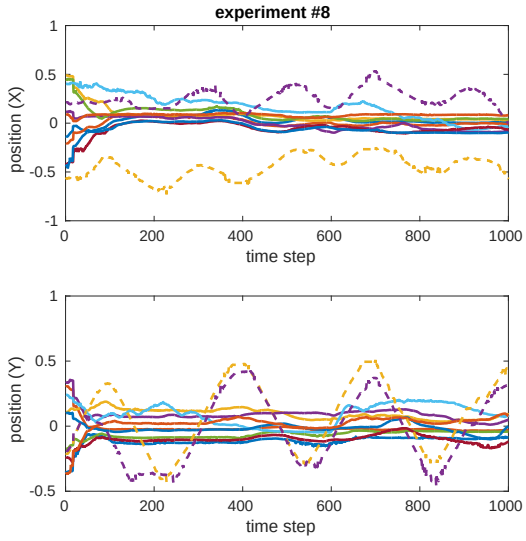


(a)

(b)

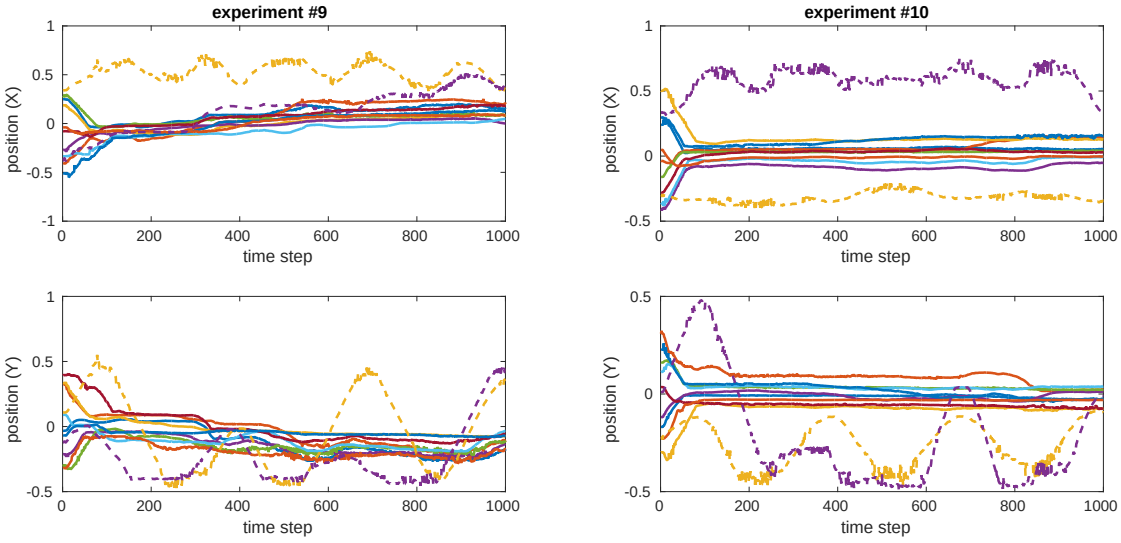


(c)



(d)

Figure 4.15: Positions change of fault-free robots during 1000 iterations (experiment #5-#8).



(a) (b)

Figure 4.16: Positions change of fault-free robots during 1000 iterations (experiment #9-#10).

4.7 Conclusion

This chapter proposed a computationally efficient, decentralized, fault-tolerant algorithm for rendezvous of a group of robots with limited sensing. We provided the convergence analysis of the proposed algorithm by borrowing several tools form ergodic theory, matrix theory, and graph theory. A number of experimental results using a multi-robot testbed as well as a suite of simulation results are provided to illustrate the theoretical results. Future work will involve relaxing the dimensionless robots assumption, *e.g.*, considering fat robot models [100] with collision avoidance, and proposing tighter connectivity constraints for each robot by taking into account the cooperative actions between fault-free robots.

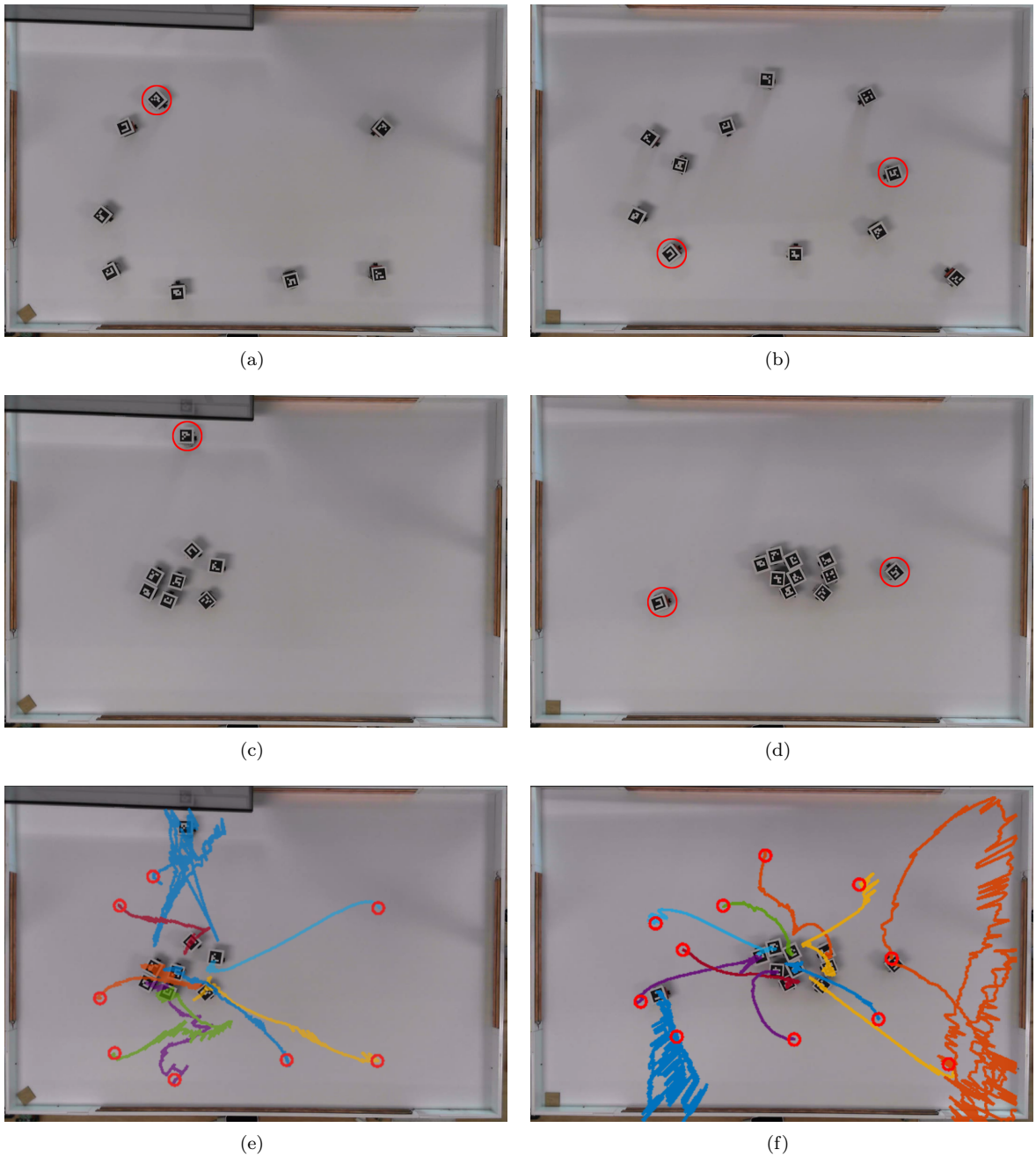


Figure 4.17: Initial configurations (1st row), final configurations (2nd row), trajectories of robots (3rd row) for experiment #5 (left column) and experiment #10 (right column). The circled robots shown in (a)-(d) are faulty.

Chapter 5

Multi-robot robust rendezvous algorithm: an optimization approach

In this chapter, we address the problem of computing distributed motion control algorithms for a multi-robot system that performs rendezvous tasks under random robot failures (*i.e.*, in the case when certain robots may randomly fail to operate properly, for example due to low battery). We assume that it is not known a priori which robots will fail, and that the functioning robots are not able to reliably discern when other robots have failed (*e.g.*, if communication constraints prevent direct verification via queries to potentially failing robots). We do, however, assume that probabilities associated to robot failures are available, and the algorithms that we develop rely on these probability distributions to compute stochastic optimal control strategies to maximize performance in an expected sense.

There is always a possibility that a few robots will fail during coordination tasks, and even, a few faulty robots may result in failure in mission. However to best of our knowledge, there have been relatively few attempts to develop fully distributed algorithms that both achieve the rendezvous task and are resilient in the face of possible failures by some of the robots. The incomplete list of bibliographies related to this topic is contained in the beginning of Chapter 4.

In this chapter, we formulate a distributed optimization problem that every functioning robot solves at each stage. The cost functional can be calculated by each robot using only the limited sensing and communication data *e.g.*, positions of its neighbors. For every functioning robot, the sequence of solutions obtained by solving the problem at each stage becomes a distributed motion control policy. The cost functional is the squared errors between position of of a robot and the average position of the robot plus its neighbors. The idea of our formulation came from Vicsek's model [10] one of the linear averaging algorithms that was verified to work under no failure cases [31].

We model the index set of robots that may fail as a stochastic quantity, and embed the failure model into our formulated problem. We found it difficult to use our proposed stochastic failure model with the previous linear averaging algorithms, *e.g.*, Vicsek's model [10, 31], circumcenter algorithm [9], Lyapunov-based consensus algorithm [37, 38, 40]. We propose a distributed stochastic cost functional

in which the faulty robots are defined with a random set such that they are excluded from the cost computation. Then, we formulate a stochastic problem that minimizes the expectation of the cost function. In addition, we consider mean-variance cost model proposed by [101] to penalize the large variance as well. In addition, to ensure connectivity of the network, we adopt the connectivity constraint from [1, 9, 23, 35], which provides a sufficient condition for rendezvous, and it is easily implementable in a distributed controller.

We choose sequential quadratic programming (SQP) as a our solution method. Instead of solving the nonlinear program directly, using SQP enables us to solve subproblems, each of which is a quadratic program (QP) with linear constraints that is relatively easy to solve.

To demonstrate our algorithms, we provide several numerical simulations in which robots fail at random. In this chapter, we consider only the case in which robot failure implies that the robot is no longer able to move, and is not able to signal to other robots that it has failed (as, *e.g.*, with power loss). Our numerical simulation results shows that our proposed algorithm guarantees statistically better rendezvous task performance than the well-known circumcenter algorithm [9] under stationary faulty robots.

This chapter is organized as follows. In Section 5.1, we formally defines the measure of rendezvous in the presence of faults. Then in Section 5.2, we formulate our distributed rendezvous problem as a 1–step optimal control problem in the presence of potential robot failure, and we investigate constraints for the problem to ensure point convergence. In Section 5.3, we proposed our stochastic robot failure model, and consider the mean–variance cost function to take into account random robot failures. We demonstrate simulation results in Section 5.4 to verify our proposed algorithms. As an extension, we propose a minimax version of our problem, and present a few simulation results with a simple example in Section 5.5. Then, we conclude this chapter in Section 5.6. This work has appeared as a conference paper in [13].

5.1 Rendezvous measure in the presence of faults

In this chapter, we propose multi-robot robust rendezvous in the presence of faults. Since we would like to see and compare the robustness of the algorithm to faults between different algorithms (also for the case robots does not converges or meet at a point), we provide a measure of the robustness. To use the measure, we assume that the interconnection topology if the graph at initial time $t = 0$, *i.e.*, $\overline{\mathcal{G}}(0)$, is connected. Then, if $\overline{\mathcal{G}}(0)$ is connected and $\mathcal{F} = \emptyset$ (*i.e.*, no faults), a correct rendezvous algorithm,

e.g., circumcenter law, will bring all robots to a single point. Otherwise, if there are faults, there is no guarantee for the convergence. Let RDV(t) be the rendezvous measure in the presence faults at the time t which is defined by

$$\mathrm{RDV}(t) = \frac{1}{\overline{n}} \sum_{i=1}^{\overline{n}} \|x_i(t) - \overline{x}_{\mathrm{avg}}(t)\|^2 \tag{5.1}$$

where $\overline{x}_{\mathrm{avg}}(t)$ is the average positions of fault-free robots at t, *i.e.*, $\overline{x}_{\mathrm{avg}}(t) = \frac{1}{\overline{n}} \sum_{i=1}^{\overline{n}} x_i(t)$. In other words, RDV($t$) is a variance of the positions of fault-free robots that shows how far robot's positions are spread out respect to the average position value. Thus, given an initially connected graph $\overline{\mathcal{G}}(0)$, in the most desirable cases, $\mathrm{RDV}(t) \to 0$ as $t \to \infty$.

5.2 Problem formulation

In the section, we formulate a minimization problem for each functioning robot that needs to be solved at each stage. Best For every functioning robot, the sequence of solutions obtained by solving the problem at each stage becomes its distributed motion control policy. Our approach is sometimes called a one–step optimization method, or the *greedy algorithm* depending on context. Our problem formulation in this section is used as a framework to apply stochastic robot failure model in the following section.

Recall that $x(t)$ is configuration of n robots deployed in $Q^n \subseteq (\mathbb{R}^d)^n$ at some time step $t \in \mathbb{Z}_{\geq 0}$. The discrete time evolution for each fault-free robot $i \in \overline{\mathcal{I}}$ is given by

$$x_i(t+1) = f_i(x_i(t),\, u_i(t)) = x_i(t) + u_i(t), \;\; t \in 0, 1, 2, \dots$$

We want to find the control policy, *i.e.*, an ordered set of control vectors $(u_i(0), u_i(1), u_i(2), \dots)$ for each functioning robot $i \in \overline{\mathcal{I}}$ which minimizes a certain cost function for $t = 0, 1, 2, \dots$.

5.2.1 Rendezvous problem

We formulate our distributed rendezvous problem based on the *Vicsek's model* [10]. In [10], Vicsek *et al.* proposed a nearest neighbor rule that is used to update the heading of self-driven particles moving at a same speed with the average of the particle's own direction and the directions of its neighbors. In a similar manner, at every stage, each functioning robot $i \in \overline{\mathcal{I}}$ senses the positions of its neighbors, *i.e.*, other agents within specified distance r, and moves towards the average of its own position and its

neighbors' positions. The discrete time evolution of the system using the algorithm is given by

$$x_i(t+1) = \frac{1}{|\mathcal{N}_i(t)|+1}\left(x_i(t) + \sum_{j\in\mathcal{N}_i(t)} x_j(t)\right). \tag{5.2}$$

It was shown by Jadbabaie *et al.* [31] that under a few assumptions on the network connectivity, agents moving with Vicsek's model converge to a consensus. Note that (5.2) is a special case of adjacency-based linear averaging algorithm [20, 31] with unweighted adjacency matrix. For details of the subject refer to Chapter 1 of [20].

Based upon the linear update rule, a minimization problem for each fault-free robot $i \in \overline{\mathcal{I}}$ at stage t is formulated as follows.

$$\underset{u_i(t)}{\text{minimize}} \quad \left\| \frac{1}{|\mathcal{N}_i(t)|+1}\left(\sum_{j\in\mathcal{N}_i(t)} x_j(t) + x_i(t)\right) - x_i(t) - u_i(t)\right\|^2. \tag{5.3}$$

The problem is to find a control $u_i(t) \in \mathbb{R}^d$ for each fault-free robot $i \in \overline{\mathcal{I}}$ that minimizes the distributed cost given by a squared distance between ith robot position at stage $t+1$, and the average positions of the ith robot's neighbors plus the robot itself at stage t. Note that if faults present among its neighbor, each fault-free robot $i \in \overline{\mathcal{I}}$ cannot calculate the correct cost value that should be the function of *only* the positions of fault-free neighbors and the robot's input. Using the previous results from [38, 102], we can state the following theorem.

Theorem 5.2.1. *If $\overline{\mathcal{G}}(t)$ is static, strongly connected, and balanced for $t \in \mathbb{Z}_{\geq 0}$,* $\mathrm{RDV}(t) \to 0$ *as* $t \to \infty$.

5.3 Optimal strategies in the presence of random robot failures

In this section, we propose a probabilistic robot failure model. Based upon our previous formulation (5.3), we will formulate a cost functional that depends on the set of positions of faulty robots. Then we propose two stochastic programs: the one that minimize the expectation of the cost, and the other that minimizes the linear combination of expectation and the variance of the cost.

5.3.1 Modeling robot failure

In this subsection, we propose a method to model the random robot failure. In particular, we will look at the local model, *i.e.*, for each fault-free robot $i \in \overline{\mathcal{I}}$, the event that any subset of its neighbors are faulty. For this, we consider a distributed probability space for ith robot at stage t that is defined with 3-turple

$$(\Omega_i(t), \mathcal{F}_i(t), P_i(t))$$

where

- $\Omega_i(t)$ is the *sample space* which is set of outcomes that corresponds to set of all possible failure configurations among the neighbors of i such that $|\Omega_i(t)| = 2^{|\mathcal{N}_i(t)|}$.
- $\mathcal{F}_i(t)$ is the *event space* which is the collection of all subset of $\Omega_i(t)$, *i.e.*, $\mathcal{F}_i(t) \in 2^{\Omega_i(t)}$.
- $P_i(t)$ is the *probability measure*, *i.e.*, probability distribution, such that $P_i(t)(\Omega_i(t)) = 1$.

We define a random set $F_i(t)$ as a map $F_i(t) : \Omega_i(t) \to 2^{\mathcal{N}_i(t)}$, and we define $n_{f_i}(t) = |F_i(t)|$. The physical meaning of $F_i(t)$ in our context is the index set of the random faults present among the set of neighbors $\mathcal{N}_i(t)$ such that $F_i(t) \subseteq \mathcal{N}_i(t)$.

5.3.2 An example of robot failure model

In this subsection, we consider the binomial distribution to model our random robot failures in cases when failure events for individual robots are *mutually independent*. Suppose that $n_{f_i}(t)$ is a random variable that is *binomially distributed* that is sum of $|\mathcal{N}_i(t)|$ mutually independent failure events identically distributed with probability p such that we may write $n_{f_i}(t) \sim Bi(|\mathcal{N}_i(t)|, p)$. For each fault-free robot $i \in \overline{\mathcal{I}}$, we consider a random variable $X_{ij}(t)$ associated with it neighbor $j \in \mathcal{N}_i(t)$ by

$$X_{ij}(t) = \begin{cases} 1, & \text{if robot } j \text{ fails} \\ 0, & \text{if robot } j \text{ does not fail} \end{cases}$$

and $X_{ij}(t) \sim Be(p)$ such that

$$P\{X_j = 1\} = p, \tag{5.4}$$

which is read as probability that the robot with index j fail equals p, and

$$P\{X_j = 0\} = 1 - p. \tag{5.5}$$

Since, in this example random variables $(X_{ij}(t))_{j\in\mathcal{N}_i(t)}$ are mutually independent, and identically distributed, we may write $n_{f_i}(t)$ as the sum of random variables $(X_{ij}(t))_{j\in\mathcal{N}_i(t)}$

$$n_{f_i}(t) = \sum_{j\in\mathcal{N}_i} X_{ij}(t). \tag{5.6}$$

Thus, the probability measure for an event that robots with index set w are faulty among $\mathcal{N}_i(t)$ is expressed by the following PMF

$$\begin{aligned} p_{F_i(t)}(w) &= P_i(t)\,\{\text{“robots with index set } w \text{ are faulty”}\} \\ &= p^{|w|}(1-p)^{|\mathcal{N}_i(t)|-|w|}, \;\; i \in \overline{\mathcal{I}},\, w \in \Omega_i(t). \end{aligned} \tag{5.7}$$

This can be read as probability of an event that robots with index w fail with probability p and rest of the robots in the neighbors do not fail with probability $1-p$. While in binomial random variable, only the number of trials, *i.e.*, experiment, and number of success (or fail) is the concern, with our model, not only the number of robots that fails $n_{f_i}(t)$, but the actual indices of the robots that fail $F_i(t)$ are important as well.

5.3.3 A stochastic optimization

If certain robots are faulty, our cost function should depend only on the positions of fault-free robots. Give some fault-free robot with index $i \in \overline{\mathcal{I}}$ at stage t, let the cost functional be a real valued function $g_i : (\mathbb{R}^d)^{\widetilde{\mathcal{N}}_i(t)} \times \Omega_i(t) \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ defined by

$$g_i(\widetilde{x}_i(t),\, F_i(t),\, u_i(t)) = \left\| \frac{1}{|\widetilde{x}_i(t)| - |F_i(t)|} \left(\sum_{\substack{x_j(t)\in\widetilde{x}_i(t),\\ \text{s.t. } j\notin F_i}} x_j(t) \right) - x_i(t) - u_i(t) \right\|^2. \tag{5.8}$$

Note that the cost functional g_i for each fault-free robot i is a function of only the local information, *e.g.*, position of its neighbors that are not in $F_i(t)$. Since (5.8) contains the random set $F_i(t)$, it is not possible for ith robot to calculate the cost. Instead each robot can minimize the expectation of the cost

provided that the probability distribution of $F_i(t)$ is known *a priori*.

$$\underset{u_i(t)}{\text{minimize}} \;\; \mathbb{E}\Big[g_i(\widetilde{x}_i(t),\, F_i(t),\, u_i(t))\Big]. \tag{5.9}$$

If one wants to penalize the input which results in relatively large variance, we may consider minmizing the linear combination of expectation and variance of the cost. This leads to a mean-variance problem given as follows.

$$\underset{u_i(t)}{\text{minimize}} \;\; \mathbb{E}\Big[g_i(\widetilde{x}_i(t),\, F_i(t),\, u_i(t))\Big] + \alpha_v \mathbf{var}\Big[g_i(\widetilde{x}_i(t),\, F_i(t),\, u_i(t))\Big] \tag{5.10}$$

where α_v is a positive weight for variance term. This approach is better known as *mean–variance model* in robust optimization (RO) literature, *e.g.* [103–105], and was first introduced by [101] on his portfolio selection model. Originally, this formulation was proposed to be used for high–risk decision making under uncertainty. According to literature on the subject, the variance is closely related to risk aversion measure. However, in many situations two quantities are conflicting (see text from multi-objective optimization, *e.g.* [106]). Also quite often, it is of its own interest to find the best α_v for Pareto efficiency.

5.3.4 Inequality constraints

In this subsection, we consider a constrained stochastic optimization by adding a few inequality constraints to (5.9), (5.10). The constraints will be used for SQP formulation in the next subsection. It is known from [30,35,50] that sufficient constraints for rendezvous (with no faults) are : first, at each time step, every robot moves to a position that is in the strict convex-hull of the positions of it neighbors (we call this *strict convex-hull constraint*), and second, at each time step, every robot maintains its connectivity with its neighbors (*connectivity constraint* [1]). The two conditions together provide sufficiency for the algorithm to converge to a point. Along with two constraints, we add velocity constraint that limits each robot's the travel distance during each stage. We define m numbers of real-valued functions $h_{ij} : (\mathbb{R}^d)^{\widetilde{\mathcal{N}}_i(t)} \times \Omega_i(t) \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ where where the subscript ij indicates the jth constraint for ith robot such that $j \in \{1, \dots m_i(t)\}$. The number $m_i(t)$ corresponds to the total number of constraints. There are 3 types of constraints among total $m_i(t)$ constraints. For convenience, we assume that each type contains $m_{i1}(t)$, $m_{i2}(t)$, $m_{i3}(t)$ number of inequalities respectively. For each fault-free robot with index $i \in \overline{\mathcal{I}}$, three types of constraints are defined as follows.

5.3.4.1 Proper convex–hull constraint

As was discussed in [50], one of the sufficient conditions for distributed rendezvous are at each time step, every robot move to the strict convex hull of the positions of its neighbors plus the position of the robot itself. In the presence of fault, each robot must move to the strict convex hull of the positions of its fault-free neighbors plus the position of the robot itself. We note that the convex hull formed with position of ith robot and its fault-free neighbors can be represented by linear inequalities as:

$$\begin{bmatrix} a_{i1}(t)^T \\ \vdots \\ a_{im_{i1}(t)}(t)^T \end{bmatrix} y \leq \begin{bmatrix} b_{i1}(t) \\ \vdots \\ b_{im_{i1}(t)}(t) \end{bmatrix} \tag{5.11}$$

where each $a_{ij}(t)$ is a $d \times 1$ vector, each $b_{ij}(t)$ is a scalar, y is a $d \times 1$ vector of reals, and $m_{i1}(t)$ is the minimum number of linear inequalities required such that $m_{i1}(t) \leq |\mathcal{N}_i(t)| - n_{f_i}(t) + 1$. Then, for all $j = 1, \ldots, m_{i1}(t)$, $x_i(t+1) = x_i(t) + u_i(t)$ must satisfy

$$h_{ij}(\widetilde{x}_i(t),\, F_i(t),\, u_i(t)) = a_{ij}(t)^T(u_i(t) + x_i(t)) - b_{ij} - \delta \leq 0 \tag{5.12}$$

where $\delta > 0$ is a small positive real number that is used to emulate strict inequality to ensure that the feasible region is in the interior of the convex-hull $\text{conv}((x_i)_{i \in \overline{N}_i(t) \cup \{i\}})$.

5.3.4.2 Connectivity constraint

The connectivity constraint provides another sufficient conditions for exact rendezvous [24, 35] of a group of robots. Recall that pairwise connectivity constraint [1] for ith robot and its neighbor l is

$$\mathcal{C}_{i,l}(t) = \overline{\mathcal{B}}\left(\frac{x_l(t) - x_i(t)}{2},\, \frac{r}{2}\right). \tag{5.13}$$

Equivalently, this can be expressed with quadratic inequalities as follows. For $j = m_{i1}(t)+1, \ldots, m_{i1}(t)+ m_{i2}(t)$,

$$h_{ij}(\widetilde{x}_i(t),\, F_i(t),\, u_i(t)) = \left\| u_i(t) - \frac{x_l(t) - x_i(t)}{2} \right\|^2 - \frac{r^2}{4} \leq 0, \tag{5.14}$$

for all $l \in \mathcal{N}_i(t)$, and since there are exactly $\left|\overline{\mathcal{N}}_i(t)\right|$ number of fault-free neighbors, $m_{i2}(t) = \left|\overline{\mathcal{N}}_i(t)\right|$.

5.3.4.3 Velocity constraint

The last constraint accounts for the maximum travel distance during each time interval. Let v_i be the maximum travel distance allowed for ith robot. Then for $j = m_{i1}(t) + m_{i2}(t) + m_{i3}(t)$,

$$h_{ij}(\widetilde{x}_i(t),\, F_i(t),\, u_i(t)) = (u_i(t))^T u_i(t) - v_i^2 \leq 0 \tag{5.15}$$

and $m_{i3}(t) = 1$.

Remark 5.3.1. *The strict convex-hull constraint is usually satisfied with local averaging algorithm, because, for each robot, the average of the positions of neighbors is necessarily in the strict convex hull of the positions of neighbors. Nevertheless, we provide the condition here to explicitly emphasize that the condition together with connectivity constraint ensures the rendezvous of a group of initially connected robots without faults.*

In our simulation, we relax both proper convex-hull constraint and connectivity constraint. This is simply because we do not know which robots are faulty and which robots are not.

5.3.5 SQP formulation

In this section we state our sub–problem which is used in SQP algorithm (See the Appendix A). First let us assume that we are given a set of positions of robots at stage $t \in \mathbb{Z}_{\geq 0}$, i.e, $x(t)$ and would like to obtain the optimal control vector for fault-free robots $\overline{u}(t)^* = \{u_1(t)^*, \ldots, u_{\overline{n}}(t)^*\}$. For each $i \in \overline{\mathcal{I}}$ and $t \in \mathbb{Z}_{\geq 0}$, we fix $\tilde{x}_i(t)$ and vary $u_i(t)$ by first letting $u_i(t) := u_i^0$ where u_i^0 is the initial iterate[1].

$$\begin{aligned}
\underset{d_{u_i}^k}{\text{minimize}} \quad & \nabla L(u_i^k,\, \lambda_i^k) d_{u_i}^k + \tfrac{1}{2} d_{u_i^k}^T H(u_i^k,\, \lambda_i^k) d_{u_i}^k \\
\text{subject to} \quad & \begin{bmatrix} \nabla h_{i1}(\widetilde{x}_i(t),\, F_i(t),\, u_i^k) \\ \vdots \\ \nabla h_{im_i(t)}(\widetilde{x}_i(t),\, F_i(t),\, u_i^k) \end{bmatrix} d_{u_i}^k + \begin{bmatrix} h_{i1}(\widetilde{x}_i(t),\, F_i(t),\, u_i^k) \\ \vdots \\ h_{im_i(t)}(\widetilde{x}_i(t),\, F_i(t),\, u_i^k) \end{bmatrix} \leq 0.
\end{aligned}$$

where $k = 0, 1, \ldots$ is the iteration number, and $d_{u_i}^k = u_i - u_i^k$. The Lagrangian is given as follows.

$$\begin{aligned}
L(u_i^k,\, \lambda_i^k) = \mathbb{E}\Big[g_i(\widetilde{x}_i(t),\, F_i(t),\, u_i^k)\Big] + \alpha_v \mathbf{var}\Big[g_i(\widetilde{x}_i(t),\, F_i(t),\, u_i^k)\Big] \\
+ \sum_{j=1}^{m_i(t)} \lambda_{ij}^k h_{ij}(\widetilde{x}_i(t),\, F_i(t),\, u_i^k).
\end{aligned}$$

[1]For example, u_i^0 can be set to $d \times 1$ vector with 0s.

Using the solution of (5.3.5), $u_i(t)$ is updated $u^k \to u_i(t)$ for each iteration step, and as the iteration proceeds, $u_i(t)$ approaches to locally optimal solution of QP.

5.4 Simulation results

This section is organized as follows. First, we compare a pure rendezvous performance between cicumcenter algorithm [1] and the local averaging algorithm when there are no faults. Then, we present a few results which compares rendezvous performance between circumcenter algorithm and our proposed algorithm under random stationary robot failures.

In our simulation we let Q to be a square workspace $[0\ 50]^2$ in $\mathbb{R}^2$, and choose n pseudo–random points in Q for the robots' initial positions. The maximum travel distance for each robot is equally set to 2, *i.e.*, $v_1 = \cdots = v_n = 2$.

5.4.1 Comparison of rendezvous task performance between circumcenter law and local averaging algorithm without failure

In this subsection, we evaluate the rendezvous performance of the local averaging algorithm with no faults (each robot's control policy is determined as sequence of solutions to the problem in (5.10) subject to inequality constraints discussed in subsection 5.3.4 by setting $p = 0$, $\alpha_v = 0$). We do this by comparing the approximate rendezvous time with circumcenter algorithm. Table 5.1 compares minimum time for approximate rendezvous, *i.e.*, the smallest $t^\star$ s.t. $\mathrm{RDV}(t) < 1\mathrm{E}-7$ for all $t \geq t^\star$, with different sensing radius between two algorithms: circumcenter and our algorithm when varying the total number of robots. The rendezvous time is measured with the minimum stage number at which every fault-free robots rendezvous at a point. For all cases, local averaging algorithm outperforms circumcenter law, and the performance gain is more evident when total number of robots is large.

Table 5.1: Comparison of rendezvous time, i.e, iteration number between two algorithms

number of robots	sensing radius	circumcenter	local averaging algorithm
10	25	14	14
50	16	21	16
100	9	24	22

5.4.2 Comparison between circumcenter law, local averaging algorithm, and our stochastic algorithm under random stationary robot failures

In this section, we compare resiliency to stationary robot failure between circumcenter algorithm, local averaging algorithm, and our proposed stochastic algorithm. Unless otherwise noted, for all the algorithms that we compare, the both proper convex-hull constraint and the connectivity constraint were relaxed. We generate 100 random samples of different configurations with 50 robots in which the average failure rate is 10%, and each robot's sensing/communication radius is 20. For our proposed algorithm we used $p = 0.1$ and $\alpha_v = 100$.

Both Fig. 5.1 (a combined plot) and Fig. 5.2 (individual plots) show scatter plots which compare mean of positions of functioning robots between algorithms after 50 stages, using 100 random samples that are generated with different configurations with 50 robots in which the average failure rate is 10%. Fig. 5.3 (a combined plot) Fig. 5.4 (individual plots), and Fig. 5.5 (histogram) show RDV(50), *i.e.*, the variance of positions of functioning robots between algorithms obtained with the same settings after 50 stages, where the horizontal axis is the sample number and the vertical axes are RDV(50). We chose the final stage to be 50, because change in the values after 50 stages are insignificant. As can be seen from the figures, local averaging algorithm results in relatively smaller variance than circumcenter algorithm, either with or without connectivity constraints. Also, stochastic version of our local averaging algorithm—where the failure model has binomial distribution with $p = 0.1$, and $\alpha_v = 100$—shows relatively smaller variance than the local averaging algorithm.

According to the statistical results, our proposed stochastic version of local averaging algorithm is less sensitive to stationary robot failures than other rendezvous algorithms.

5.5 A min-max problem

In both Fig. 5.3 and Fig. 5.4, a few peaks are observed from our stochastic algorithm with $p = 0.1$ and $\alpha_v = 100$. This means that the worst-case performance of our proposed algorithm is poor. This is not surprising because our proposed algorithm finds solution which minimize the cost in an expected sense.

We plan to consider minimax version of our program by minimizing the linear combination of expected cost and the variance given the worst-case probability distributions under a few constraints on the probability distribution.

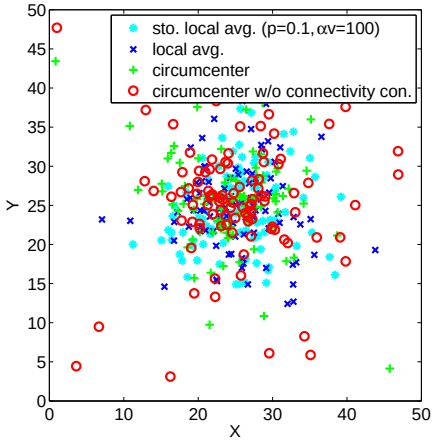


Figure 5.1: Scatter plots of positions mean of functioning robots between different algorithms after 50 stages with 100 samples where avg. failure rate is 10%.

5.5.1 Poisson-binomial probability distribution

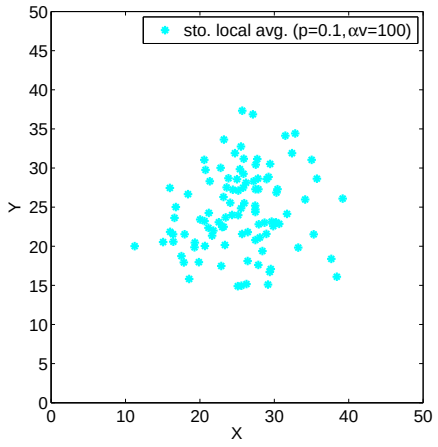
First we relax the previous assumption on identically distributed probability of failure events for different robots. The motivation for this is to develop more robust method which can handle broader cases of failure scenarios in which failure rate may not be identical for different robots. While the stochastic program is minimization of linear combination of expected function and variance term which requires probability distribution to be known a priori, in minimax stochastic program—we are about to propose—this is not the case. As long as the probability distribution is properly constrained, the solution to the problem can be obtained. For our problem, we consider the probability distribution to be a generalized version of binomial distribution known as *Poisson-Binomial distribution*. The formal definition is contained in various texts on probability theory, *e.g.*, [107].

Our Poisson-binomially distributed random variable is expressed by $n_{f_i}(t) \sim PBD((p_j)_{j\in\mathcal{N}_i(t)})$ such that it is the sum of Bernoulli distributed independent random variables $(X_{ij}(t))_{j\in\mathcal{N}_i(t)}$

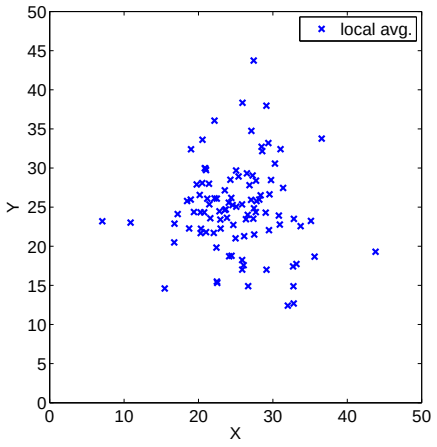
$$n_{f_i}(t) = \sum_{j\in\mathcal{N}_i(t)} X_{ij}(t). \tag{5.16}$$

In particular, each $X_{ij}(t)$ is random variable with values $\{0, 1\}$ such that

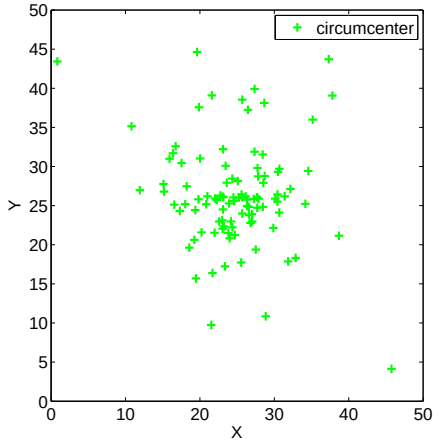
$$P\{X_{ij}(t) = 1\} = p_j, \tag{5.17}$$



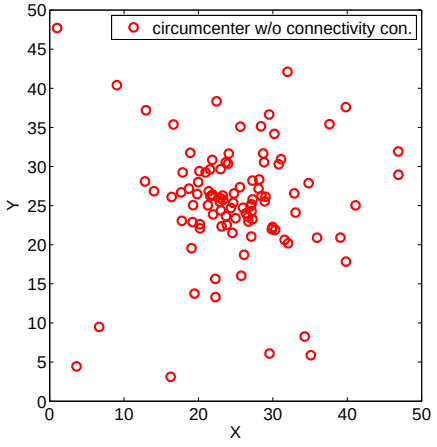
(a) circumcenter law



(b) circumcenter law without enforcing connectivity constraint



(c) local averaging



(d) stochastic local averaging

Figure 5.2: Scatter plots of positions mean of functioning robots between different algorithms after 50 stages with 100 samples where avg. failure rate is 10%.

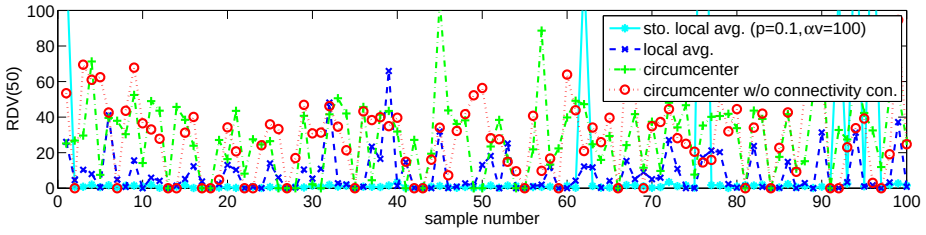
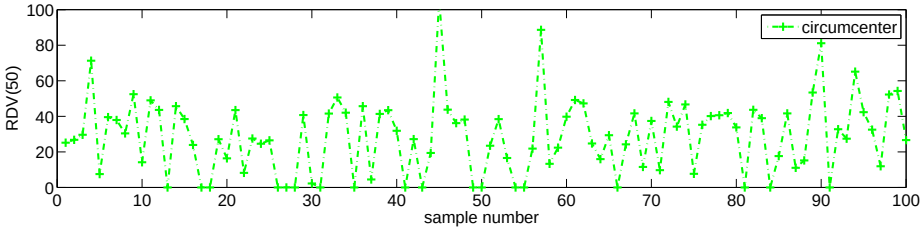
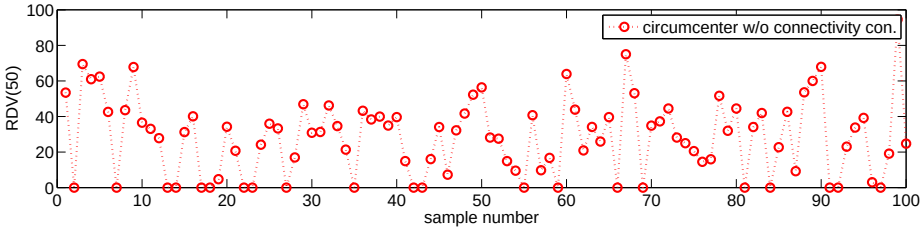


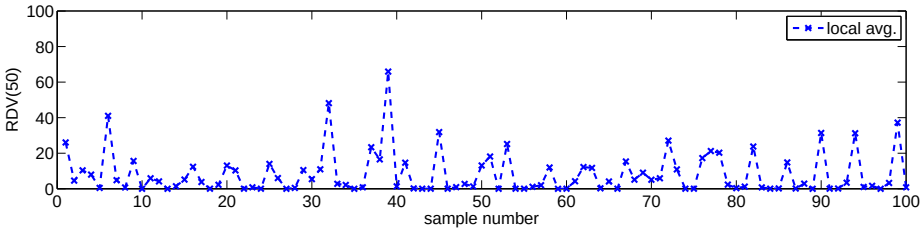
Figure 5.3: Comparison of RDV(50) values between different algorithms after 50 stages with 100 samples where avg. failure rate is 10%.



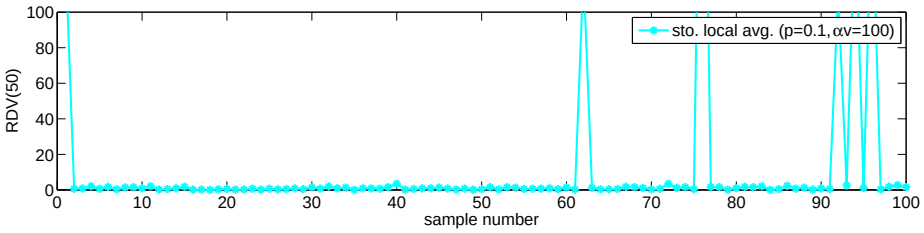
(a) circumcenter law



(b) cirumcenter law without enforcing connectivity constraint

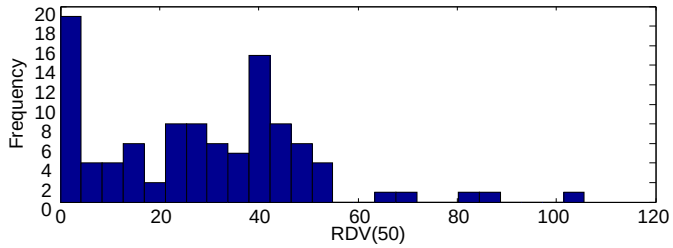


(c) local averaging algorithm

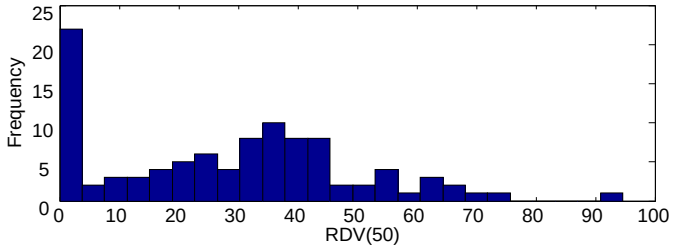


(d) stochastic local averaging algorithm

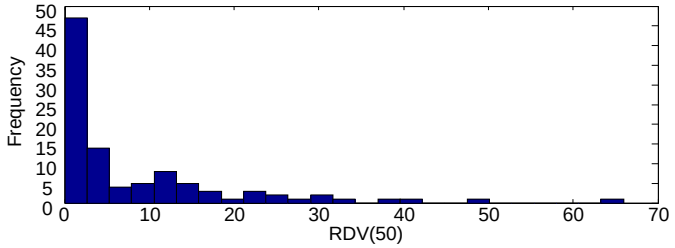
Figure 5.4: Comparison of RDV(50) values between different algorithms after 50 stages with 100 samples where avg. failure rate is 10%.



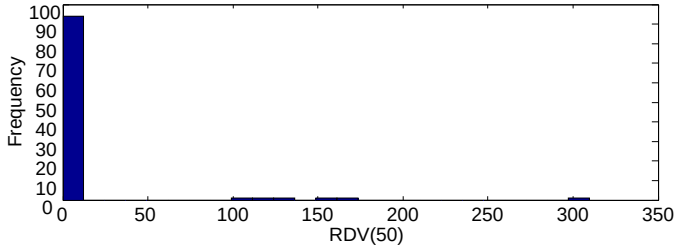
(a) circucenter law



(b) cirumcenter law without enforcing connectivity constraint



(c) local averaging



(d) stochastic local averaging

Figure 5.5: Comparison of RDV(50) values between different algorithms after 50 stages with 100 samples where avg. failure rate is 10%.

that is read as probability that the robot with index j fail equals p_j, and

$$P\{X_{ij}(t) = 0\} = 1 - p_j \tag{5.18}$$

is read as probability that the robot with index j does not fail is $1 - p_j$. Note that $p_j = p_h$ is not necessarily true for $j,\ h \in \mathcal{N}_i(t)$ with $j \neq h$. Thus given some vector $(p_j)_{j\in\mathcal{N}_i(t)} \in [0,\ 1]^{|\mathcal{N}_i(t)|}$, and the Poisson-Binomial distributed random variable $n_{f_i}(t)$, the PMF for the random set $F_i(t)$ is defined by

$$p_{F_i(t)}(w) = \prod_{j\in w} p_j \prod_{h\in\overline{\mathcal{N}}_i(t)} (1 - p_h),\ \ i \in \overline{\mathcal{I}},\ w \subseteq \mathcal{N}_i(t). \tag{5.19}$$

5.5.2 Stochastic minimax program

In this subsection, we briefly explain our plan to formulate the problem when the probability distribution is not known *a priori* and we would like to obtain solution that is robust under inaccurate probability distribution information. For more details on stochastic minimax program see texts, *e.g.*, [108] and reference therein.

Recall that Poisson-binomial distribution is characterized by an ordered set of failure rates of ith robot's neighbors, *i.e.*, $(p_j)_{j\in\mathcal{N}_i(t)}$. For convenience, we define a new symbol to define a set $\mathbb{P}_i := (p_j)_{j\in\mathcal{N}_i(t)}$. Then, each fault-free robot $i \in \overline{\mathcal{I}}$ solves at each stage $t \in \{0, 1, \ldots, \infty\}$ the following stochastic minimax problem[2]:

$$\begin{aligned}
&\underset{u_i(t)}{\text{minimize}} \left\{ \max_{\mathbb{P}_i} \left\{ \mathbb{E}_{\mathbb{P}_i}\Big[g_i(\widetilde{x}_i(t),\, F_i(t),\, u_i^k) \Big] + \alpha_v \underset{\mathbb{P}_i}{\mathbf{var}}\Big[g_i(\widetilde{x}_i(t),\, F_i(t),\, u_i^k) \Big] \right\} \right\} \\
&\text{subject to} \qquad \mu_f - \epsilon_f \leq \frac{\sum_{j\in\mathcal{N}_i(t)} p_j}{|\mathcal{N}_i(t)|} \leq \mu_f + \epsilon_f \\
&\qquad\qquad\quad a_j \leq p_j \leq b_j,\ \ j \in \mathcal{N}_i(t)
\end{aligned} \tag{5.20}$$

where a_j and b_j are numbers between $[0,\ 1]$ that is lower bound and upper bound for the failure rate of robot j respectively, μ_f is the average value of failure rate for all robots such that $0 \leq \mu_f \leq 1$, and ϵ_f is the error bound such that $0 \leq \epsilon_f \leq \min(\mu_f,\, 1 - \mu_f)$. We assume that these values μ_f, ϵ_f, a_j, b_j are known. We plan to show via simulation, how our proposed method handles the worst-case scenarios compared to from our previous method under same settings. Also, we plan to provide analytical results

[2]The form of the construct is similar to that found in [108].

that prove approximate rendezvous of our algorithm in the presence of random faults.

5.5.3 One step minimax solution: an example

In this section, we will obtain a one-step solution for (5.20) with a simple example. For any given $F_i(t) \subseteq \mathcal{N}_i(t)$ and $\widetilde{x}_i(t) \in \mathbb{R}^{d(|\mathcal{N}_i(t)|+1)}$ we define

$$\hat{x}_i(t) := \frac{1}{|\widetilde{x}_i(t)| - |F_i(t)| + 1} \left(x_i(t) + \sum_{\substack{x_j(t) \in \widetilde{x}_i(t), \\ \text{s.t. } j \notin F_i}} x_j(t) \right).$$

For convenience, we suppress t for the moment, and define the cost $C(u_i)$ to be an explicit function of u_i:

$$\begin{aligned} C(u_i) &:= \mathbb{E}\Big[g_i(\widetilde{x}_i,\, F_i,\, u_i)\Big] + \alpha_v \mathbf{var}\Big[g_i(\widetilde{x}_i,\, F_i,\, u_i)\Big] \\ &= \mathbb{E}\Big[\|\hat{x}_i - (x_i + u_i)\|^2\Big] + \alpha_v \mathbf{var}\Big[\|\hat{x}_i - (x_i + u_u)\|^2\Big]. \end{aligned}$$

We note that the following lemma holds.

Lemma 5.5.1. *$C(u_i)$ is convex.*

Proof. For the proof, we obtain the gradient $\nabla_{u_i} C(u_i)$ and Hessian $\mathbf{H}_{u_i} C(u_i)$.

$$\begin{aligned} \nabla_{u_i} C(u_i) = -2\mathbb{E}\Big[(\hat{x}_i - (x_i + u_i))\Big] + \alpha_v * \Big(-4\mathbb{E}\Big[(\hat{x}_i - (x_i + u_i))^3\Big] \\ + 4\mathbb{E}\Big[(\hat{x}_i - (x_i + u_i))\Big]\mathbb{E}\Big[\|\hat{x}_i - (x_i + u_i)\|^2\Big]\Big), \end{aligned}$$

and the Hessian $\mathbf{H}_{u_i} C(u_i)$ is

$$\begin{aligned} \mathbf{H}_{u_i} C(u_i) &= \left(2 + \alpha_v \left(8\mathbb{E}\Big[\|\hat{x}_i - (x_i + u_i)\|^2\Big] - 8\mathbb{E}\Big[(\hat{x}_i - (x_i + u_i))\Big]^2\right)\right)^\top \mathbf{I}_d \\ &= \Big(2 + 8\alpha_v \mathbf{var}\Big[(\hat{x}_i - (x_i + u_u))\Big]\Big)^\top \mathbf{I}_d \end{aligned}$$

where $\mathbf{1}_d$ is $d \times d$ identity matrix Since each variance term is non-negative, $\mathbf{H}_{u_i} C(u_i)$ is positive-semi definite for all u_i. Thus $C(u_i)$ is convex. ■

Now we are ready to present our simulation result. For this simulation, we consider $n = 4$ robots with identifiers 1, 2, 3, 4 initially deployed in a square workspace $[0, 1] \times [0, 1]$ as seen from Fig. 5.6.

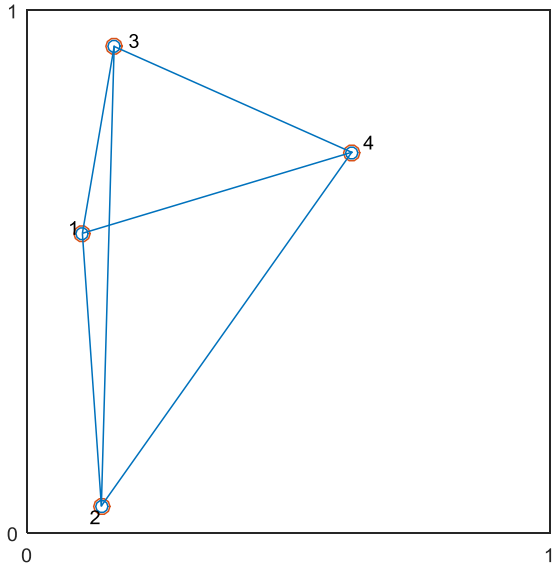


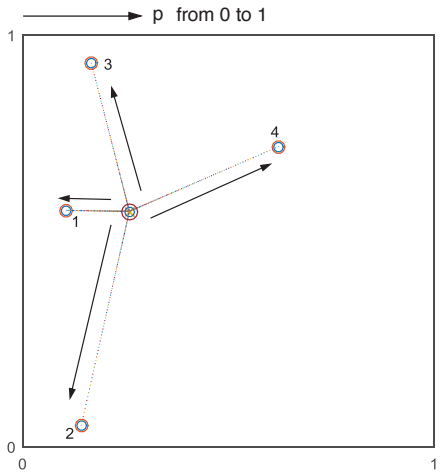
Figure 5.6: Initial configuration with 4 robots in $[0, 1]^2$.

We assume that the sensing range r and maximum velocity per stage $v_{\max}$ is unbounded.

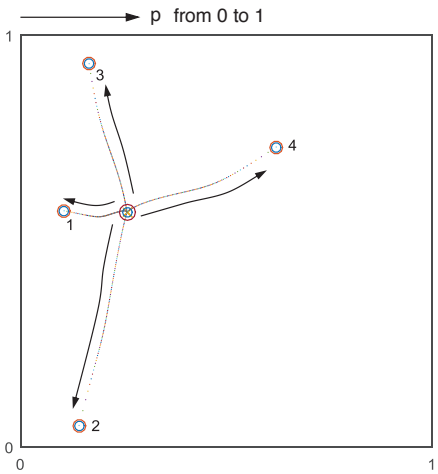
We learned from Lemma 5.5.1 that the cost function is convex with respect to u_i, however there is no direct relationship between the cost and p, thus we cannot directly obtain solution of (5.20). Instead we solve the minimization problem by varying p from 0 to 1 with increment of 0.01.

Fig 5.7 shows the one-step minimization solution with different α_v values. Obviously, when $p = 0$, *i.e.*, probability that each node fails is zero, every node will move to the coordinate average of the positions of all robots. Also, when $p = 1$, *i.e.*, probably that each node fails is one, every node will not move at all. Varying α_v from 0 to 1000 result in move *conservative behavior* in that each robot will not make a move until reaching relatively high value of p.

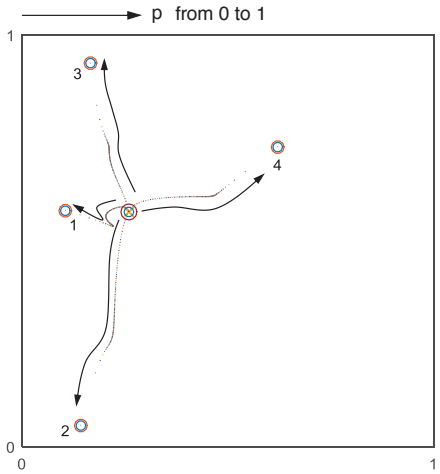
Fig 5.8 shows RDV(1) with respect to p from 0 to 1. The maximum RDV(1) appears near $p = 0.75$ when $\alpha_v = 0$ and shifting left toward $p = 0.58$ as we increased α_v. The value p which maximizes RDV(1) is not the same for all nodes, but very similar to each other. The minimax solution for this problem can be obtained by finding the value p at each peak.



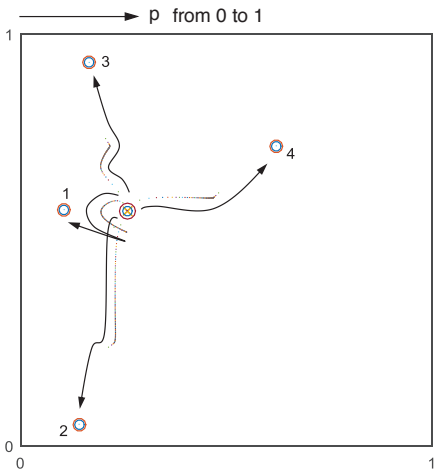
(a) $\alpha_v = 0$



(b) $\alpha_v = 10$

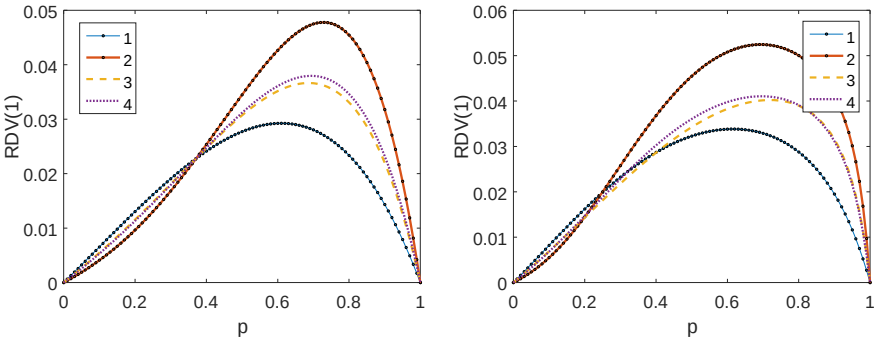


(c) $\alpha_v = 100$



(d) $\alpha_v = 1000$

Figure 5.7: One step optimization result when varying p from 0 to 1.



(a) $\alpha_v = 0$ (b) $\alpha_v = 10$

(c) $\alpha_v = 100$ (d) $\alpha_v = 1000$

Figure 5.8: RDV(1) with respect to p from 0 to 1.

5.6 Conclusion

In this study, we proposed an optimization-based control policy for fault-tolerant multi-robot rendezvous. We have provided a few numerical simulation results to show that our approach is statistically better than other rendezvous methods in the literature. Also, we proposed a minimax version of the problem and obtained solution for a simple example. Solutions for more complicated problems for MRS with large neighborhood size can be obtained via other numerical optimization methods, *e.g.*, Monte Carlo sample average approach [109–111].

Chapter 6

Related studies: multi-robot deployment

The multi-robot deployment problem studies control algorithms for multi-robot system that enable a group of robots to collectively maximize *Quality of Service* (QoS) that could be provided within a bounded space. The multi-robot deployment problem is closely related to the classical locational optimization problem [17, 112, 113] from operations research. The locational optimization problem is a spatial resource allocation problem where one is interested in finding the optimal resource allocation rule that maximizes some quantity of interest. A few examples of this include studying the optimal animal territory sharing method, optimal placement of mailboxes in a city, optimal design of vector quantizers for minimum distortion, optimal microphone placement for detection of sound sources, and optimal olfactory recepter placement. The solutions for those problems are typically obtained via mathematical programming [112].

Independent from these facility location problems, there are multi-robot deployment problems [22, 114–120]. The main objective for those studies is to propose *single time* deployment algorithms for *Mobile Sensor Networks*[1], which are optimal for specific interests, *e.g.*, Quality of Service (QoS), sensor reading performance, resource allocation efficiency, target detection probability, etc. These problems [22,116,118] are sometime called *coverage control* problems, and they show a sharp contrast to *coverage path-planning* problems [121–123], which aim to find the best area coverage algorithm for mobile robots that visit all points in a bounded space by sweeping the robots through the space.

Among various multi-robot deployment methods, we are interested in *Voronoi-based* coverage control algorithms [16, 114, 115, 124, 125] for multi-robot systems that utilize the classical *Voronoi diagram*. In these algorithms, each robot moves towards the centroid of its associated region (Voronoi region). It has been shown in a number of papers [20, 118, 126] that, if every robot/node/point updates its position with *Lloyd's method* [127], the collective QoS, *e.g.*, target detection performance, will be maximized. These algorithms are *decentralized* in that each robot only requires local data, *e.g.*, positions

[1] A multi-robot system where each robot is equipped with sensors with limited ad hoc sensing/communication capabilities

of other robots in the neighborhood, for the computation in the algorithm. More recently, based upon the above mentioned studies regarding Voronoi-based coverage control, there has been much research which studies more practical issues that could be found in the same topic [114, 124, 125]. Nowzari *et al.*, [124] studied a Voronoi diagram-based coverage control problem where individual agents operate with outdated information about each others' locations. Laventall and Cortés [125] assume that each robot is equipped with a limited-range anisotropic sensor, and present a solution to decentralized Voronoi coverage in non-convex polygonal environments. In their algoritm, each robot uses a combination of the Tangent bug algorithm [128] and Lloyd's algorithm for its control strategy. Kwok and Martinez [114] propose distributed coverage algorithms for mobile sensor networks in which agents have limited power to move. Their approach accounts for power constraints by assigning time-varying generalized Voronoi regions to each robot.

Independently of the above mentioned studies, there are partially decentralized approaches [129,130] for coverage control problems that assume joint target detection capability with multiple robots[2]. In [129], a probability distribution encodes the frequency of random events that can occur, and each mobile robot is equipped with a range limited sensor. Communication cost is considered as a limiting constraint, and a gradient-based algorithm that requires only local information from each sensor is proposed to locally maximize the joint-detection probabilities of the random events. In [130], the joint probability of missed detection is explicitly computed, and used to derive a gradient descent algorithm to minimize the total probability of missed detection, given the prior probability of individual sensor failure.

This chapter is organized as follows. In Section 6.1, we state the multi-robot deployment problem. In Section 6.2, we state a few definitions and introduce some notations which help our sequel presentation. Next we define a cost function based on the missed detection probability in Section 6.3. In Section 6.4, we formally revisit the discrete-time Lloyd's algorithm and its convergence properties using LaSalle's Invariance Principle in Section 6.5. Lastly, in Section 6.6, we discuss all the approaches used in the section.

6.1 The problem statement

The multi-robot deployment problem we are interested in this chapter can be formulated as a problem of minimizing the probability of missed detection of targets by a group of robots [130]. Let $x = (x_1, \ldots, x_n)$

[2]Multiple robots can cooperate in detecting each target.

be an ordered set of positions of n robots in some bounded Euclidean space $Q \subset \mathbb{R}^d$. Roughly stated, the multi-robot optimal deployment problem is given by,

$$\min_{x \in Q^n} P\{\text{missed detection of targets} \,|\, x\} \tag{6.1}$$

where the probability of missed detection is a function of positions of robots x that depends on the *distribution* of targets in the bounded space Q. Note that, we are interested in *distributed* motion control strategies that can be used to solve the problem (6.1). In other words, each robot $i = 1, \ldots, n$ solves a sub-problem, *i.e.*, a distributed counterpart, using only *local* resources, *e.g.*, position of its neighbors, and by doing so, all robots collectively solve the problem (6.1).

The problem formulation of (6.1) is useful when there is a possibility that some robots from a group could fail [130]. In most of the distributed deployment algorithms, non-intersecting regions are assigned to different robots [22,118] respectively. Thus, if some robots fail, targets in the regions associated with those faulty robots are left being uncovered/undetected. On the other hand, if we assume that *joint detection*[3] is available, then every target in the space Q is being monitored by at least one robot. The formulation of (6.1) could not only be used to model distributed deployment problems [118], but can handle more of the general cases of partially distributed deployment problems [130] that are designed for failure-prone systems. In this chapter, we describe typical distributed static deployment problems [118]. In Chapter 7, we propose a robust deployment strategy where multiple robots can jointly detect each target.

6.2 A few definitions

In this section, we review a few definitions related to the Voronoi diagram[4], and define the interconnection topology of a group of robots that will be used to define our problem in the sequel.

6.2.1 Voronoi diagram

Let us start this subsection, by reviewing the definition of the ordinary Voronoi diagram in Euclidean space $\mathbb{R}^d$.

[3] Two or more robots cooperatively detect each target.

[4] We use the terms Voronoi diagram, Voronoi tessellation, and Voronoi partition interchangeably throughout this chapter.

Definition 6.2.1 (The ordinary Voronoi diagram)**.** *Let $x = (x_1, \ldots, x_n)$ be an ordered set of positions of n distinct points, where $x_i \in \mathbb{R}^d$. We define by*

$$V_i(x) = \left\{ q \in \mathbb{R}^d \,\middle|\, \|q - x_i\| \leq \|q - x_j\| \; j \neq i,\, j \in \mathcal{I} \right\} \tag{6.2}$$

the ordinary Voronoi region associated with the generator position x_i, and the set given by

$$\mathscr{V}(x) = \{V_1(x), \ldots, V_n(x)\}, \tag{6.3}$$

the ordinary Voronoi diagram generated by $x = \{x_1, \ldots, x_n\}$.

It is known that there are algorithms with computational complexity of $O(n \log n)$ to obtain the Voronoi diagram given finite number of points [17,131]. We note that in the ordinary Voronoi diagram, some of the regions could be unbounded. There is a similar concept to ordinary Voronoi diagram when the domain upon which the diagram is defined is bounded, *i.e.*, there are no unbounded Voronoi regions, as follows:

Definition 6.2.2 (The ordinary Voronoi diagram bounded by Q [17])**.** *An ordinary Voronoi diagram,* i.e.*, partition, bounded by $Q \subset \mathbb{R}^d$ is defined by*

$$\mathscr{V}_{\cap Q}(x) = \{V_1(x) \cap Q, \ldots, V_n(x) \cap Q\} \tag{6.4}$$

where the subscript $\cap Q$ is used to denote the fact that Voronoi diagram is defined over the bounded space Q. If some Voronoi region V_j shares boundary with Q, i.e.*, $V_j \cap \partial Q \neq \emptyset$, we call V_j a* boundary Voronoi region*.*

In the sequel, if the position set x and the workspace Q are clear from the context, by dropping $\cap Q$ from the subscript and removing (x), we simply express the bounded Voronoi partition in (6.4) as $\mathscr{V} = \{V_1, \ldots, V_n\}$. Next, we define the dynamic Voronoi diagram, which is useful when positions of robots change w.r.t. time.

Definition 6.2.3 (Dynamic Voronoi diagram [132])**.** *Let $x(t) = (x_1(t), \ldots, x_n(t))$ be an ordered set of positions of n robots at time $t = 1, 2, \ldots$, where $x_i(t) \in Q \subseteq \mathbb{R}^d$. We call*

$$V_i(x(t)) = \{q \in Q \,|\, \|q - x_i(t)\| \leq \|q - x_j(t)\|\,,\; j \neq i,\; j \in \mathcal{I}\}\,, \quad t = 0,\, 1,\, 2 \ldots, \tag{6.5}$$

the dynamic Voronoi region of robot i at time t [132]. Also, we let $\mathscr{V}(x(t)) = \{V_1(x(t)), \ldots, V_n(x(t))\}$ be the dynamic Voronoi diagram generated by $x(t)$. For brevity, we may simply write $\mathscr{V}(x(t)) = \{V_1(x(t)), \ldots, V_n(x(t))\}$ as $\mathscr{V}(t) = \{V_1(t), \ldots, V_n(t)\}$.

6.2.2 The interconnection topology: A Delaunay graph

For the problem in the current chapter, the interconnection topology of a group of n robots with position set x is represented by the Delaunay graph denoted by $\mathcal{G}_{\mathrm{D}} = (\mathcal{V}, \mathcal{E})$. Then, given $i, j \in \mathcal{V}$ with $i \neq j$, $(i, j) \in \mathcal{E}$, if and only if $V_i \cap V_j \neq \emptyset$. Also, we say that robot i and robot j are *Voronoi neighbors* to each other if $(i, j) \in \mathcal{E}$.

6.3 A probabilistic cost function

In this section, we derive the probability that a set of n robots equipped with sensors at some configuration $x \in Q^n$ will fail to detect targets in a bounded space. Our derivation method follows that given in [130]. We define the target location to be a random vector $\mathrm{X} \in Q \subset \mathbb{R}^d$ with probability density function, *i.e.*, target distribution function, $\phi : Q \to \mathbb{R}_{\geq 0}$ such that $\int_Q \phi(q)\, dq = 1$. We denote by D_i, the event that robot i located at x_i detects the target, and by $\overline{D_i}$, the *complement* event that robot located at $x_i \in Q$ fails to detect the target. It is often found in the literature, *e.g.* [22,118], that for each sensor attached to a robot, the sensing, *i.e.*,the target detection, performance degrades with distance between the sensor and a target. In a similar manner, we use *non-decreasing*, *continuous* functions $f_i : \mathbb{R}_{\geq 0} \to [0,\ 1]$ associated with robot $i = 1, \ldots, n$ to model the sensor performance. In particular, given a robot located at x_i, and a target located at q, we use f_i to explicitly show the dependence of the missed-detection probability on the Euclidean distance $\|q - x_i\|$ between q and x_i as:

$$P\left\{\overline{D_i} \,\middle|\, \mathrm{X} = q\right\} := f_i(\|q - x_i\|). \tag{6.6}$$

Now let us consider the case when all n robots participate in detecting a target located at $q \in Q$, and let $\overline{D}$ be a collective event that no robot detects the target. Then, by assuming that the set of n events $\{\overline{D_1}, \ldots, \overline{D_n}\}$ are *mutually independent*, using (6.6), the conditional probability that all n robots fail to detect the target located at q, *i.e.*, $P\left\{\overline{D} \,\middle|\, \mathrm{X} = q\right\}$, is obtained with

$$P\left\{\overline{D} \,\middle|\, \mathrm{X} = q\right\} = \prod_{i=1}^{n} P\left\{\overline{D_i} \,\middle|\, \mathrm{X} = q\right\} = \prod_{i=1}^{n} f_i(\|q - x_i\|). \tag{6.7}$$

The total probability of missed detection is then

$$P\left\{\overline{D}\,\middle|\,\mathrm{X}\in Q\right\}=\int_{Q}P\left\{\overline{D}\,\middle|\,\mathrm{X}=q\right\}\,\phi(q)dq. \tag{6.8}$$

Since $\mathrm{X}\in Q$ from the left hand side of (6.8) implies all targets in workspace Q, we may remove it for simplicity. Then, if we partition Q into n regions $\{W_1,\ldots,W_n\}$ we may write $P(\overline{D})$ as

$$P\left\{\overline{D}\right\}=\sum_{i=1}^{n}\int_{W_i}\prod_{i=1}^{n}P\left\{\overline{D_i}\,\middle|\,\mathrm{X}=q\right\}\phi(q)\,dq. \tag{6.9}$$

For a special case often found in the literature [118] where each robot i can only detect targets in its associated region W_i , $P\left\{\overline{D}_i\,\middle|\,\mathrm{X}=q\right\}$ can be simply obtained by

$$P\left\{\overline{D}_i\,\middle|\,\mathrm{X}=q\right\}=\begin{cases}f_i(\|q-x_i\|), & \text{If } q\in W_i\\ 1. & \text{If } q\notin W_i\end{cases} \tag{6.10}$$

By applying (6.10) to (6.9), we obtain

$$P_1\left\{\overline{D}\right\}:=\sum_{i=1}^{n}\int_{W_i}f_i(\|q-x_i\|)\phi(q)\,dq \tag{6.11}$$

where we use subscript 1 in $P_1\left\{\overline{D}\right\}$ to emphasize the fact that each robot i detects only the target that lies in its associated region[5] W_i. In the sequel, it will be used to express (6.11) explicitly as a function of both the sensor locations x and the partition $\mathscr{W}$. For this purpose, we define

$$\mathcal{H}(x,\mathscr{W}):=P_1\left\{\overline{D}\right\}=\sum_{i=1}^{n}\int_{W_i}f_i(\|q-x_i\|)\phi(q)\,dq. \tag{6.12}$$

We note that if we let x' be set of positions of n robots, and $\mathscr{W}'$, $\mathscr{W}^\star$ be two different partitions of Q. Then, the equation $\mathcal{H}(x',\mathscr{W}')=\mathcal{H}(x',\mathscr{W}^\star)$ is not necessarily true. In other words, even with an identical configuration of a group of robots, the cost need not be the same for different partitioning methods of a workspace Q into n regions.

[5]For the case in Chapter 7 where k robots are used to detect target in some properly defined region, *e.g.*, order$-k$ region, we use subscript k instead, *i.e.*, $P_k\{\overline{D}\}$.

6.4 The deployment algorithm: a discrete time Lloyd's algorithm

In this section, we formally define a few terms, then introduce the discrete time Lloyd's algorithm for a static-deployment of a group of robots over a bounded space. We provide a few propositions to show that the cost (6.12) is a non-increasing function along the evolution of the algorithm. Then, using LaSalle's Invariance principle [49], we show that all robots evolving with Lloyd's algorithm will converge to a point set which forms a special type of tessellation named as the Centroidal Voronoi tessellation (CVT) [126].

6.4.1 Discrete-time Lloyd's algorithm

In this subsection, we introduce the discrete-time Lloyd's algorithm for static deployment of a group of robots that is based upon Lloyd's method [127]. Lloyd's method was originally used for least squares quantization in Pulse-Code Modulation (PCM). According to Du *et al.*, [126], Lloyd's method [127] is one of the fixed point iteration algorithms which has two major stages per each algorithm step. In the first stage, a Voronoi diagram with current point set is obtained, and centroids for the Voronoi regions are calculated. In the second stage, current points are updated with the computed centroids of their Voronoi regions. For more details of the algorithm, see *e.g.* [17, 133].

Before we proceed further, we formally define the centroid of each robot's Voronoi region. Given $n-$tuple of points $x = (x_1, \ldots, x_n)$ in a bounded space Q with $x_i \in Q$, the Voronoi partition of the workspace Q given the configuration x is given as a set $\mathscr{V} = \{V_1, \ldots, V_n\}$. Let the mapping $C_{V_i} : Q^n \to Q$, $i = 1, \ldots, n$ be defined by

$$C_{V_i}(x) = \frac{\int_{V_i} q\, \phi(q)\, dq}{\int_{V_i} \phi(q)\, dq}, \tag{6.13}$$

which maps a point set x to the centroid of Voronoi region V_i for each $i = 1, \ldots, n$. Recall that $\phi : Q \to \mathbb{R}_{\geq 0}$ is the target distribution function. It will be shown rigorously in the sequel that the map $C_{V_i} : Q^n \to Q$ is a continuous mapping away from the degenerate points where distinct points collapse[6] [134].

[6]Roughly speaking, the map C_{V_i} is continuous on Q^n except for a set of points where two or more points are at a same location.

The discrete-time Lloyd's algorithm is an application of Lloyd's method for mobile sensor networks; at each iteration step, every robot moves towards the centroid of its Voronoi region within its maximum travel distance. Let $v_{\max} > 0$ be the maximum travel distance for every robot during one stage. In the discrete-time Lloyd's algorithm, at stage t each robot i evolves with the following equation.

$$x_i(t+1) = x_i(t) + \xi_i(t)(C_{V_i}(x(t)) - x_i(t)), \quad t = 0,\, 1,\, 2, \ldots, \tag{6.14}$$

where

$$\xi_i(t) = \begin{cases} 1, & \text{If } \|C_{V_i}(x(t)) - x_i(t)\| \le v_{\max} \\ \frac{v_{\max}}{\|C_{V_i}(x(t)) - x_i(t)\|}. & \text{Otherwise} \end{cases} \tag{6.15}$$

The discrete-time Lloyd's algorithm (6.14-6.15) can be represented by a single-valued map $T^{\mathrm{dl}} : Q^n \to Q^n$ where its component map for robot i, $T_i^{\mathrm{dl}} : Q^n \to Q$ is defined by

$$T_i^{\mathrm{dl}}(x(t)) = x_i(t) + \xi_i(t)(C_{V_i}(x(t)) - x_i(t)), \tag{6.16}$$

where $\xi_i(t)$ is obtained from (6.15).

6.4.2 Two important propositions

In this subsection, we state two propositions that will be used to depict the properties of the discrete-time Lloyd's algorithm. First we assume that every robot has an identical target detection performance.

Assumption 6.4.1. *We assume that function $f_i : \mathbb{R}_{\ge 0} \to [0,\ 1]$ for $i = 1, \ldots, n$ is identical for all robots,* i.e., *$f_1 = \cdots = f_n$.*

Unless otherwise noted, assumption 6.4.1 will be applied to all theorems, propositions that will be stated in the sequel. In the following proposition, we show that the Voronoi partition is a locally optimal partitioning scheme with respect to the cost function (6.12).

Proposition 6.4.1 (Cortes *et al.*, [118], Du *et al.*, [126])**.** *Let $x = (x_1, \ldots, x_n)$ be an ordered set of positions of n distinct robots where $x_i \in Q \subset \mathbb{R}^d$. Then, given the Voronoi partition $\mathscr{V}(x) = \{V_1(x), \ldots, V_n(x)\}$ of Q, for each partition $\mathscr{W} = \{W_1, \ldots, W_n\}$ of Q, the following inequality holds:*

$$\mathcal{H}(x, \mathscr{V}(x)) \le \mathcal{H}(x, \mathscr{W}). \tag{6.17}$$

The equality holds only if $\mathscr{V}(x) = \mathscr{W}$. In other words, the Voronoi partition is the minimizing partitioning scheme w.r.t. the cost function provided in (6.12).

Proof. For convenience, let $\mathscr{V} := \mathscr{V}(x)$, and $V_i := V_i(x)$ for each $i = 1, \ldots, n$. Let $\mathcal{J}$ be the collection of all possible subsets of $\mathcal{I} = \{1, \ldots, n\}$. Then the cardinality of the set $\mathcal{J}$ is $|\mathcal{J}| = 2^n$. Given the generator point set $x \in Q^n$, any partition $\mathscr{W} = \{W_1, \ldots, W_n\}$ of Q, and for each $i = 1, \ldots, n$ with its Voronoi region $V_i \in \mathscr{V}$, we may find some element, which itself is a set, $J^* \in \mathcal{J}$, that is the solution to the following problem:

$$\begin{aligned} \min_{J \in \mathcal{J}} \quad & |J| \\ \text{subject to} \quad & V_i \subseteq \bigcup_{j \in J} W_j. \end{aligned}$$

By solving the problem for each $i = 1, \ldots, n$, we obtain the smallest number of regions in $\{W_j\}_{j=1}^n$ whose union includes V_i. By using the definition of ordinary Voronoi diagram from definition 6.4 and assumption 6.4.1, the following inequality can be obtained.

$$\int_{V_i} f_i(\|q - x_i\|)\phi(q)\, dq \leq \sum_{j \in J^*} \int_{V_i \cap W_j} f_j(\|q - x_j\|)\phi(q)\, dq, \quad i = 1, \ldots, n. \tag{6.18}$$

We note that except for the trivial case $|J^*| = 1$ with $W_j = V_i$, the inequality in (6.18) is *strict* for each $i = 1, \ldots, n$. We may apply summation from 1 to n over both hand sides of (6.18) to obtain:

$$\begin{aligned} \sum_{i=1}^n \int_{V_i} f_i(\|q - x_i\|)\phi(q)\, dq &\leq \sum_{i=1}^n \sum_{j \in J^*} \int_{V_i \cap W_j} f_j(\|q - x_j\|)\phi(q)\, dq \\ &= \sum_{h=1}^n \int_{W_h} f_h(\|q - x_h\|)\phi(q)\, dq, \end{aligned}$$

which implies that $\mathcal{H}(x, \mathscr{V}) \leq \mathcal{H}(x, \mathscr{W})$ holds as required. We note that the equality holds only when $\mathscr{V} = \mathscr{W}$. ■

According to Proposition 6.4.1, the Voronoi partition is a minimizing partitioning scheme with respect to the cost function (6.12). In the following definition, we introduce a special partition type called centroidal Voronoi tessellation (CVT).

Definition 6.4.1 (A Centroidal Voronoi Tessellation (CVT) [126])**.** *Let $x = (x_1, \ldots, x_n)$ be an ordered set of positions of n distinct robots where $x_i \in Q \subset \mathbb{R}^d$. We call a Voronoi partition $\mathscr{V} = \{V_1, \ldots, V_n\}$ the centroidal Voronoi tessellation (CVT) if, $x_i = C_{V_i}(x)$ holds for all $i = 1, \ldots, n$.*

In the next proposition, we show that CVT is the special type of partition of Q which has an optimizing property with respect to (6.12).

Proposition 6.4.2 (Cortes *et al.*, [118], Du *et al.*, [126]). *Let $x = (x_1, \ldots, x_n)$ be an ordered set of positions of n robots where $x_i \in Q \subset \mathbb{R}^d$. A group of n robots execute the discrete-time Lloyd's algorithm that is defined by a map $T^{\mathrm{dl}} : Q^n \to Q^n$ whose component map T_i^{dl} for $i = 1, \ldots, n$ is provided in (6.15-6.16). For all position sets x that are distinct[7], the following inequality holds:*

$$\mathcal{H}(T^{\mathrm{dl}}(x), \mathscr{V}(x)) \leq \mathcal{H}(x, \mathscr{V}(x)) \tag{6.19}$$

where the equality in (6.19) holds only if $x_i = C_{V_i}(x)$ for all $i = 1, \ldots, n$.

The proposition states that the cost function is non-increasing along the trajectory generated by Lloyd's algorithm T^{dl}, and the configuration that forms the CVT is the critical point for the cost function (6.12). The proof of the proposition uses the *Parallel Axis Theorem*, and the complete proof is provided in [118, 126].

6.5 Convergence of the discrete-time Lloyd's algorithm

In this section, we show the correctness of the discrete-time Lloyd's algorithm. Here by correctness, we mean the convergence of the algorithm to a locally optimal configuration which forms the CVT of the workspace.

To be used in the sequel, we define another cost function $\mathcal{L} : Q^n \to \mathbb{R}_{\geq 0}$ that shows explicit dependence on the configuration x, when the underlying partition is the Voronoi partition $\mathscr{V}(x)$. $\mathcal{L} : (\mathbb{R}^d)^n \to \mathbb{R}_{\geq 0}$ is defined by:

$$\mathcal{L}(x(t)) := \mathcal{H}(x(t), \mathscr{V}(x(t))), \quad t = 0,\, 1,\, 2, \ldots \tag{6.20}$$

Next, we show that our cost function $\mathcal{L} : (\mathbb{R}^d)^n \to \mathbb{R}_{\geq 0}$ bears the following property.

Proposition 6.5.1 (Du *et al.* [126, 134], Cortes *et al.* [47, 118]). *The function $\mathcal{L}$ is continuous and non-increasing along the map T^{dl},* i.e.*,*

$$\mathcal{L}(T^{\mathrm{dl}}(x(t))) \leq \mathcal{L}(x(t)), \quad t = 0,\, 1,\, 2, \ldots. \tag{6.21}$$

[7] No two points in x coincide.

The equality holds only if $x(t)$ forms a CVT.

Proof. First by the definition of the function $\mathcal{H}(x, \mathscr{W})$ in (6.12) and the continuity of the functions f_i for $i = 1, \ldots, n$ on their domains, it can be verified that the function $\mathcal{L}$ is continuous on Q^n. Next, by Proposition 6.4.2, for each $t = 0, 1, 2, \ldots,$

$$\mathcal{H}(T^{\mathrm{dl}}(x(t)), \mathscr{V}(x(t))) \leq \mathcal{H}(x(t), \mathscr{V}(x(t))) \tag{6.22}$$

holds. By Proposition 6.4.1, given the generator point $p \in Q^n$, the Voronoi partition $\mathscr{V}(p)$ of p is the partition minimizing the function $\mathcal{H}(q, \mathscr{V}(q))$. Using the property,

$$\mathcal{H}(T^{\mathrm{dl}}(x(t)), \mathscr{V}(T^{\mathrm{dl}}(x(t)))) \leq \mathcal{H}(T^{\mathrm{dl}}(x(t)), \mathscr{V}(x(t))) \leq \mathcal{H}(x(t), \mathscr{V}(x(t))), \tag{6.23}$$

holds. Thus by the above inequality (6.23), $\mathcal{H}(T^{\mathrm{dl}}(x(t)), \mathscr{V}(T^{\mathrm{dl}}(x(t)))) \leq \mathcal{H}(x(t), \mathscr{V}(x(t)))$ and by (6.20) holds, which implies

$$\mathcal{L}(T^{\mathrm{dl}}(x(t))) \leq \mathcal{L}(x(t)), \quad t = 0, 1, 2, \ldots \tag{6.24}$$

as required. By Proposition 6.4.2, equality holds only at the critical points that are exactly the configurations that form a CVT. ■

We note that given n robots if two or more robots are at a same location, according to Definition 6.2.2, it is not possible to define a Voronoi diagram with n Voronoi regions. We must exclude such co-location events to ensure the continuity of the map $C_{V_i} : Q^n \to Q$ defined in (6.13). Before doing so, we formally define the notion of degeneracy of a point set.

Definition 6.5.1 (A degenerate point set)**.** *A multiset of n points $x = [x_1, \ldots, x_n] \in Q^n \subseteq (\mathbb{R}^d)^n$ is called* degenerate *if any two or more points in x are at the same location. Otherwise, the set is* non–degenerate*.*

Let $\widetilde{Q}^n$ be a $n-$tuple of points that are non-degenerate, *i.e.*,

$$\widetilde{Q}^n := \{q = (q_1, \ldots, q_n) \in Q^n \mid q_i \neq q_j,\, i \neq j,\, i, j \in \mathcal{I}\}. \tag{6.25}$$

Then, we note that the following proposition holds.

Proposition 6.5.2 (Du *et al.* [126,134], Cortes *et al.* [47,118])**.** *The map T is continuous on $\widetilde{Q}^n$.*

Next, we define a set E by

$$E := \{x' \in \widetilde{Q}^n \subset (\mathbb{R}^d)^n \mid \mathcal{L}(T^{\mathrm{dl}}(x')) = \mathcal{L}(x')\}. \tag{6.26}$$

We note that E is the non-degenerate set of critical points of $\mathcal{L}$ that is exactly the non-degenerate set of $n-$tuple points that form CVTs.

According to Proposition 3.5 in [126], each limit point in the trajectory generated by Lloyd's algorithm in any dimensional space is non-degenerate, provided that a few general assumptions[8] are satisfied. Based upon the non-degeneracy of limit points of trajectories generated by Lloyd's algorithm, and the definition of the discrete-time Lloyd's algorithm given in (6.14-6.15), we can ensure that the following proposition holds:

Proposition 6.5.3. *If initial positions of n robots are distinct,* i.e., $x(0) \in \widetilde{Q}^n$, *then every $n-$tuple point set in the trajectory $(x(t))_{t=0}^{\infty}$ of T is non-degenerate and every point along the trajectory is bounded.*

We are ready to state the main result in the following theorem which states that a group of robots executing the discrete-time Lloyd's algorithm will converge to a distinct, *i.e.*, non–degenerate, point set that forms a CVT.

Theorem 6.5.1 (Convergence of discrete-time Lloyd's algorithm: Cortes *et al.* [47, 118], Du *et al.* [134])**.** *Let $x(t) = [x_1(t), \ldots, x_n(t)]$ be the multiset of positions of n robots at time $t \in \mathbb{Z}_{\geq 0}$ where $x_i(t) \in Q \subset \mathbb{R}^d$. Suppose that Q is bounded and convex, and the initial positions of the n robots are distinct. If all robots evolve with the discrete-time Lloyd's algorithm defined by a map $T^{\mathrm{dl}} : Q^n \to Q^n$ (6.15-6.16), then the set of positions of the n robots will converge to a distinct $n-$tuple of points that forms a CVT.*

Proof. The idea of our proof (*i.e.*, the ideas contained in the proofs of previous propositions 6.5.1, 6.5.2, 6.5.3 and the idea of using the LaSalle Invariance principle, Theorem D.4.1), is similar to the proof of Proposition 3.4 [47], Theorem 2.6 [134], and Proposition 3.3 [118]. From previous propositions, we note that the following are true.

- According to Proposition 6.5.1, the function $\mathcal{L}$ is *continuous* on Q^n, and is *non-increasing* along the map T^{dl} on Q^n. This implies that the function $\mathcal{L}$ is continuous on $\widetilde{Q}^n$, and is non-increasing along the map T^{dl} on $\widetilde{Q}^n$.

[8] The workspace Q must be convex, the function ϕ must be integrable.

- According to Proposition 6.5.2, the map T^{dl} is *continuous* on $\widetilde{Q}^n$.
- According to Proposition 6.5.3, every $n-$tuple point set along the trajectory $(x(t))_{t=0}^{\infty}$ of T^{dl} is *non-degenerate* and *bounded*.

If we define M to be the largest positively invariant set contained in $E = \{x' \in \widetilde{Q}^n \subset (\mathbb{R}^d)^n \mid \mathcal{L}(T^{\mathrm{dl}}(x')) = \mathcal{L}(x')\}$, then, using LaSalle's Invariance Principle (Theorem D.4.1) with $\mathcal{L}$ as a Lyapunov function, it can be guaranteed that there is some $c \in \mathbb{R}$ such that the trajectory $(x(t))_{t=0}^{\infty}$ of T^{dl} approaches to $M \cap \mathcal{L}^{-1}(c)$. Since E is the set of critical points of $\mathcal{L}$, M must be contained in the set of non-degenerate points that form CVTs.

Next, we show that the entire sequence $(x(t))_{t=0}^{\infty}$ converges to a point. Since every point in the trajectory $(x(t))_{t=0}^{\infty}$ of T^{dl} in $\widetilde{Q}^n$ is bounded, there is a subsequence $(x[t_i])_{t_i=0}^{\infty}$ such that $t_i \to \infty$ and $x[t_i] \to x^\star$ where $x^\star \in M$. Now using the fact that the function $\mathcal{L}$ is continuous, and non-increasing along T^{dl} on $\widetilde{Q}^n$, it can be shown that the entire sequence converges to $x^\star$ that forms a CVT. ■

6.6 Conclusion

In this chapter, we reviewed one of the distributed static deployment algorithms where each robot chooses its sensing region to be a Voronoi region, and executes Lloyd's algorithm to move toward the centroid of the Voronoi region. This approach has been extremely popular in the literature of mobile sensor networks [16, 114, 115, 124, 125], because of the fact each Voronoi region can be calculated using only local information by each robot, and Lloyd's algorithm guarantees convergence of positions of the robots to one of the locally optimal configurations (that form CVTs) given a certain cost function related to Quality of Service (QoS). Thus, the approach is *distributed*, *optimal*, and *correct*[9]. Based upon these results, in Chapter 7, we propose a static deployment algorithm that is robust to random sensor failures by endowing robots with joint target detection capabilities that were discussed in [130].

[9]In this context, Lloyd's algorithm is correct in the sense that robots' positions *converge* to equilibrium points that form a CVT.

Chapter 7

Multi-robot robust deployment algorithm

In recent years, mobile sensor networks (MSNs) have found increasing applicability for problems including surveillance, search and rescue mission, natural disaster forecast, animal habitat monitoring, exploration of hazardous environment (see, e.g., the surveys given in [135–137]). Many of these applications require sensor deployment in hostile environments, which can lead to failure of individual nodes in the network. In this chapter, we address the problem of sensor failure by developing methods for robust sensor deployment.

Much research in the MSN literature has been devoted to the problem of controlling sensor movement such that the sensor nodes maintain maximum coverage of their regions of interest. This coverage control problem is closely related to the locational optimization problem [17, 112, 126], which deals with the optimal placement of resources in a spatial domain. Various distributed motion control algorithms have been presented that drive the nodes to their optimal positions (see, e.g., [22, 115, 118, 129]), however, these algorithms have not typically considered the possibility of sensor failure, and converge to optimal sensor configurations under the assumption that all sensors function properly and accurately implement the control algorithm. With such approaches, failure of a single sensor can lead to network failure, for example, in the problem of target detection.

One of the primary reasons that the failure of a single node can be so significant is that many distributed MSN algorithms simplify communication and computation requirements by partitioning the workspace into regions, and then assigning only one node per region. Thus, failure of a single node results in one region that is not covered by the network.

A number of approaches have been proposed in which multiple sensors can work cooperatively to achieve coverage, including [129, 130]. In [129], a probability distribution encodes the frequency of random events that can occur, and each mobile node is equipped with range limited sensor. Communication cost is considered as a limiting constraint, and a gradient-based algorithm that requires only local information to each sensor is proposed to locally maximize the joint-detection probabilities of the

random events. In [130], the joint probability of missed detection is explicitly computed, and used to derive a gradient descent algorithm to minimize the total probability of missed detection, given the prior probability of individual sensor failure.

A closely related problem to ours from computer science is the $k-$*coverage problem* [138–142], which is typically aimed for intruder detection applications. The objective of a general $k-$coverge problem is to place minimum number of sensors in a bounded area such that every target in the area is covered by at least k sensors. The $k-$coverage problem is similar to ours in that both pursue robust coverage, i.e., more than a one sensor detects every target. However there are fundamental differences between the two problems. First, in the $k-$coverage problem, it is assumed that there are *redundant* sensors available for a bounded area. Thus, the major concern is to decide which sensors to use, and which sensors to put to sleep to minimize energy consumption. On the other hand, in our case, the number of sensors is not free to choose as in $k-$coverage problem. Instead, the number of sensors is given, and the goal is to find an optimal configuration/motion control strategy for the sensors that maximizes collective target detection performance under the assumption that each target in the workspace is being covered by k sensors. Second, while in the $k-$coverage problem it is typically assumed that there are a finite number of targets in the workspace and that sensors are deployed in a grid, in our problem, the workspace is *continuous* such that targets are distributed over the space, and sensors are not contained to move on a grid.

While most of above mentioned studies assumes that sensor locations are exact, there are exceptions. In [143], Vu and Zheng modeled uncertainty in sensor locations by disk areas with possibly different radius centered at their nominal locations. They introduced the concept of order$-k$ max Voronoi diagram[1] in order to determine minimum sensing radius needed to ensure the worst-case $k-$coverage. Pierson et al., [146, 147] proposed an adaptive, distributed on-line algorithm for multi-robot coverage. In their algorithm, which bases on the classical results of [47], each node uses weighting adaptation law which assigns weights on their neighbors by comparing their relative performance (degradation occurs due to errors in sensor readings [147] or actuation errors [146]). Lyapunov-like proof was used to show the convergence of their algorithm to an invariance set. As it was discussed in the conclusion from [146], their current algorithm is guaranteed to work in the presence of robots with simple actuation or sensor errors, but not with malicious or faulty robots. We will try to address such issues in this work.

This chapter expands upon the method presented in [130]. As with most previous approaches, we

[1]Their concept of order$-k$ max Voronoi diagram is based on the order$-k$ Voronoi diagram [144, 145]. Additionally, they make use of sensing radii of sensors in their definition.

partition the workspace into regions. Unlike those approaches, we consider the case in which k sensors with exact locations are assigned to guard each region in the partition. Thus, for any region in the partition, if one sensor fails, $k-1$ sensors remain functional. By varying the choice of k, we obtain the classical approach when $k = 1$, and the case of full workspace coverage (the case considered in [130]) when $k = n$, for a network of n sensors. Values of k from 2 to $n-1$ provide successively more accurate approximations to the full coverage (i.e., complete knowledge of the workspace) at the expense of increasing computation and communication requirements. For $k > 1$, our method is robust in the sense that even if one sensor fails, there are other sensors that can successfully detect the target.

The main results in this chapter are as follows. First, we show that the order$-k$ Voronoi partition is the optimal partition of the workspace when each sensor is assigned to those order$-k$ Voronoi regions for which it is a generator (i.e., when the order$-k$ Voronoi region for each sensor is the set of all points in the workspace for which it is one of the k nearest sensors). We provide conditions for the optimal configurations at which each sensor is located. This is a generalization of the classical result for the problem where $k = 1$, in which case the Centroidal Voronoi tessellation is known to be optimal, and the optimal configuration is such that each sensor is positioned at the centroid of its Voronoi regions. Finally, we present a distributed algorithm for our optimal sensor placement problem whose convergence property is proven.

One benefit to our approach is that it allows communication and computation costs to be considered against the relative performance gains as k ranges from 1 to n, and as computation ranges from fully decentralized to fully centralized. For $1 < k < n$, our approach is regarded as *decentralized*, however not *fully decentralized* in that additional local communication/sensing could be necessary.

The remainder of the chapter is organized as follows. A few notations and terminologies are introduced in Section 7.1. We begin the Section 7.2 by formulating an appropriate cost function that corresponds to the probability of missed detection for order$-k$ redundancy in sensing. In Section 7.2.1, we introduce generalized Voronoi partition, and show that the order$-k$ Voronoi partition is the optimal partition. In Section 7.2.2, given any order$-k$ partition, we provide conditions for configurations that minimize the cost function. Then, in Section 7.3 we introduce a robust deployment algorithm and show its convergence properties via *LaSalle's Invariance Principle* tailored for set-valued maps. Simulation results with $k = 2$ are shown in Section 7.4. Section 7.5 concludes this chapter by presenting a number of future directions.

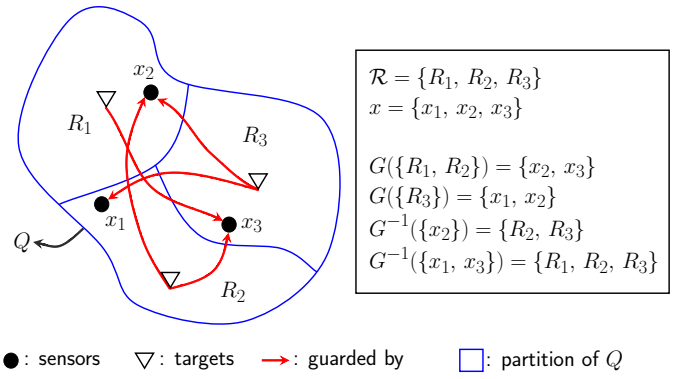


Figure 7.1: The mapping G.

7.1 Preliminaries

7.1.1 Workspace partitioning

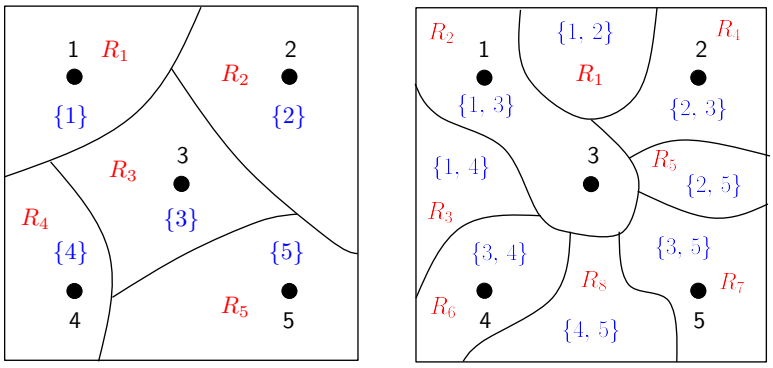
Let $\mathcal{R} = \{R_1, \ldots, R_m\}$ be a partition of Q, into m *disjoint* regions. Due to the definition of a *partition* of a bounded space resulting regions are disjoint, and union of those regions complete the space, we have $Q = \bigcup_{i=1}^{m} R_i$, where $R_i \cap R_j = \emptyset$ for every pair $i, j \in \{1, \ldots, m\}$ with $i \neq j$.

7.1.2 $k-$redundant coverage

In this chapter, we consider the multi-agent *target detection* problem where every mobile sensor *guards* some particular region in Q, which we will call *guarded region* throughout the text. In other words, each robot has its own associated region where the robot will always detect targets in the region with some positive probability. Thus, our model generalizes *Voronoi sensing* [118] where each sensor has its own Voronoi region whose generator is exactly the robot's position.

For a given partition $\mathcal{R}$ of a bounded space $Q \subset \mathbb{R}$ into m regions, and set of n sensors x in Q, let $G : 2^{\mathcal{R}} \to 2^{x}$ be a map that assign a set of regions to a set of sensors. We assume that the *inverse map* G^{-1} always exists. Then, given $l \in \mathbb{N}$ with $l \leq m$ regions $R_{i_1}, \ldots, R_{i_l} \subset Q$, $G(\{R_{i_1}, \ldots, R_{i_l}\})$ gives the set of sensors that guard those l regions. Inversely, given $h \leq n$ sensors $x_{i_1}, \ldots, x_{i_h} \in Q$, $G^{-1}(\{x_{i_1}, \ldots, x_{i_h}\})$ give the set of regions that are guarded by those h sensors. An illustrative example of the map G is given in Fig. 7.1.

To add robustness to sensor failures, we propose a redundant coverage method where each region from a partition is guarded by exactly $k \in \mathbb{N}$ (where $1 \leq k \leq n$) number of sensors. The notion is summarized below.



(a) k=1 (b) k=2

Figure 7.2: The order$-k$ partitions of a square workspace.

Definition 7.1.1 ($k-$redundant coverage)**.** *Consider a bounded space $Q \in \mathbb{R}^d$ where target is continuously distributed over Q, and n sensors are deployed in Q.* $k-$redundant coverage *is special type of sensor coverage where every target is guarded by exactly k sensors.*

If necessary, we will use the term *order$-k$ region* to denote the region guarded by exactly k sensors, and the *order$-k$ partition* for the partition where every region is order$-k$.

Fig. 7.2 shows another examples of order$-k$ partitions of a bounded square workspace. Fig. 7.2(a) shows an example of with $k = 1$, and Fig. 7.2(b) shows an example with $k = 2$. The order-1 partition in Fig. 7.2(a) can be expressed as $\mathcal{R} = \{R_1, \ldots, R_5\}$ each is guarded by robot with index $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5\}$ respectively. In a similar manner, the order-2 partition in Fig. 7.2(b) can be expressed as $\mathcal{R} = \{R_1, \ldots, R_8\}$ each is guarded by a pair of robots with indices $\{1, 2\}$, $\{1, 3\}$, $\{1, 4\}$, $\{1, 5\}$, $\{2, 3\}$, $\{2, 4\}$, $\{2, 5\}$, $\{3, 4\}$, $\{3, 5\}$, $\{4, 5\}$ respectively. We note that there are no regions guarded by a pair of robots with indices $\{1, 4\}$ nor $\{2, 4\}$.

7.1.3 Additional notations

For a given target, we denote by D_i, the event that robot positioned at x_i detects the target, and by $\overline{D_i}$, the *complement* event that the robot positioned at x_i fails to detect the target. Consider case when all n robots participate in detecting a particular target located at $q \in Q$. In this case, we denote by $\overline{D}$, the collective event that no robots detects the target. We define the target location to be a random vector $\mathrm{X} \in Q \subset \mathbb{R}^d$ with probability density function (PDF), i.e., target distribution function, $\phi : Q \to \mathbb{R}_{\geq 0}$

where

$$\int_Q \phi(q)\, dq = 1.$$

We may now formulate a function for the probability of missed target detection given a specific partition $\mathcal{R}$, set of sensor locations x, and mapping of regions to sensors G.

7.1.4 Probability of missed-detection revisited

In this section, we derive the probability that a set of n robots equipped with sensors at some configuration x each $x_i \in Q \subset \mathbb{R}^d$. This time, we will take into account the mapping G. Our derivation method follows closely to that given in [130].

We consider the *probabilistic sensor detection model*. The conditional probability $P\left\{\overline{D_i} \,\middle|\, \mathrm{X} = q\right\}$ measures the missed-detection probability when a target q is at some distance from the sensor located at x_i. The model reflects the behavior of range sensing devices such as infrared and ultrasound sensors. We will assume that n events $\overline{D_1}, \ldots, \overline{D_n}$ are *mutually independent* such that the conditional probability that all n robots fail to detect a specific target located at q is obtained by

$$P\left\{\overline{D} \,\middle|\, \mathrm{X} = q\right\} = \prod_{i=1}^{n} P\left\{\overline{D_i} \,\middle|\, \mathrm{X} = q\right\}.$$

Hence, the total probability of missed detection targets distributed over Q by n robots is

$$\begin{aligned} P\left\{\overline{D}\right\} := P\left\{\overline{D} \,\middle|\, \mathrm{X} \in Q\right\} &= \int_Q P\left\{\overline{D} \,\middle|\, \mathrm{X} = q\right\} \phi(q)\, dq \\ &= \int_Q \prod_{i=1}^{n} P\left\{\overline{D_i} \,\middle|\, \mathrm{X} = q\right\} \phi(q)\, dq \end{aligned}$$

where we omitted $\mathrm{X} \in Q$ on the left hand side for simplicity (provided that Q is obvious from the context). If Q can be partitioned into, let's say, m dis-joint regions, then we can break the integral into a sum of integrals, each taken over a disjoint region among the partition $\mathcal{R}$ as

$$P\left\{\overline{D}\right\} = \sum_{j=1}^{m} \int_{R_j} \prod_{i=1}^{n} P\left\{\overline{D_i} \,\middle|\, \mathrm{X} = q\right\} \phi(q)\, dq.$$

For a special case where $m = n$, and each robot at x_i has exactly one and only region $R_i \in \mathcal{R}$ to guard, $P\left\{\overline{D}\right\}$ is given by

$$P\left\{\overline{D}\right\} = \sum_{i=1}^{n} \int_{R_i} P\left\{\overline{D_i} \,\middle|\, \mathrm{X} = q\right\} \phi(q)\, dq.$$

which takes exactly the same form to that found in [Du, Cortes, etc], and it is used as a performance measure for fully decentralized deployment of MSNs, or as the distortion of vecter quantizers.

In particular, under $k-$redundant coverage, where each region is guarded by exactly k sensors, we have

$$P\left\{\overline{D}\right\} = \sum_{j=1}^{m} \int_{R_j} \prod_{x_i \in G(R_j)} P\left\{\overline{D_i} \,\middle|\, \mathrm{X} = q\right\} \phi(q)\, dq.$$

Note that in the product term of the RHS, only sensors that guard the region R_j are considered. As the measure for the likelihood that a sensor positioned at x_i fail to detect target located at q, we use twice differentiable, continuous, non-decreasing function $f_i : \mathbb{R}_{\geq 0} \to [0,\, 1]$ such that

$$P(\overline{D_i} \mid \mathrm{X} = q) = \begin{cases} f_i(\|q - x_i\|), & \text{if } q \in G^{-1}(x_i) \\ 1. & \text{otherwise} \end{cases} \tag{7.1}$$

Generally, for the multi-sensor coverage/deployment problem (see e.g., [47, 118, 126], etc), the likelihood that some target will be detected by a sensor has inverse relation to the Euclidean distance between target and the sensor position. For this case, (7.1) is interpreted as follows. If a particular target located at q is guarded by agent i, the probability that it will be missed by the sensor is non-increasing with respect to the distance to the target. Otherwise if the target is not guarded by the agent i, of course it will *always* be undetected by the agent i.

7.2 Optimality Criterion

In order to frame the multi-robot $k-$redundent deployment problem as an optimization problem, we use the probability of missed detection of targets in Q by n robots as our cost function, and attempt to minimize the cost with respect to the choice of partition, assignment of guards to each disjoint region, and configuration of robots. To make *explicit* dependence on the three quantities we have just mentioned, we define $\mathcal{H}$ to be a real-valued function of x, $\mathcal{R}$, and G whose function value, i.e., cost, is given by

$$\mathcal{H}(x, \mathcal{R}, G) = \sum_{j=1}^{m} \int_{R_j} \prod_{x_i \in G(R_j)} f_i(\|q - x_i\|) \phi(q)\, dq \tag{7.2}$$

where x, $\mathcal{R}$, G is as before. The optimization problem is merely to choose right parameters $x, \mathcal{R}$ and G minimizing the cost

$$\min_{x} \min_{\mathcal{R}} \min_{G} \mathcal{H}(x, \mathcal{R}, G). \tag{7.3}$$

We will now show how to select x, $\mathcal{R}$, and G. First result states that for a fixed configuration x, the order$-k$ Voronoi partition and the associated set of generators are the necessary condition for x to be a minimizer. While it is not yet clear how to choose the an optimal configuration under the order$-k$ Voronoi tessellation that provides both necessity and sufficiency, we will propose an algorithm in the next section which will achieve local optimality in a limited sense.

Since a typical MSN consists of hundreds to thousands of robots equipped identical sensors, we assume throughout:

Assumption 7.2.1 (indistinguishability)**.** *The functions $f_i : \mathbb{R}_{\geq 0} \to [0, 1]$, $i = 1, \ldots, n$ are twice continuously differentiable identical functions, i.e., $f_1 = \cdots = f_n$, that are non-decreasing along $\mathbb{R}_{\geq 0}$,*

And to avoid pathological cases[2], we will assume throughout:

Assumption 7.2.2 (general quadratic position [17, 148])**.** *For a given set of points $x \in (\mathbb{R}^d)^n$ where $n \geq 4$:*

(a) x is finite and distinct,

(b) (the non-cosphericity) there does not exists a hyper sphere, C, such that points $x_{l1}, \ldots, x_{lk} \in x (k \geq 4)$ are on C and all points in $x \setminus \{x_{l1}, \ldots, x_{lk}\}$ are outside C,

(c) (the non-collinearity) the points in x are not on the same line.

7.2.1 The optimal partition

7.2.1.1 An ordinary Voronoi diagram

We first review the ordinary Voronoi diagram defined over the Euclidean space.

Definition 7.2.1 (A Voronoi partition [149])**.** *Let $x = \{x_1, \ldots, x_n\}$ be a set of n distinct points where $x_i \in \mathbb{R}^d$.*

$$V(x_i) = \{q \in \mathbb{R}^d \mid \|q - x_i\| \leq \|q - x_j\| \; j \in \mathcal{I} \setminus \{i\}\}$$

is the ordinary Voronoi region associated with the generator positioned at x_i. We denote by

$$\mathscr{V}(x) = \{V(x_1), \ldots, V(x_n)\}$$

the ordinary Voronoi digram generated with x.

[2] A number of examples are discussed in [17].

We now consider the generalized version of the Voronoi diagram which was proposed by Shamos and Hoey [144].

7.2.1.2 An order$-k$ Voronoi partition

Definition 7.2.2 (An order$-k$ Voronoi partition [144])**.** *Let $x = \{x_1, \ldots, x_n\}$ be a set of points where $x_l \in \mathbb{R}^d$. An order$-k$ Voronoi region associated with a point set $\mathcal{U} = \{x_{l_1}, \ldots, x_{l_k}\}$ with size k is given by*

$$V(\mathcal{U}) = \{q \in \mathbb{R}^d \mid \max_{x_j \in \mathcal{U}} \|q - x_j\| \leq \min_{x_h \in x \setminus \mathcal{U}} \|q - x_h\|\}$$

where we call $\mathcal{U}$ the generators of $V(\mathcal{U})$. The set of all such regions comprise the order$-k$ Voronoi diagram, $\mathscr{V}_k(x)$ which is given by

$$\mathscr{V}_k(x) = \{V(\mathcal{U}) \mid \mathcal{U} \subset x, |\mathcal{U}| = k\}.$$

To be used in the sequel, if we let m be the binomial coefficient indexed by n and k

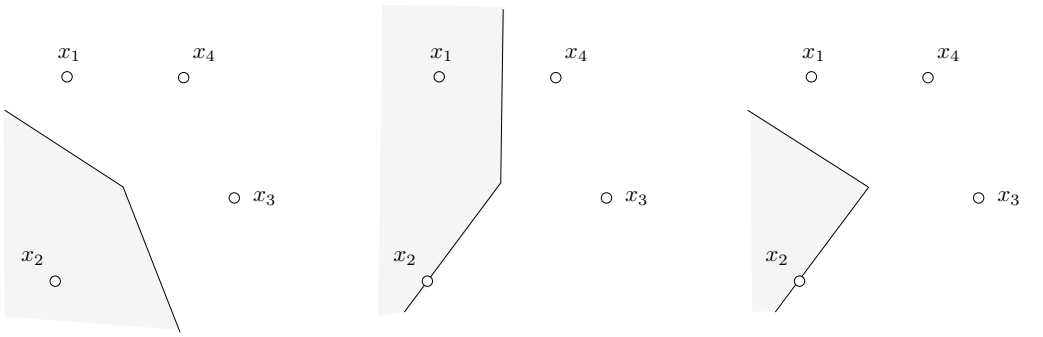
$$m := \binom{n}{k},$$

The size of each order$-k$ Voronoi partition with n generators is henceforth upper-bounded by m, i.e., $|\mathscr{V}_k(x)| \leq m$. Depending on the choice of the subset $\mathcal{U} \subset x$ some Voronoi regions associated with the point set, i.e., $V(\mathcal{U})$, can be empty. Additional details about obtaining the number of non-empty Voronoi regions of all orders is discussed by Shamos and Hoey [144]. Note that given some set of k points $\mathcal{U} \subset x$, $V(\mathcal{U})$ is the locus of points closer or equal to some point in $\mathcal{U}$ to any other point not in $\mathcal{U}$. Thus, alternatively using the halfspace we may define $V(\mathcal{U})$ by

$$V(\mathcal{U}) = \bigcap_{x_i \in \mathcal{U}, x_j \in x \setminus \mathcal{U}} H(x_i, x_j) = \bigcap_{x_i \in \mathcal{U}} \bigcap_{x_j \in x \setminus \mathcal{U}} H(x_i, x_j) \tag{7.4}$$

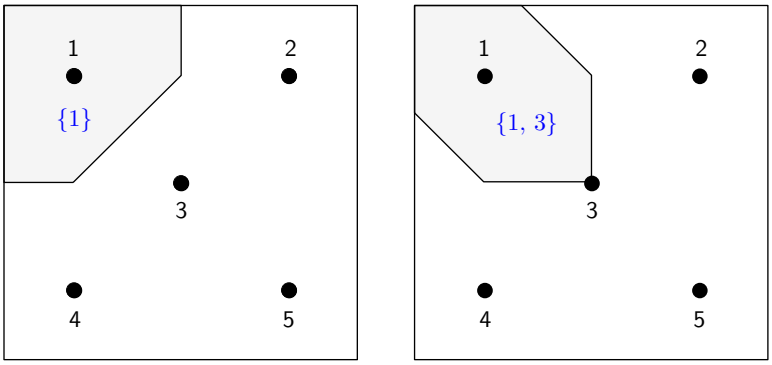
where $H(x_i, x_j)$ is the closed halfspace of $\mathbb{R}^d$ of points closer or equal to x_i than to x_j. It is shown in Fig. 7.3, an example procedure to obtain an order$-k$ Voronoi region, associated with some point set $\mathcal{U} = \{x_1, x_2\}$ using (7.4). It is reported in [145] that the worst-case complexity for obtaining the order$-k$ Voronoi diagram with n points in $\mathbb{R}^d$ is $O(k(n-k))$.

Fig. 7.4 shows examples of order$-k$ Voronoi regions with $k = 1, 2$. In Fig. 7.4(a), the shaded region



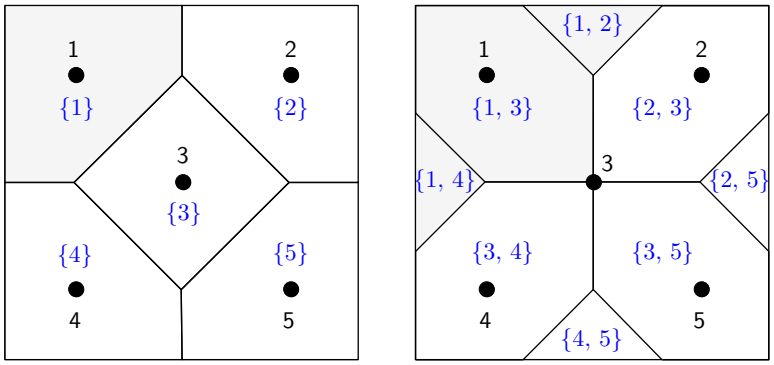
(a) $\bigcap_{x_j \in x \setminus \mathcal{U}} H(x_2, x_j)$ (b) $\bigcap_{x_j \in x \setminus \mathcal{U}} H(x_1, x_j)$ (c) $\bigcap_{x_i \in \mathcal{U}} \bigcap_{x_j \in x \setminus \mathcal{U}} H(x_i, x_j)$

Figure 7.3: A procedure to compute Voronoi region $V(\mathcal{U})$. In this example $x = \{x_1, x_2, x_3, x_4\}, \mathcal{U} = \{x_1, x_2\}$.



(a) k=1 (b) k=2

Figure 7.4: The order$-k$ Voronoi regions.



(a) k=1 (b) k=2

Figure 7.5: 1^{st} node's guarded regions under the order$-k$ Voronoi partition.

$V_1(\{x_1\})$ is associated with $\{1\}$ with $k = 1$, and in Fig. 7.4(b) the region $V_2(\{x_1, x_3\})$ is associated with $\{1, 3\}$ with $k = 2$. Fig. 7.5 shows the 1^{st} node's guarded region under the order$-k$ Voronoi tessellations of the a square workspace with $k = 1, 2$ respectively. The shaded areas show regions associated with robot 1 under order-1 Voronoi partition in Fig. 7.5(a), and under order-2 Voronoi partition in Fig. 7.5(b) respectively.

7.2.1.3 The optimal partition, and mapping

Given an order$-k$ partition, we define a special type of mapping $G_k : 2^{\mathscr{V}_k} \to 2^x$, which assigns *each order$-k$ region to its generator.*

Theorem 7.2.1. *For $k-$redundant coverage, the cost $\mathcal{H}(x, \mathcal{R}, G)$ is minimized only if $\mathcal{R} = \mathscr{V}_k$, and $G = G_k$.*

Proof. For the proof we use the definition of order$-k$ Voronoi diagram. Consider a set $\mathcal{U} \subset x$ with size k whose associated order$-k$ Voronoi region is non-empty. Then by Definition 7.2.2

$$V_k(\mathcal{U}) = \{q \in \mathbb{R}^d \mid \max_{x_j \in \mathcal{U}} \|q - x_j\| \leq \min_{x_h \in x \setminus \mathcal{U}} \|q - x_h\|\}$$

where $\mathcal{U} = G_k(V_k(\mathcal{U}))$. We claim that for each $q \in V_k(\mathcal{U})$

$$\prod_{x_j \in \mathcal{U}} \|q - x_j\| \leq \prod_{x_h \in \mathcal{T}} \|q - x_h\| \tag{7.5}$$

holds for all $\mathcal{T} \subset x$ with $|\mathcal{T}| = k$. We consider two cases. **Case 1:** $\mathcal{U} = \mathcal{T}$

The equality (7.5) holds trivially.

Case 2: $\mathcal{U} \neq \mathcal{T}$

In this case, $\mathcal{T} \setminus \mathcal{U} \neq \emptyset$ such that we may express RHS of (7.5) by

$$\prod_{x_h \in \mathcal{T} \setminus \mathcal{U}} \|q - x_h\| \prod_{x_l \in \mathcal{U} \cap \mathcal{T}} \|q - x_l\|$$

In a similar manner the left hand side of (7.5) can be expressed as

$$\prod_{x_i \in \mathcal{U} \setminus \mathcal{T}} \|q - x_j\| \prod_{x_j \in \mathcal{U} \cap \mathcal{T}} \|q - x_j\| .$$

We note that $|\mathcal{T} \setminus \mathcal{U}| = |\mathcal{U} \setminus \mathcal{T}| = |\mathcal{T} - \mathcal{T} \cap \mathcal{U}| = |\mathcal{U} - \mathcal{T} \cap \mathcal{U}|$, and for all $x_h \in \mathcal{T} \setminus \mathcal{U}$

$$\max_{x_i \in \mathcal{U}} \|q - x_i\| \leq \|q - x_h\| ,$$

which implies

$$\prod_{x_i \in \mathcal{U} \setminus \mathcal{T}} \|q - x_j\| \leq \prod_{x_h \in \mathcal{T} \setminus \mathcal{U}} \|q - x_h\| .$$

Hence, (7.5) holds as claimed.

Under Assumption 7.2.1, for each value of $q \in V_k(\mathcal{U})$

$$\prod_{x_j \in \mathcal{U}} f_j(\|q - x_j\|)\phi(q) \leq \prod_{x_h \in \mathcal{T}} f_h(\|q - x_h\|)\phi(q). \tag{7.6}$$

Without loss of generality, for general partition $\mathcal{R}$ that is not necessarily an order$-k$ Voronoi partition, we define an index set for regions from $\mathcal{R}$ which has non-empty intersection with $V_k(\mathcal{U})$ by

$$\mathcal{I}_{\mathcal{U}} := \{j \in \{1, \ldots, m\} \mid R_j \cap V_k(\mathcal{U}) \neq \emptyset\}$$

Then, we claim

$$\int_{V_k(\mathcal{U})} \prod_{x_j \in \mathcal{U}} f_j(\|q - x_j\|)\phi(q)\, dq \leq \sum_{j \in \mathcal{I}_{\mathcal{U}}} \int_{R_j \cap V_k(\mathcal{U})} \prod_{x_h \in G(R_j)} f_h(\|q - x_h\|)\phi(q)\, dq. \tag{7.7}$$

Since integration from both hand sides of (7.7) is taken over $V_k(\mathcal{U})$ with density function ϕ, and (7.6) holds for every choice of $q \in V_k(\mathcal{U})$, the inequality in (7.7) holds.

After taking summation over the workspace Q, we have

$$\sum_{\mathcal{U}\subset x,\,|\mathcal{U}|=k,\,\mathcal{U}=G_k(V_k(\mathcal{U}))}\int_{V_k(\mathcal{U})}\prod_{x_j\in\mathcal{U}} f_j(\|q-x_j\|)\phi(q)\,dq \leq \sum_{j=1}^{m}\int_{R_j}\prod_{x_h\in G(R_j)} f_h(\|q-x_h\|)\phi(q)\,dq. \quad (7.8)$$

If $\mathcal{R}\neq\mathscr{V}_k(x)$, i.e., $\mathcal{R}$ is not an order$-k$ Voronoi tessellation of Q given x, (7.8) must hold with strict inequality over some measurable set of Q. Thus, we have shown so far that $\mathcal{H}$ is minimized when every region in $\mathcal{R}$ is chosen to be exactly the order$-k$ Voronoi region associated with the its k generators. ■

7.2.2 Optimal sensor configuration

7.2.2.1 Non-convexity of the cost function (separately convex over w.r.t. each variable)

The proof of Proposition 7.2.1 depends on the following lemma.

Lemma 7.2.1. *Let g_1 and g_2 be convex functions defined over a convex set $\mathcal{C}\subset\mathbb{R}^d$. Then for each w_1, $w_2\in\mathbb{R}_{\geq 0}$ the weighted sum of two functions $w_1g_1+w_2g_2$ is also convex over $\mathcal{C}$.*

We will omit the proof of Lemma 7.2.1. The proof of Lemma 7.2.1 follows from the definition of the convex functions and is contained in [150–152].

Proposition 7.2.1. *For a $k-$redundant coverage, if Q is convex, and ϕ is bounded on Q, then for $\mathcal{R}$ and G, $\mathcal{H}(x,\mathcal{R},G)$ is separately convex with respect to each variable $x_1,\ldots,x_n$.*

Proof. Without loss of generality, for a given partition $\mathcal{R}$ and G, we may choose $i\in\mathcal{I}$. If x_j is fixed for all $j\in\mathcal{I}\setminus\{i\}$, then the function $\mathcal{H}(x,\mathcal{R},G)$ is merely infinite sums of weighted convex functions $f_i(\|q-x_i\|)$ for particular $q\in Q$ plus a constant term. Under Assumption 7.2.1, and given conditions, i.e., Q is bounded subset of $\mathbb{R}^d$ that is convex, and ϕ is bounded on Q, it follows from Lemma 7.2.1 that $\mathcal{H}(x,\mathcal{R},G)$ is convex with respect to x_i over its domain Q. We can repeat this process for every $i\in\mathcal{I}$. This completes the proof. ■

7.2.2.2 Distributed cost function

Consider a cost function associated with each individual robot. We refer to the cost solely due to ith robot's effort for target detection by *Distributed cost*. For each $i\in\mathcal{I}$, consider a map $\mathcal{M}_i: Q^n\to\mathbb{R}_{\geq 0}$ defined by

$$\mathcal{M}_i(x) := \sum_{R_j\in G^{-1}(x_i)}\int_{R_j}\prod_{x_l\in G(R_j)} f_l(\|q-x_l\|)\phi(q)\,dq \quad (7.9)$$

where the choice of $\mathcal{R}$ and G is implicit. An important relationship between total cost and sum of all n distributed costs is provided in the following proposition.

Proposition 7.2.2. *Given $k-$redundant coverage*

$$\mathcal{H}(x, \mathcal{R}, G) = \frac{1}{k} \sum_{i=1}^{n} \mathcal{M}_i(x) \tag{7.10}$$

Proof. The proof is immediate by combining two equations (7.2) and (7.9).

$$\begin{aligned} k\mathcal{H}(x, \mathcal{R}, G) &= k \sum_{j=1}^{m} \int_{R_j} \prod_{x_i \in G(R_j)} f_i(\|q - x_i\|) \phi(q)\, dq \\ &= \sum_{j=1}^{m} k \int_{R_j} \prod_{x_i \in G(R_j)} f_i(\|q - x_i\|) \phi(q)\, dq \\ &= \sum_{i=1}^{n} \sum_{R_j \in G^{-1}(x_i)} \int_{R_i} \prod_{x_l \in G(R_j)} f_l(\|q - x_l\|) \phi(q)\, dq \\ &= \sum_{i=1}^{n} \mathcal{M}_i(x) \end{aligned} \tag{7.11}$$

By the definition of $k-$redundant coverage, each disjoint region is guarded by exactly k sensors. By dividing both hand sides of (7.11) by k, (7.10) is obtained. ■

Proposition 7.2.3. *For a given $k-$redundant coverage, $\mathcal{R}$ and G, $x^* \in Q^n$ is a critical point of $\mathcal{H}(x, \mathcal{R}, G)$ if and only if $\nabla_{x_i} \mathcal{M}_i(x^*) = \mathbf{0}_{d\times 1}$ for all $i = 1, \ldots, n$.*

Proof. By direct comparison between (7.2) (7.9), we obtain

$$\nabla_{x_i} \mathcal{H}(x, \mathcal{R}, G) = \nabla_{x_i} \mathcal{M}_i(x), \;\; i = 1, \ldots, n \tag{7.12}$$

If $x^* := \{x_1^*, \ldots, x_n^*\} \in Q^n$ is the common critical point for all functions $\mathcal{M}_1, \ldots, \mathcal{M}_n$, i.e., $\nabla_{x_i} \mathcal{M}_i(x^*) \mid_{x_i = x_i^*} = \mathbf{0}_{d\times 1}$ for all $i \in \mathcal{I}$. Then using the relationship (7.12), for each $i \in \mathcal{I}$ $\nabla_{x_i} \mathcal{H}(x^*, \mathcal{R}, G) = \nabla_{x_i} \mathcal{M}_i(x^*) = \mathbf{0}_{d\times 1}$ which implies that x^* is the critical point of $\mathcal{H}(x, \cdot, \cdot)$. Only if part can be verified in a similar manner. ■

7.2.2.3 $1-$redundant coverage (the non-redundant case)

The proof of Proposition 7.2.4 depends on the following lemmas and theorems:

Lemma 7.2.2. *A real square matrix* $\mathbf{M}$ *is positive definite if and only if* $\det(\mathbf{M}) > 0$.

Lemma 7.2.3. *Let* $\mathbf{M}_1, \ldots, \mathbf{M}_n$ *be* $l \times l$ *real square matrices, and* $\mathbf{M} := \text{diag}(\mathbf{M}_1, \ldots, \mathbf{M}_n)$ *be a* $nl \times nl$ *block diagonal matrix. Then*

$$\det(\mathbf{M}) = \prod_{i=1}^{n} \det(\mathbf{M}_i).$$

Proofs for the two lemmas are found in texts on matrix analysis e.g., [153].

Theorem 7.2.2. *Consider a real-valued function* $g(y)$ *with continuous first and second derivatives defined over some convex set* D. *The function* $g(y)$ *is strictly convex if and only if the Hessian* $\mathbf{H}g(y)$ *is positive definite for all* $x \in D$.

Theorem 7.2.3. *If* $g : \mathbb{R}^d \to \mathbb{R}$ *is a twice differentiable, (strictly) convex function over some convex set* $D \subset \mathbb{R}^d$, *then any critical points of* g *in* D *is a (strict) global minimizer of* g.

Proofs for both Theorem 7.2.2 and 7.2.3 are found in non-linear programming texts, e.g., [150, 152].

Proposition 7.2.4. *For a* $1-$*redundant coverage, if* Q *is a bounded subset of* $\mathbb{R}^d$ *that is convex and* x^* *is a critical point of* $\mathcal{H}(x, \mathcal{R}, G)$ *for each choice of* $\mathcal{R}$ *and* G, *then* x^* *is a strict global minimizer.*

Proof. Recall that for the given assumptions the family of identical functions $f_1, \ldots, f_n$ are strictly convex. If $k = 1$, each region region R_i is guarded by only one agent positioned at x_i, such that the innermost product term in $\mathcal{M}_i(x)$ contains only $f_i(\|q - x_i\|)$. Thus, using Proposition 7.2.3 we have

$$\nabla_{x_j} \nabla_{x_i} \mathcal{H}(x, \mathcal{R}, G) = \nabla_{x_j} \nabla_{x_i} \mathcal{M}_i(x) = \begin{cases} \mathbf{0}_{d \times d}, & \text{if } j \neq i \\ \nabla^2_{x_i} \mathcal{M}_i, & \text{if } j = i \end{cases}$$

and the Hessian is a block matrix given by

$$\mathbf{H}\mathcal{H}(x, \mathcal{R}, G) = \begin{bmatrix} \nabla^2_{x_1} \mathcal{M}_1(x) & \mathbf{0}_{d \times d} & \cdots & \mathbf{0}_{d \times d} \\ \mathbf{0}_{d \times d} & \nabla^2_{x_2} \mathcal{M}_2(x) & \cdots & \mathbf{0}_{d \times d} \\ \vdots & & & \\ \mathbf{0}_{d \times d} & \mathbf{0}_{d \times d} & \cdots & \nabla^2_{x_n} \mathcal{M}_n(x). \end{bmatrix}$$

We claim that every $d \times d$ block matrix on the diagonal is positive definite. For all $i \in \mathcal{I}$ by Proposition 7.2.1, $\nabla^2_{x_i} \mathcal{M}_i(x) \succ 0$, and by Lemma 7.2.2 this implies $\det(\nabla^2_{x_i} \mathcal{M}_i(x)) > 0$. It follows from Lemma

7.2.3

$$\det(\mathbf{H}\mathcal{H}(x,\mathcal{R},G)) = \prod_{i=1}^{n} \det(\nabla^2_{x_i}\mathcal{M}_i(x)) > 0.$$

Again, by applying Lemma 7.2.2

$$\mathbf{H}\mathcal{H}(x,\mathcal{R},G) \succ 0,$$

holds for each $\mathcal{R}$, G. It follows from Theorem 7.2.2 that $\mathcal{H}$ is strictly convex over Q^n. Let $x^* := (x_1^*, \ldots, x_n^*)$. be the critical point of $\mathcal{H}$. Then by Theorem 7.2.3, the *strict* convexity of the function implies that x^* is the strict global minimizer for $\mathcal{H}$ over Q^n. ■

7.3 Our algorithm: a robust deployment algorithm

7.3.1 Algorithm description

We proposed a distributed iterative algorithm where at each iteration step only a set of robots that are *sufficiently independent* to each other move. The meaning of "sufficient independence" depends on whether positions for the set of robots are *decoupled* in the cost function $\mathcal{H}$. For a given configuration $x \in (\mathbb{R}^d)^n$ of n sensors, let $\mathbb{S}(x)$ be the *collection* of all sets of sensor positions that do not guard a common region:

$$\mathbb{S}(x) := \{\mathcal{S} \subset x \mid \forall x_i,\, x_j \in \mathcal{S} \text{ with } x_i \neq x_j,\, \nexists \mathcal{U} = G_k(V_k(\mathcal{U})) \neq \emptyset \text{ s.t. } x_i,\, x_j \in \mathcal{U}\}\,.$$

We note that $\mathbb{S}(x)$ is *uniquely* determined by the configuration x. In our proposed algorithm, at each time step only a group of robots whose set of positions are one of the elements of the collection $\mathbb{S}(x)$ will operate. For our algorithm, we assume that each robot is capable of, or have access to the following information.

(A) Total number of robots in the system (MRS)

(B) Global clock

(C) Self identifier

(D) Peer-to-Peer(P2P) communication with its local neighbors

Based on the capabilities, at each time step, every robot communicates with its local neighbors to decide whether to operate or not. The followings are involved in such processes.

Algorithm 3: A procedure to collectively obtain $\mathcal{S}$ via P2P communication

Require:

- Each robot knows n; the total number of robots
- Each robot knows global clock
- Each robot knows its self identifier
- Each robot can communication with its neighbors

for *each iteration* t **do**
  $\mathcal{S} \leftarrow \emptyset$, **active**$[i] \leftarrow 0$ for all $i \in \mathcal{I}$ **for** *each robot* $i \in \mathcal{I}$ **do**
    if mod$(t, n) = i$ *or* **mod**$(t, n) = 0$ *and* $i = n$ **then**
      active$[i] \leftarrow 1$
    end
  end
  for *each robot* $i \in \mathcal{I}$ **do**
    if mod$(t, n) \neq i$ **then**
      if active$[j] = 0$ *for all* x_j *such that there exists* $\mathcal{U} = G_k(V_k(\mathcal{U})) \subset x$ *with* $x_i, x_j \in \mathcal{U}$ **then**
        if *for all* j, $i < j$ **then**
          active$[i] \leftarrow 1$
        end
      end
    end
    if active$[i] = 1$ **then**
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{x_i\}$
    end
  end
  return $\mathcal{S}$
end

(a) Robot should check whether its neighbors are active

(b) Robot should check whether its neighbor and the robot itself guard a common region.

(c) Robot should check whether its neighbor has a smaller identifier compared to its own identifier.

The details of the algorithm is summarized in Algorithm 3 and 4.

Remark: In fact, (iv) the inter-agent communication is not necessary but sufficient. We shall show that as long as the information (i)-(iii) is available to every robot, our algorithm will converge. It is expected that more communications will improve convergence speed of our proposed algorithm, Nevertheless, since it is not of our main concern we will not discuss the topic further in the chapter.

7.3.2 Optimal property of the algorithm

We will derive an expression for the case when only a subset of sensor positions $\mathcal{S} \subseteq x$, where $\mathcal{S} \in \mathbb{S}(x)$, operate, such that $\mathcal{H}(x, \mathcal{R}, G)$ becomes essentially a function of $\mathcal{S}, \mathcal{R}$ and G. It is convenient to define

Algorithm 4: A robust deployment algorithm

Require: $\mathcal{S} \in \mathbb{S}(x), v_{\max}$
for *each iteration* **do**
  for *each robot* $i \in \mathcal{I}$ **do**
    if $x_i \in \mathcal{S}$ **then**
      obtain x_i^* that solves $\nabla_{x_i}\mathcal{M}(x)\,|_{x_i=x_i^*} = \mathbf{0}_{d\times 1}$
      if $\|x_i - x_i^*\| \le v_{\max}$ **then**
        $x_i \leftarrow x_i^*$
      else $x_i \leftarrow x_i + v_{\max}\frac{x_i^* - x_i}{\|x_i - x_i^*\|}$
      end
    end
  end
end

another function $\overline{\mathcal{H}}$ by

$$\overline{\mathcal{H}}(\mathcal{S},\mathcal{R},G) := \mathcal{H}(x,\mathcal{R},G)\Big|_{(x\setminus\mathcal{S}\text{ is fixed})}$$

For convenience, let $s := |\mathcal{S}|$ be the cardinality of $\mathcal{S}$.

Proposition 7.3.1. *For a $k-$redundant coverage, if Q is a compact convex subset of $\mathbb{R}^d$ and ϕ is bounded over Q, then for each $\mathcal{R}$, G, $x \in Q^n$, and $\mathcal{S} \in \mathbb{S}(x)$, the critical point $\mathcal{S}^*$ of $\overline{\mathcal{H}}(\mathcal{S},\mathcal{R},G)$ is the strict global minimizer for the function over Q^s.*

Proof. The proof is almost identical to that of Proposition 7.2.4. Similar to what we have in Proposition 7.2.3, given $x \in Q^n$ and $\mathcal{S} \in \mathbb{S}(x)$ for each x_i, $x_j \in \mathcal{S}$

$$\nabla_{x_j}\nabla_{x_i}\overline{\mathcal{H}}(\mathcal{S},\mathcal{R},G) = \nabla_{x_j}\nabla_{x_i}\mathcal{M}_i(x) = \begin{cases} \mathbf{0}_{d\times d}, & \text{if } x_j \in \mathcal{S}\setminus\{x_i\} \\ \nabla^2_{x_i}\mathcal{M}_i(x). & \text{if } x_j = x_i, \end{cases}$$

provided that $x \setminus \mathcal{S}$ is fixed. Using the relation we have obtained so far, the Hessian $\mathbf{H}\overline{\mathcal{H}}(\mathcal{S},\mathcal{R},G)$ is obtained as

$$\mathbf{H}\overline{\mathcal{H}}(\mathcal{S},x,G) = \begin{bmatrix} \nabla^2_{x_{i_1}}\mathcal{M}_{i_1}(x) & \mathbf{0}_{d\times d} & \cdots & \mathbf{0}_{d\times d} \\ \mathbf{0}_{d\times d} & \nabla^2_{x_{i_2}}\mathcal{M}_{i_2}(x) & \cdots & \mathbf{0}_{d\times d} \\ \vdots & & & \\ \mathbf{0}_{d\times d} & \mathbf{0}_{d\times d} & \cdots & \nabla^2_{x_{i_s}}\mathcal{M}_{i_s}(x) \end{bmatrix}$$

where $x_{i_j} \in \mathcal{S}$ for all $j \in \{1,\dots,s\}$, and $i_l < i_{l+1}$ for $l = 1,\dots,s-1$. Similar to Proposition 7.2.1, for the given condition $\overline{\mathcal{H}}(\mathcal{S},\mathcal{R},G)$ is seperately convex to $x_i \in \mathcal{S}$. Thus, each diagonal block matrix is

positive definite. It follows from Lemma 7.2.2, $\nabla^2_{x_{i_j}} \mathcal{M}_{i_j}(x) \succ 0$ for all $x_{i_j} \neq x^*_{i_j}$, and this implies

$$\det(\nabla^2_{x_{i_j}} \mathcal{M}_{i_j}(x)) > 0.$$

By Lemma 7.2.3, provided that $x \setminus \mathcal{S}$ is fixed,

$$\det(\mathbf{H}\overline{\mathcal{H}}(\mathcal{S}, x, G)) = \prod_{j=1}^{s} \det(\nabla^2_{x_{i_j}} \mathcal{M}_{i_j}(x)) > 0.$$

Again by Lemma 7.2.2,

$$\mathbf{H}\overline{\mathcal{H}}(\mathcal{S}, \mathcal{R}, G) \succ 0 \tag{7.13}$$

holds for each $\mathcal{R}$, G, Then by Theorem 7.2.2, (7.13) implies that $\overline{\mathcal{H}}$ is strictly convex over Q^s. If we let $\mathcal{S}^* = (x^*_{i_1}, \ldots, x^*_{i_s})$ is the critical point of $\overline{\mathcal{H}}$ over Q^s, then by Theorem 7.2.3, the strict convexity of the function implies that the critical point $\mathcal{S}^*$ is the strict global minimizer for $\overline{\mathcal{H}}$ over Q^s. ■

7.3.3 Convergence of our algorithm

For a given $x \in Q^n$ and $\mathcal{S} \in \mathbb{S}(x)$, our algorithm can be represented by a *single-valued map* $\widetilde{T}^{\mathrm{rd}} : Q^n \times Q^s \to Q^n$ comprised of n component maps $\widetilde{T}^{\mathrm{rd}} = (\widetilde{T}^{\mathrm{rd}}_1, \ldots, \widetilde{T}^{\mathrm{rd}}_n)$ where each map is defined by

$$\widetilde{T}^{\mathrm{rd}}_i(x, \mathcal{S}) = \begin{cases} x^\star_i, & \text{if } x_i \in \mathcal{S} \\ x_i, & \text{otherwise} \end{cases} \tag{7.14}$$

where

$$x^\star_i = \begin{cases} x^*_i, & \text{if } \|x_i - x^*_i\| \leq v_{\max} \\ x_i + v_{\max} \dfrac{x^*_i - x_i}{\|x_i - x^*_i\|}, & \text{if } \|x_i - x^*_i\| > v_{\max} \end{cases}$$

provided that maximum displacement per stage is set to $v_{\max}$ for all agents, and $x^*_i \in Q$ solves $\nabla_{x_i} \mathcal{M}(x) \mid_{x_i = x^*_i} = \mathbf{0}_{d \times 1}$.

Since, for a given x and $\mathbb{S}(x)$ there are *finite* number of possible ways to collectively choose $\mathcal{S}$ for each time step, it would be convenient to define our algorithm using a *set-valued mapping*. Let $T^{\mathrm{rd}} : Q^n \to 2^{Q^n}$ be a set-valued map given by

$$T^{\mathrm{rd}}(x) := \{\widetilde{T}^{\mathrm{rd}}(x, \mathcal{S}) \in Q^n \mid \mathcal{S} \in \mathbb{S}(x)\}. \tag{7.15}$$

Thus, our algorithm is expressed as a difference inclusion using the set-valued map T^{rd} by

$$x(t+1) \in T^{\mathrm{rd}}(x(t)), \quad t \in \mathbb{Z}_{\geq 0}. \tag{7.16}$$

where $x(0) \in Q^n$, $Q \subset \mathbb{R}^d$.

Note that for a given $\mathcal{S} \in \mathbb{S}(x)$, each single-valued map $\widetilde{T}^{\mathrm{rd}} : Q^n \times Q^s \to Q^n$ is continuous on its domain Q^n. Here, the collection $\mathbb{S}(x)$ is uniquely determined by the configuration x. Due to the *continuity* of each single-valued map $\widetilde{T}^{\mathrm{rd}}$, it follows by results from set-valued analysis, i.e., [154], that the set-valued map T^{rd} is *closed* on $Q^n \subset (\mathbb{R}^d)^n$. See the Appendix D for more details on this.

The proof of Theorem 7.3.1 depends on the following proposition.

Proposition 7.3.2. *If $\widetilde{T}^{\mathrm{rd}} : Q^n \to Q^n$ is continuous, then $T^{\mathrm{rd}} : Q^n \to 2^{Q^n}$ is closed.*

Proof. We first note that for a given $x \subset Q^n \subseteq (R^d)^n$, we may choose a collection $\mathbb{S}(x)$. Due to the finiteness of $\mathbb{S}(x)$, the set $T^{\mathrm{rd}}(x)$ is also *finite*. Also, we note that for every choice of $\mathcal{S}$, the map $\widetilde{T}^{\mathrm{rd}}$ is continuous on Q^n. Consider a convergent sequence of point set $(\mathcal{X}(k))_{l=0}^{\infty}$

$$\mathcal{X}(l) \to \mathcal{X} \tag{7.17}$$

where $\mathcal{X}(l) \in Q^n$, and $\mathcal{X} \in Q^n$. By the definition of continuous functions, (7.17) implies

$$\widetilde{T}^{\mathrm{rd}}(\mathcal{X}(k), \mathcal{S}) \to \widetilde{T}^{\mathrm{rd}}(\mathcal{X}, \mathcal{S}).$$

Without loss of generality, if we let

$$\mathcal{Y}(k) := \widetilde{T}^{\mathrm{rd}}(\mathcal{X}(l), \mathcal{S})$$

and

$$\mathcal{Y} := \widetilde{T}^{\mathrm{rd}}(\mathcal{X}, \mathcal{S})$$

then certainly

$$\mathcal{Y}(k) \to \mathcal{Y}.$$

For each $\mathcal{S}$, using the relation (7.15)

$$\mathcal{Y}(k) = \widetilde{T}^{\mathrm{rd}}(\mathcal{X}(k), \mathcal{S}) \in T^{\mathrm{rd}}(\mathcal{X}(k)),$$

and

$$\mathcal{Y} = \widetilde{T}^{\mathrm{rd}}(\mathcal{X}, \mathcal{S}) \in T^{\mathrm{rd}}(\mathcal{X}).$$

Since $\mathcal{Y} \in T^{\mathrm{rd}}(\mathcal{X})$ holds under every choice of $\mathcal{S} \in \mathbb{S}(x)$, it follows from Definition D.2.1 that the map T^{rd} is closed. ■

We now state the main theorem of this chapter.

Theorem 7.3.1 (Convergence of our algorithm)**.** *Consider a group of n robots initially deployed at $x(0) = \{x_1(0), \ldots, x_n(0)\}$ in $Q \subseteq \mathbb{R}^d$. Under Assumption 7.2.1, and $k-$redundant coverage, if Q is bounded subset of $\mathbb{R}^d$ that is convex, and ϕ is bounded on Q, then a sequence $(x(t))_{t\in\mathbb{Z}_{\geq 0}}$ generated by our algorithm from $x(0)$ approaches the set of critical points for $\mathcal{H}$.*

Proof. By Proposition 7.3.1, for a given $x \in Q^n$, $\mathcal{R}$, and G

$$\overline{\mathcal{H}}(\mathcal{S}^*, \mathcal{R}, G) \leq \overline{\mathcal{H}}(\mathcal{S}, \mathcal{R}, G)$$

holds for $\mathcal{S} \in \mathbb{S}(x)$ where $\mathcal{S}^*$ is a critical point of $\overline{\mathcal{H}}$ on Q^s provided that $x \setminus \mathcal{S}$ is fixed. Thus, for $\mathcal{R} = \mathscr{V}_k(x)$, and $G = G_k$

$$\overline{\mathcal{H}}(\mathcal{S}^*, \mathscr{V}_k(x), G_k) \leq \overline{\mathcal{H}}(\mathcal{S}, \mathscr{V}_k(x), G_k) \tag{7.18}$$

holds and using (7.14), (7.18) implies:

$$\begin{aligned}\mathcal{H}(\widetilde{T}^{\mathrm{rd}}(x, \mathcal{S}), \mathscr{V}_k(x), G_k) &= \overline{\mathcal{H}}((\widetilde{T}_i^{\mathrm{rd}}(x, \mathcal{S}))_{x_i\in\mathcal{S}}, \mathscr{V}_k(x), G_k) \\ &\leq \overline{\mathcal{H}}(\mathcal{S}, \mathscr{V}_k(x), G_k) = \mathcal{H}(x, \mathscr{V}_k(x), G_k)\end{aligned}$$

which holds as long as $x \setminus \mathcal{S}$ is fixed. Since the above inequality holds for every $\mathcal{S} \in \mathbb{S}(x)$, we may apply the set-valued map T^{rd} which was previously defined (7.15). Thus, for each $\mathcal{Z} \in T^{\mathrm{rd}}(x)$

$$\mathcal{H}(\mathcal{Z}, \mathscr{V}_k(x), G_k) \leq \mathcal{H}(x, \mathscr{V}_k(x), G_k).$$

Applying Theorem 7.2.1,

$$\mathcal{H}(\mathcal{Z}, \mathscr{V}_k(\mathcal{Z}), G_k) \leq \mathcal{H}(\mathcal{Z}, \mathscr{V}_k(x), G_k) \leq \mathcal{H}(x, \mathscr{V}_k(x), G_k)$$

for all $\mathcal{Z} \in T^{\mathrm{rd}}(x)$. This implies

$$\mathcal{H}(\mathcal{Z}, \mathscr{V}_k(\mathcal{Z}), G_k) \leq \mathcal{H}(x, \mathscr{V}_k(x), G_k) \tag{7.19}$$

for all $\mathcal{Z} \in T^{\mathrm{rd}}(x)$. Consider $\mathcal{L}(x) := \mathcal{H}(x, \mathscr{V}_k(x), G_k)$ as a Lyapunov function candidate for (7.16) where $\mathscr{V}_k(x)$, and G_k is uniquely determined by x. We claim that $\mathcal{L}$ is a Lyapunov function for (7.16) on Q^n. To show this, $\mathcal{L}$ must be continuous and non-decreasing along the trajectory $(x(t))_{t=0}^{\infty}$ generated by (7.16), and bounded below. First, it follows from (7.19) for each $x \subset Q^n$

$$\mathcal{L}(\mathcal{Z}) \leq \mathcal{L}(x) \tag{7.20}$$

holds for all $\mathcal{Z} \in T^{\mathrm{rd}}(x)$. Due to the continuity of $\mathcal{H}$ on Q^n, $\mathcal{L}$ is also continuous on Q^n, and because $\mathcal{H}$ is the probability measure $\mathcal{L}$ bounded below by 0. Hence, $\mathcal{L}$ is a proper Lyapunov function for (7.16).

Consider the algorithm that is generated by the set-valued map as in (7.16) Note that every set of points generated from $x(0)$ is bounded in Q^n [solutions to constrained convex program obtained at each iteration is bounded in Q^n], thus trajectory $(x(t))_{t=0}^{\infty}$ is *bounded*. Also, by Proposition 7.3.2, the set-valued map T^{rd} is *closed* on Q^n.

Let E be a set of positions for n robots defined by

$$E = \left\{ \mathcal{Y} \in Q^n \mid \exists \mathcal{Z} \in T^{\mathrm{rd}}(\mathcal{Y}) \text{ s.t. } \mathcal{H}(\mathcal{Z}, \mathscr{V}_k(\mathcal{Z}), G_k) = \mathcal{H}(\mathcal{Y}, \mathscr{V}_k(\mathcal{Y}), G_k) \right\}.$$

Under the obtained conditions so far, it follows from *LaSalle's Invariance principle* for algorithms defined via set-value maps (see the Theorem D.4.1) by taking the limit as $t \to \infty$, the sequence generated by our algorithm approaches the limit set M that is the largest weakly positively invariant set contained in E.

It remains to who that the set M is exactly the set of critical points for $\mathcal{H}$. Let C be a set of critical points for $\mathcal{H}$ given by

$$C = \{ \mathcal{X} \in Q^n \mid \nabla_x \mathcal{H}(x, \mathscr{V}_k(x), G_k) \mid_{x=\mathcal{X}} = \mathbf{0}_{nd \times 1} \}.$$

Let us show that $M = C$. Clearly, $C \subset M$. To prove the other inclusion $M \subset C$, we reason by contradiction. Consider $x = \{x_1, \ldots, x_n\} \in M \setminus C$. Then by Proposition 7.2.3, there exists $i \in \mathcal{I}$ such that $\nabla_{x_i} \mathcal{M}_i(x) \neq \mathbf{0}_{d \times 1}$. Since $\mathcal{M}_i$ is strictly convex over Q, there is a critical point $x_i^* \neq x_i$ such that

$\nabla_{x_i}\mathcal{M}_i(x)\mid_{x_i=x_i^*}=\mathbf{0}_{d\times 1}$ and

$$\mathcal{M}_i(\{x_1,\ldots,x_i^*,\ldots,x_n\})<\mathcal{M}_i(\{x_1,\ldots,x_i,\ldots,x_n\}). \tag{7.21}$$

For each $j\in\mathcal{I}\setminus\{i\}$ we consider two parts:

$$\begin{aligned}\mathcal{M}_j(x)=&\underbrace{\sum_{R_h\in G^{-1}(x_j)\setminus G^{-1}(x_i)}\int_{R_h}\prod_{x_l\in G(R_h)}f_l(\|q-x_l\|)\phi(q)\,dq}_{\mathcal{M}_{j1}(x)}\\&+\underbrace{\sum_{R_h\in G^{-1}(x_j)\cap G^{-1}(x_i)}\int_{R_h}\prod_{x_l\in G(R_h)}f_l(\|q-x_l\|)\phi(q)\,dq}_{\mathcal{M}_{j2}(x)}.\end{aligned}$$

The first part on the right hand side does not changes its value due to x_i, only the second part on the right hand side does. This is because the x_i is observed in the inner-most product term in the second part only. Note that if $G^{-1}(x_j)\cap G^{-1}(x_i)=\emptyset$ the second part vanishes. Consider the sum of all $\mathcal{M}_j(x)$ for which $j\in\mathcal{I}\setminus\{i\}$

$$\sum_{j\in\mathcal{I}\setminus\{i\}}\mathcal{M}_j(x)=\sum_{j\in\mathcal{I}\setminus\{i\}}\mathcal{M}_{j1}(x)+\sum_{j\in\mathcal{I}\setminus\{i\}}\mathcal{M}_{j2}(x)$$

We claim that the following inequality holds

$$\sum_{j\in\mathcal{I}\setminus\{i\}}\mathcal{M}_j(\{x_1,\ldots,x_i^*,\ldots,x_n\})<\sum_{j\in\mathcal{I}\setminus\{i\}}\mathcal{M}_j(\{x_1,\ldots,x_i,\ldots,x_n\}). \tag{7.22}$$

Since the value $\sum_{j\in\mathcal{I}\setminus\{i\}}\mathcal{M}_{j1}(x)$ are identical on both hand sides with respect to the change in x_i, we merely need to show

$$\sum_{j\in\mathcal{I}\setminus\{i\}}\mathcal{M}_{j2}(\{x_1,\ldots,x_i^*,\ldots,x_n\})<\sum_{j\in\mathcal{I}\setminus\{i\}}\mathcal{M}_{j2}(\{x_1,\ldots,x_i,\ldots,x_n\}).$$

We note that

$$\begin{aligned}\sum_{j\in\mathcal{I}\setminus\{i\}}\mathcal{M}_{j2}(x)&=\sum_{j\in\mathcal{I}\setminus\{i\}}\sum_{R_h\in G^{-1}(x_j)\cap G^{-1}(x_i)}\int_{R_h}\prod_{l\in G(R_h)}f_l(\|q-x_l\|)\phi(q)\,dq\\&=(k-1)\mathcal{M}_i(x).\end{aligned}$$

Hence,

$$(k-1)\mathcal{M}_i(\{x_1,\ldots,x_i^*,\ldots,x_n\}) =$$
$$\sum_{j\in\mathcal{I}\setminus\{i\}} \mathcal{M}_{j2}(\{x_1,\ldots,x_i^*,\ldots,x_n\}) < \sum_{j\in\mathcal{I}\setminus\{i\}} \mathcal{M}_{j2}(\{x_1,\ldots,x_i,\ldots,x_n\})$$
$$= (k-1)\mathcal{M}_i(\{x_1,\ldots,x_i,\ldots,x_n\})$$

and our claim is true. Combining (7.21) and (7.22) yields:

$$\sum_{j\in\mathcal{I}} \mathcal{M}_j(\{x_1,\ldots,x_i^*,\ldots,x_n\}) < \sum_{j\in\mathcal{I}} \mathcal{M}_j(\{x_1,\ldots,x_i,\ldots,x_n\}). \tag{7.23}$$

Applying Proposition 7.2.2 to the inequality (7.23) yields

$$\mathcal{L}(\{x_1,\ldots,x_i^*,\ldots,x_n\}) < \mathcal{L}(\{x_1,\ldots,x_i,\ldots,x_n\})$$

such that $\{x_1,\ldots,x_i^*,\ldots,x_n\} \notin M$ where

$$\{x_1,\ldots,x_i^*,\ldots,x_n\} \in T^{\mathrm{rd}}(\{x_1,\ldots,x_i,\ldots,x_n\}).$$

This contradicts the fact that M is weakly positively invariant. Thus $M \subset C$, and together with $C \subset M$ we have shown that $M = C$. Hence the sequence $(x(t))_{t\in\mathbb{Z}_{\geq 0}}$ generated by our algorithm defined via set-valued map T^{rd} from $x(0) \in Q^n$ approaches the set C that is the set of critical points for $\mathcal{H}$. ■

The proposition does not tell if the convergence is to an global optimal configurations. Nonetheless, the limit points are locally optimal solutions for the set of all distributed cost functions.

7.4 Simulation results

In the numerical simulation, we consider a network of 10, 20, and 30 mobile robots deployed in a planer square workspace $[0, 1]^2$ in which targets are uniformly distributed. Those robots use either Lloyd's algorithm [127] or our proposed algorithm with $k = 2$ as their control strategy. We consider *synchronous*, *discrete-time* control policy, similar to the motion cycle (look, compute, and move) that is found in the other literature on multi-robot systems (See e.g., [28]). For Lloyd's algorithm, it is assumed that sensing range for each robot is large enough to detect targets in its associated Voronoi

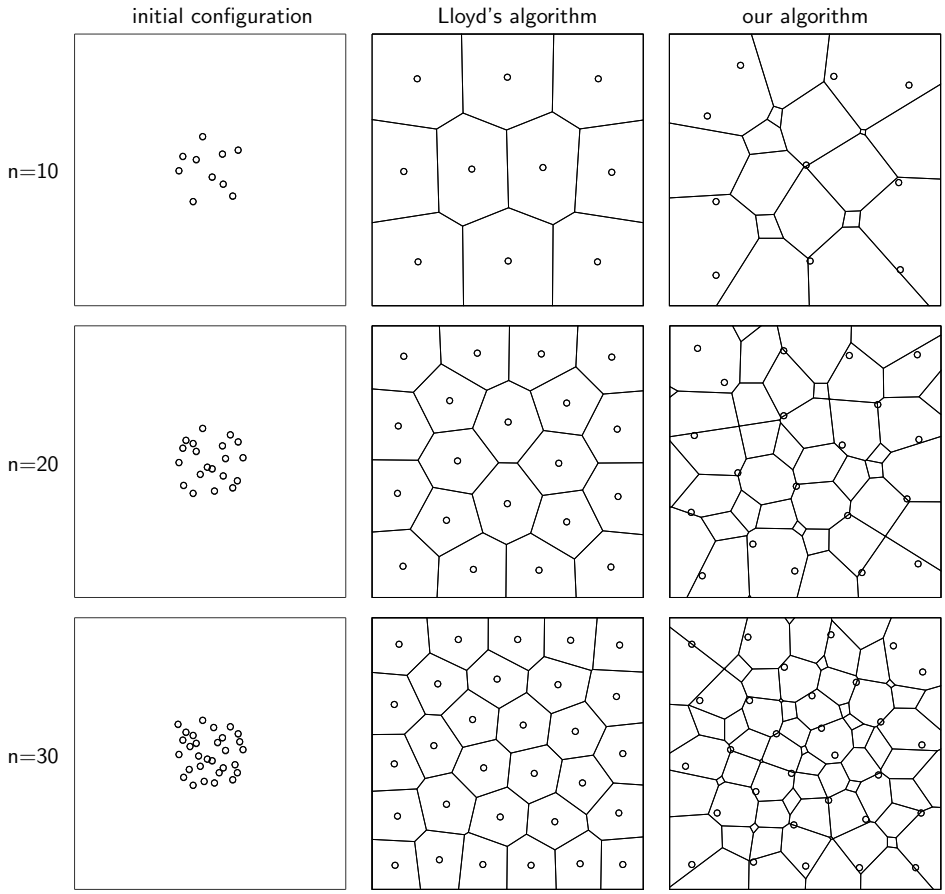


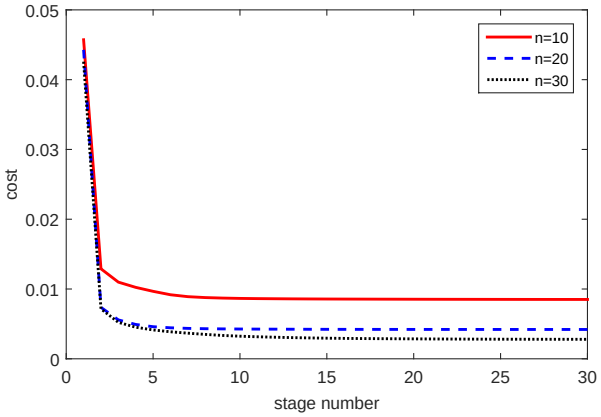
Figure 7.6: 1st column: initial configurations, 2nd column: configurations after 100 stages with Lloyd's algorithm, 3rd column: configurations after 100 stages with our algorithm, 1st row: $n = 10$, 2nd row: $n = 20$, 3rd row: $n = 30$ (filled circles: positions of robots, lines: partition of the workspace)

region. Similarly, for our algorithm, it is assumed that sensing range for each robot is large enough to detect targets in its associated order−2 Voronoi region, and each robot can communicate with its order−2 Voronoi neighbors.

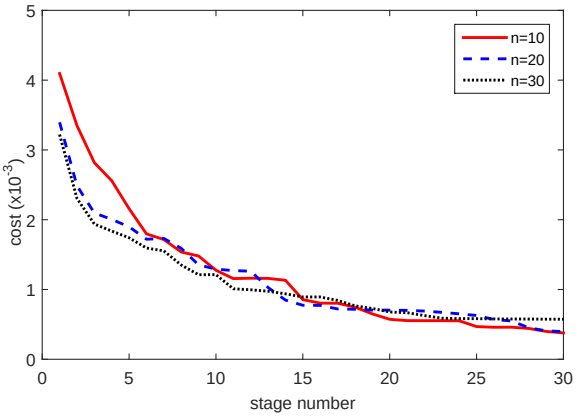
7.4.0.1 Convergence test

It is shown in the first, second and third row of Fig. 7.6 the case when $n = 10$, $n = 20$, and $n = 30$ respectively. The 1st column shows the initial configuration of robots, and the 2nd column shows configurations after 100 stages when robots execute Lloyd's algorithm, and the 3rd column shows configurations after 100 stages when robots execute our proposed algorithm.

It is shown in Fig. 7.7 the cost over 30 stages with $n = 10$, 20, and 30 when initial configuration is given as the first columns of Fig. 7.6. It is clear that as predicted by the analysis, both algorithms converge within some error bound after 100 stages; however the cost values from our algorithms are



(a)



(b)

Figure 7.7: Cost comparison between (a) Lloyd's algorithm, and (b) our algorithm over 30 stages.

Table 7.1: Types of failure

Faulty system	Servo	Sensor
Type 1		∘
Type 2	∘	
Type 3	∘	∘

much lower than that observed from Lloyd's algorithm. This is owe to the fact that, in our algorithm, multiple robots (i.e., in our case $k = 2$) enhances the overall target detection probability by jointly detecting each target.

7.4.0.2 Robustness to failures

In the second simulation example, we consider the case of 10 robots in which part of them are faulty robots. The initial configuration is shown in the 1st row of Fig. 7.6, and we will very the number of faulty robots by $n_f = 1, 2, 3, 4, 5$. We show via a few examples that our algorithm is relatively more robust to faulty robots' behavior compared to Lloyd's algorithm. For this, we consider 3 types of failures.

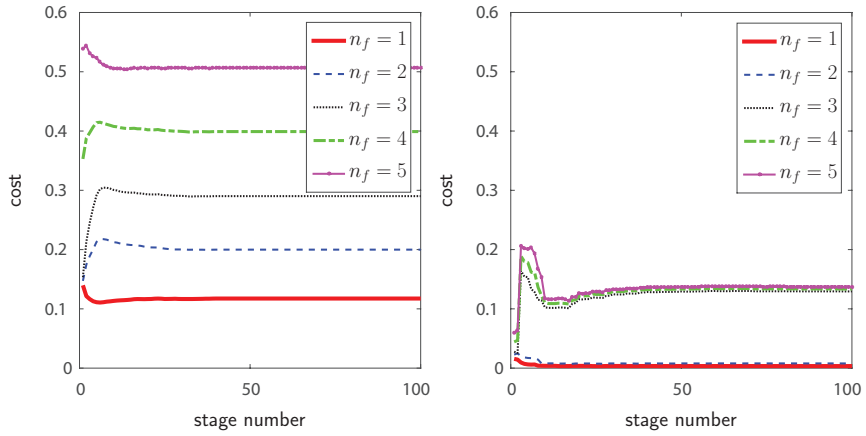
- *Type 1*: This is the case when faulty robots fail to record target detection data. This could occur when storage device is corrupt. In this case, the data collected by any faulty robot is not reliable. Thus, we may simply assume that target missed-detection probability is 1 by every faulty robot over its associated sensing region.

- *Type 2*: This is the case when servo system in each faulty robot completely fails such that every faulty robot remain stationary.

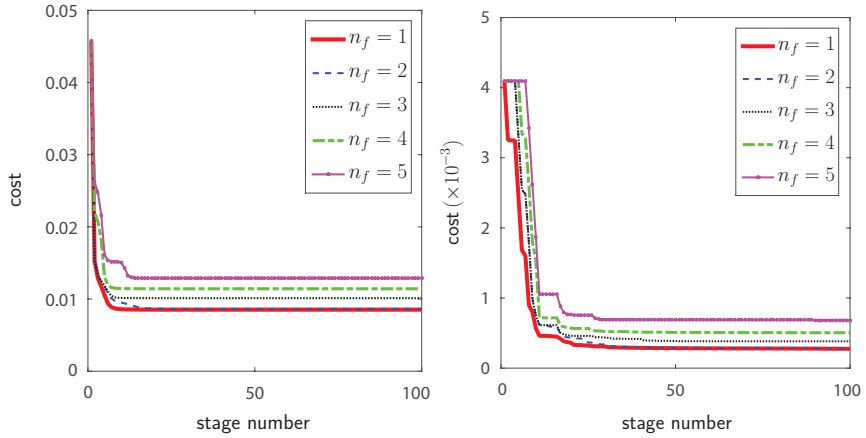
- *Type 3*: This is the case when faulty robot loses its ability to move or detect the target, i.e., complete failure.

Table. 7.1 summarizes 3 types of failures we consider in this simulation. In Fig 7.8 comparisons between cost values over 100 stages are made between Lloyd's algorithm and our proposed algorithm with different types of faulty robots.

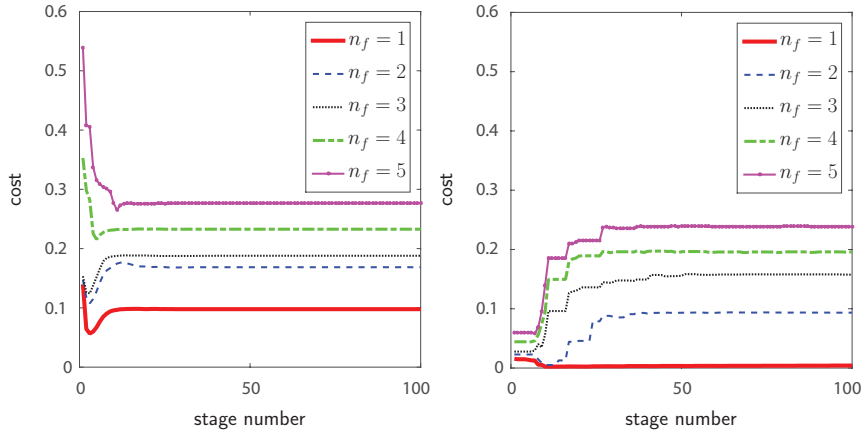
First, we consider the case when faulty robots always fail to correctly report target detection results, i.e., Type 1 failure. This could be one of the many examples when faulty robots are malicious or deceptive such that they execute the algorithm just like fault-free robots; however they are not honest about the detected target data. It is shown in Fig. 7.8(a) the overall costs obtained with two algorithms are the highest among all failure types, i.e., Type 1–3, that we consider in this example. This is roughly



(a) Type 1 failure

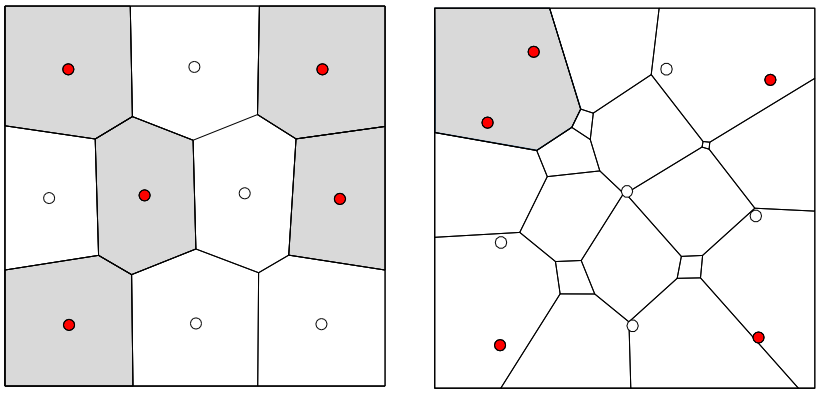


(b) Type 2 failure



(c) Type 3 failure

Figure 7.8: Cost comparison between (a) Lloyd's algorithm, and (b) our algorithm over 100 stages under different failure types.



(a) (b)

Figure 7.9: Comparison of coverage holes with (a) Lloyd's algorithm, and (b) our algorithm after 100 stage (blank circles: fault-free robots, filled circles: faulty robots).

due to the fact that for both algorithms, robots converges to their optimal configuration, and if faulty robots do not detect those targets in their associated regions that is proven to be optimal, number targets that are being missed-detected becomes the largest[3].

Next, we consider the case when faulty robots are stubborn, i.e., Type 2 failure. It is shown in Fig. 7.8(b) the overall cost change is similar to that is shown in Fig. 7.7. As seen in the figure, the higher the number of faulty robots, the higher the cost value.

Finally, we consider the case when all the faulty robots completely fail. It is shown in Fig 7.8(c) that for both algorithms the overall cost values are the largest among all three failure scenarios. In this case, our algorithm achieved slightly better detection performance than to that obtained with Lloyd's algorithm.

7.4.0.3 Coverage hole

Before we proceed to the next simulation, we briefly introduce a new term *coverage hole*. The coverage hole is an area of a workspace in $\mathbb{R}^2$ that is left undetected by any of the fault-free robots. In other words, it is the non-empty region where the missed-detection probability equals to 1. For the simulation we consider 10 robots with initial condition as the 1st row of Fig. 7.6, in which 5 robots fail with Type 1. It is shown in Fig. 7.9 the coverage hole for two algorithms, i.e., Lloyd's algorithm, and our algorithm at 30th stage. Our algorithm shows smaller are of coverage hole than that observed with Lloyd's algorithm. As seen in Fig. 7.9(b), the coverage hole is found at the top–left corner of because it is the only area which is associated with two robots that are both faulty at the same time.

[3]In fact, targets are being distributed over the continuous workspace with density ϕ such that they are not countable.

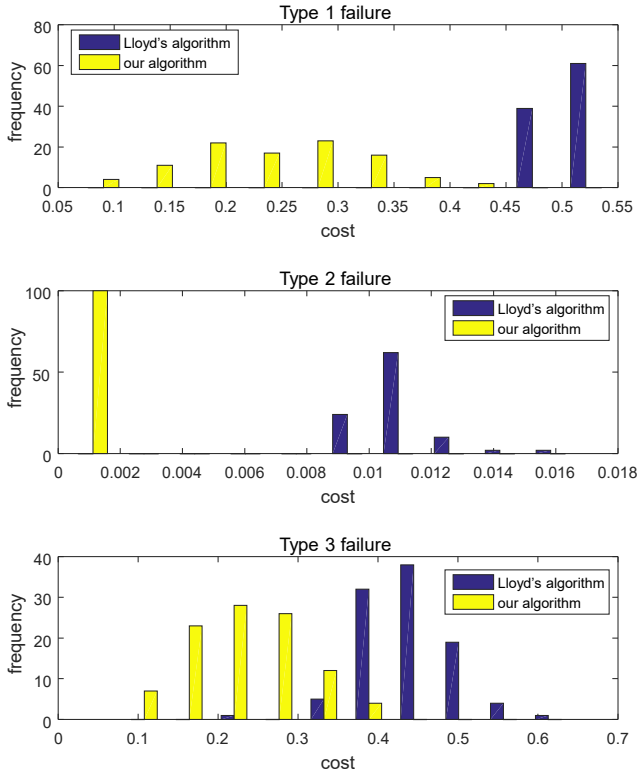


Figure 7.10: Performance of the two approaches with respect to 3 types of failure.

7.4.0.4 Statistical results

In the last simulation, we show via statistical results that our algorithm always performs well, i.e., the previous simulation results were not only the special cases. We consider 100 random deployments of 10 robots' and for each deployment 5 faulty robots are sampled without replacement. Simulations were performed for three types of failure models. Fig. 7.10 consists of histograms, each compares the frequency of the cost value at 30th stage determined by the two algorithms under different types of failures. Overall, the result shows that our algorithm is statistically more robust to all 3 types of failure modes than to Lloyd's algorithm. Especially as can been seen the second histogram from Fig. 7.10, our algorithm performs notably better with respect to Type 2 failure than to Lloyd's algorithm compared to other two histograms.

7.4.0.5 A counter example

Recall that in our algorithm for a given x, we consider $\mathbb{S}(x)$, and $\mathcal{S}$ in order to choice configuration that are sufficiently independent to each other. And at each iteration, only the sufficiently independent set of agents execute their command. Given the identical setting with $n = 10$ appears in top-left of Fig.

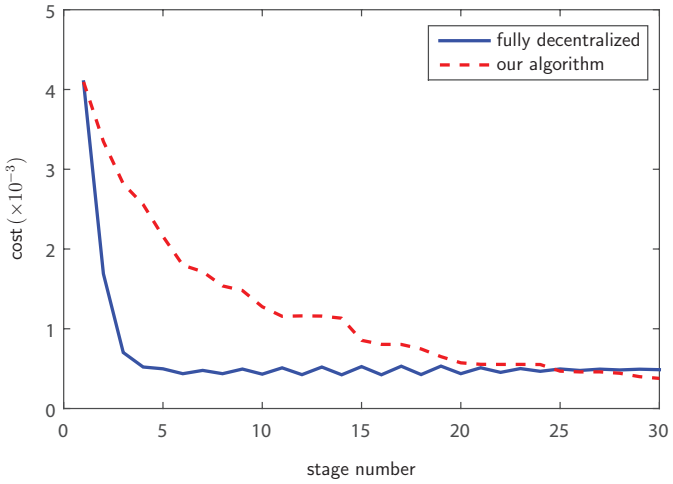


Figure 7.11: A counter example.

7.6, Fig. 7.11 compares the cost changes over 30 stages between our algorithm where only the robots with identifies in $\mathcal{S}$ move, and a fully decentralized algorithm where at each step all robots move. As can be seen, the counter example shows that cost fluctuates which implies that the fully decentralized method is not a descent algorithm. The main reason is the distributed cost function which each agent optimizes are coupled with state of the neighboring agents. Hence due to the coupled terms if all robots move to their critical points at the same time, convergence cannot be guaranteed.

7.5 Conclusion

In this chapter, we present a robust deployment algorithm based on the generalized Voronoi tessellations [144]. We showed that the order$-k$ Voronoi tessellation is the optimal partition type, and the optimal configuration is found at the critical points of distributed cost functions. Also, we proposed a descent algorithm and showed the convergence of the proposed algorithm using LaSalle's Invariance principle. A number of simulation results were present to show that our proposed algorithm is indeed robust under robot failures in statistical sense at the cost of potential increase in energy.

Our current method can be easily extended for practical implementations. A few examples are sensor model in which each node is limited by its sensing range, and where detection performance is radially non–uniform, e.g., anisotropic sensors. Also, recall that the distributed computation of $\mathcal{S}$—the index set of robots that does not guard the common regions—requires P2P communication between adjacent sensor nodes. During the P2P communication process, two nodes exchange their identifier values and

make the comparison. In fact, it is still possible to achieve convergence without the necessity of P2P communication nor violating our assumption. This can be done by commanding each node to move periodically according to the global clock and its identifier. However this method still requires each node to be aware of its identifier, total number of robots, and the global clock. We expect more practical algorithms which can achieve convergence faster than the one contained in this text.

Chapter 8

Conclusion

This thesis is about fault-tolerant control policies for multi-robot systems that are both distributed and scalable. We consider failures that include random failures experienced by individual robots, as well as the possible malicious behavior by some of the robots. Compared to other problems of designing distributed coordination strategies for multi-robot systems, *e.g.*, game theoretic or robust control approaches, our problem has its own difficulty due to faulty robots being indistinguishable members of the multi-robot system. One of the key motivations for this work lies in the following observation: In every physical multirobot system, robots are prone to fail. Despite the inherent robustness of the robotic swam or multi-robot systems to measurement errors or disturbances, partial failure of robots can bring out nontrivial performance degradation, delay, or failure of certain coordinated missions.

In this work, we presented high level fault-tolerant algorithms for multi-robot systems in the presence of faulty robots, and we attempted to either find minimally restrictive conditions under which a designed controller provides convergence guarantee, or design controllers that optimize performance in the presence of individual failures. So far, I have studied two problems, rendezvous and deployment, *i.e.*, coverage control, among many other coordination tasks for multi-robot systems. Our algorithms are constructed by techniques used in discrete geometry, and convergence of the algorithms is analyzed with nonlinear analysis. Current results show that our proposed algorithms are significantly robust to arbitrary behavior of faulty robots compared to existing algorithms found in the literature, while the trade-offs are made by slightly increased computational complexity or additional conditions imposed on the interconnection topology between robots. Although several aspects of the relevant topics for fault-tolerance for MRS have been explored in this thesis, many issues of concern and interest remain for future study. In the next section, we outline a few directions in which a new step can be taken.

8.1 Future works

In this section, we outline a few directions in which immediate progress can be made from the work in this dissertation.

8.1.1 Multi-robot robust rendezvous problem: the combinatorial approach

- *Assynchronous system*: Our approach can be easily extended to asynchronous system which has the same fault-tolerance guarantees as the current algorithm designed for synchronous system. A number of studies on this topic can be found in [34, 89].

- *Fault-tolerant rendezvous of UAVs*: Our method can be applied to the path-planning of a group of unmanned aerial vehicles. For this, additional local planner such as potential field based planner, or sampling-based planner, *e.g.*, PRM, RRT, can be used.

8.1.2 Multi-robot robust rendezvous problem: the optimization approach

- *A distributed robust decision making*: The minimax version of our program—which considers the linear combination of expected cost and the variance given the worst-case probability distributions under a few constraints on probability distribution—can be used for a *robust decision making* (see the opinion dynamics literature, *e.g.*, [50]) where each states can only observe states in its neighborhood.

- *Sample average approach*: Solutions to much more complicated versions of our problem with large neighborhood size can be obtained via utilizing other numerical optimization methods, *e.g.*, Monte Carlo sample average approach [109–111].

8.1.3 Multi-robot robust deployment problem

- *Practical sensor model*: A more realistic sensor model in which each node is limited by its maximum sensing range, and where detection performance is radially non-uniform, *e.g.*, anisotropic sensors, can be considered by simple substitution of the sensor model.

- *A general robust deployment algorithm*: One advantage of our approach is its generality. By varying k from 1 to n, one obtains the fully decentralized deployment strategy whose solution is known as CVT, or obtains the semi-distributed method whose solution is local minimums.

Currently, our numerical simulation demonstrates results with k up to 2. We postulate that we can develop effective solutions for any k.

8.1.4 Malicious nodes' strategy

Previously in our study [4, 5], we considered the performance of distributed control algorithms for networked robotic systems when one or more robots are *malicious*. In particular, we investigate the performance of the contemporary coordination algorithms, *e.g.,* circumcenter [5], Lloyd's algorithm [4], when one or more agents act maliciously to maximally disrupt the coordinated performance by group of cooperative agents. The malicious agents' algorithm are obtained via finite-horizon dynamic programming. A suite of simulation results demonstrates that the malicious nodes' actions can change the interconnection topology of the cooperative agents. Hence, we speculate that the method can be used to evaluate any fault-tolerant algorithms, not limited to those algorithms presented in Chapter 4, 5, and 7.

Appendix A

Sequential quadratic programming (SQP)

In this section, we review the sequential quadratic programming (SQP) method proposed by Wilson [155], which will in part be used to solve our series of one–step optimization problems in this chapter. The following is a typical non–linear programming problem with inequality constraints:

$$\begin{aligned} &\text{minimize} && g(x) \\ &\text{subject to} && h_i(x) \leq 0,\ i = 1, \ldots, m, \end{aligned}$$

where g, and h_i are all real–valued convex functions twice differentiable on a convex set $\mathcal{C} \subset \mathbb{R}^n$. Recall that the Lagrangian $L(x,\, \lambda)$ is defined by

$$L(x,\, \lambda) = g(x) + \sum_{j=1}^{m} \lambda_i h_i(x) \tag{A.1}$$

where the *Lagrange multiplier* λ is given as an $m \times 1$ vector $\lambda = (\lambda_1, \ldots, \lambda_m)^T$. The Hessian $H(x,\, \lambda)$ is an $n \times n$ square matrix of second-order partial derivatives of the Lagrangian $L(x,\, \lambda)$.

SQP is an iterative method. Given the current iterate x^k where $k = 0, 1, \ldots$ one can obtain the next iterate x^{k+1} by solving a sub–problem that is a quadratic programming (QP) problem. Assume that at the kth iteration, x^k and λ^k are specified. First, we obtained the second order–approximation of the Lagrangian centered at x^k by fixing λ^k.

$$L(x,\, \lambda) = L(x^k,\, \lambda^k) + \nabla L(x^k,\, \lambda^k)^T d_x + \frac{1}{2} d_x^T\, H(x^k,\, \lambda^k)\, d_x,$$

where $d_x = x - x^k$. The algorithm finds the best search directions d_x^* for each algorithm step by solving the following quadratic sub–problem that is a form of QP with linear inequality constraints with respect to the $n \times 1$ vector d_x:

$$\begin{aligned} \text{minimize} \quad & \nabla L(x^k,\, \lambda_i^k)^T d_x + \frac{1}{2} d_x^T\, H(x^k,\, \lambda^k)\, d_x \\ \text{subject to} \quad & \nabla h_i(x^k)^T d_x + h_i(x^k) \leq 0, \;\; i = 1, \ldots, m. \end{aligned}$$

Both the cost and the constraint functions are given as approximations, 2nd and 1st order respectively, of the original function at x^k in which the direction vector is d_x. One may use readily available QP solvers (e.g. CGAL, CPLEX, MATLAB) to solve the sub–problem. The solution d_x^* is used to obtain the next iterates x^{k+1}, λ^{k+1} as

$$x^{k+1} = x^k + \alpha d_x^*, \quad \lambda^{k+1} = \lambda^k + \alpha d_\lambda^*,$$

where $d_\lambda^* = \lambda_{qp}^* - \lambda^k$, λ_{qp}^* is the optimal multiplier of the above QP, and α is a step–size parameter. The algorithm terminates when solution is within some specified error bound. More details on convergence proofs are contained in various numerical optimization literature (see e.g., [156]).

Appendix B

A few results from Matrix theory

The coefficient of ergodicity was formally defined in [90, 91]. It provides a measure of *ergodicity*[1] of row-stochastic matrices. Given a row-stochastic matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$, the coefficient of ergodicity $\lambda(\mathbf{P})$ is defined by:

$$\lambda(\mathbf{P}) = 1 - \min_{i,j} \sum_{h} \min\left([\mathbf{P}]_{ih}, [\mathbf{P}]_{jh}\right). \tag{B.1}$$

The following lemma is immediate from (B.1).

Lemma B.0.1. *Consider a row-stochastic matrix* $\mathbf{P}$*. If* $\mathbf{P} \in \mathbb{R}^{n\times n}$ *has at least one column, all of whose elements are lower bounded by* $\tau > 0$*, then* $\lambda(\mathbf{P}) \leq 1 - \tau$*.*

As was stated in [92], $\lambda(\mathbf{P})$ is one of the matrix norms such that sub-multiplicity property of matrix norm holds for coefficient of ergodicity as well. In other words, given any $t \in \mathbb{Z}_+$, for row-stochastic matrices $\mathbf{S}(0), \ldots, \mathbf{S}(l)$, with $\mathbf{S}(l) \in \mathbb{R}^{n\times n}$ the following inequality holds

$$\prod_{l=0}^{t} \lambda\left(\mathbf{S}(l)\right) \leq \lambda\left(\prod_{l=0}^{t} \mathbf{S}(l)\right). \tag{B.2}$$

Given any square matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$, another quantity $\delta(\mathbf{P})$ was defined previously in [91] by

$$\delta(\mathbf{P}) = \max_{j} \max_{i_1, i_2} \left|[\mathbf{P}]_{i_1 j} - [\mathbf{P}]_{i_2 j}\right|. \tag{B.3}$$

The value $\delta(\mathbf{P})$ is called *maximum range of* $\mathbf{P}$ that is the maximum difference between any pair of elements in the same column [91], and is closely related to coefficient of ergodicity $\lambda(\mathbf{P})$ of matrix $\mathbf{P}$ such that sometimes it is also called *coefficient of ergoditicy* as well e.g., [91]. It provides the upper-bound for the difference in the rows among all columns. There is an inequality relation between the above mentioned two coefficients of ergodicities of product of stochastic matrices, and it is stated in the following lemma.

[1] Roughly speaking, the ergodicity is used to describe limiting behavior for sequence of matrix products [90].

Lemma B.0.2 (Hajnal [90] and Wolfowitz [91])**.** *For row-stochastic matrices* $\mathbf{S}(0), \ldots, \mathbf{S}(t)$*, with* $\mathbf{S}(l) \in \mathbb{R}^{n \times n}$

$$\delta\left(\prod_{l=0}^{t} \mathbf{S}(l)\right) \leq \prod_{l=0}^{t} \lambda\left(\mathbf{S}(l)\right). \tag{B.4}$$

holds for $t = 0, 1, 2, \ldots$.

Appendix C

Proofs for propositions/lemmas from Chapter 4

Proposition C.0.1. *Given a point set x in $\mathbb{R}^d$, any point $p \in \mathrm{ri}(\mathrm{conv}(x))$ can be written as non-zero convex combination of all points in $\mathrm{ver}(\mathrm{conv}(x))$ whenever $\mathrm{ri}(\mathrm{conv}(x)) \neq \emptyset$.*

Proof. Consider $m \in \mathrm{conv}(x)$ be the *barycenter*, i.e., coordinate average, of all corners of $\mathrm{conv}(x)$. For each $p \in \mathrm{ri}(\mathrm{conv}(x))$, there is $\epsilon > 0$ which makes a point $q = (1+\epsilon)p - \epsilon m$ contained in $\mathrm{ri}(\mathrm{conv}(x))$. Writing p in terms of q and m,

$$p = \frac{1}{1+\epsilon} q + \frac{\epsilon}{1+\epsilon} m$$

Since $q \in \mathrm{ri}(\mathrm{conv}(x))$, it can also be written as a convex combination of some of the corner points in $\mathrm{ver}(\mathrm{conv}(x))$. Recall that m is the arithmetic mean of all the corner points in $\mathrm{ver}(\mathrm{conv}(x))$. Thus, p can be represented by non-zero convex combination of all vertices $\mathrm{ver}(\mathrm{conv}(x))$ as claimed. ■

The proof of Lemma C.0.3 depends on the following propositions.

Proposition C.0.2. *Let $\mathbf{A}_i$, $\mathbf{B}_i \in \mathbb{R}^{n\times n}$ non-negative matrices for $i = 1, \ldots, m$. If for each i, there is $\tau_i > 0$ such that $\mathbf{A}_i \geq \mathbf{B}_i \geq \tau_i \mathbf{I}_n$ for $i = 1, \ldots, m$,*

$$\mathbf{A}_m \mathbf{A}_{m-1} \cdots \mathbf{A}_1 \geq \mathbf{B}_m \mathbf{B}_{m-1} \cdots \mathbf{B}_1 \geq \prod_{i=1}^{m} \tau_i \mathbf{I}_n.$$

Proof. $\mathbf{A}_i \geq \mathbf{B}_i$ implies $\mathbf{A}_i = \mathbf{B}_i + \mathbf{B}_i'$ where $\mathbf{B}_i' \geq 0$. Since $\mathbf{B}_i$ has positive diagonals, with $\tau_i > 0$ there is $\mathbf{B}_i'' \geq 0$ such that

$$\mathbf{B}_i = \mathbf{B}_i'' + \tau_i \mathbf{I}_n$$

We will prove by induction that, for all $m \in \mathbb{N}$

$$\mathbf{A}_m \mathbf{A}_{m-1} \cdots \mathbf{A}_1 \geq \mathbf{B}_m \mathbf{B}_{m-1} \cdots \mathbf{B}_1 \geq \prod_{i=1}^{m} \tau_i \mathbf{I}_n. \tag{C.1}$$

For the case $m = 1$ the proof is trivial.

Base case: When $m = 2$, the first term of (C.1) is

$$\begin{aligned}\mathbf{A}_2\mathbf{A}_1 &= (\mathbf{B}_2 + \mathbf{B}_2')(\mathbf{B}_1 + \mathbf{B}_1') \\ &= (\mathbf{B}_2'' + \mathbf{B}_2' + \tau_2\mathbf{I}_n)(\mathbf{B}_1'' + \mathbf{B}_1' + \tau_1\mathbf{I}_n) \\ &= \mathbf{B}_2''\mathbf{B}_1'' + \mathbf{B}_2''\mathbf{B}_1' + \tau_1\mathbf{B}_2'' + \mathbf{B}_2'\mathbf{B}_1'' \\ &+ \mathbf{B}_2'\mathbf{B}_1' + \tau_1\mathbf{B}_1' + \tau_2'\mathbf{B}_1'' + \tau_2'\mathbf{B}_1' + \tau_1\tau_2\mathbf{I}_n, \end{aligned} \tag{C.2}$$

and the second term of (C.1) is

$$\begin{aligned}\mathbf{B}_2\mathbf{B}_1 &= (\mathbf{B}_2'' + \tau_2\mathbf{I}_n)(\mathbf{B}_1'' + \tau_1\mathbf{I}_n) \\ &= \mathbf{B}_2''\mathbf{B}_1'' + \tau_1\mathbf{B}_2'' + \tau_2\mathbf{B}_1'' + \tau_1\tau_2\mathbf{I}_n. \end{aligned} \tag{C.3}$$

Since all the terms found in (C.3) can also be found in (C.2), and the rest of the terms found in (C.2) are element-wise non-negative, $\mathbf{A}_2\mathbf{A}_1 \geq \mathbf{B}_2\mathbf{B}_1$, and $\mathbf{B}_2\mathbf{B}_1 \geq \tau_2\tau_1\mathbf{I}_n$.

Induction step: Let $k \in \mathbb{Z}_{\geq 0}$ and suppose that (C.1) is true for k. If we let

$$\mathbf{C}_k := \prod_{i=1}^{k} \mathbf{A}_i,\ \mathbf{D}_k := \prod_{i=1}^{k} \mathbf{B}_i,\ \tau' := \prod_{i=1}^{k} \tau_i$$

then

$$\mathbf{C}_k \geq \mathbf{D}_k \geq \tau'\mathbf{I}_n.$$

We claim that

$$\mathbf{A}_{k+1}\mathbf{A}_k \cdots \mathbf{A}_1 \geq \mathbf{B}_{k+1}\mathbf{B}_k \cdots \mathbf{B}_1 \geq \prod_{i=1}^{k+1} \tau_i\mathbf{I}_n.$$

By substituting terms up to k by $\mathbf{C}_k$, $\mathbf{D}_k$, τ' we will show

$$\mathbf{A}_{k+1}\mathbf{C}_k \geq \mathbf{B}_{k+1}\mathbf{D}_k \geq \tau_{k+1}\tau'\mathbf{I}_n. \tag{C.4}$$

We note that

$$\mathbf{A}_{k+1} \geq \mathbf{B}_{k+1} \geq \tau_{k+1}\mathbf{I}_n,$$

and there is some $\mathbf{B}'_{k+1} \geq 0$ such that

$$\mathbf{A}_{k+1} = \mathbf{B}_{k+1} + \mathbf{B}'_{k+1}.$$

Also, we have

$$\mathbf{B}_{k+1} = \mathbf{B}''_{k+1} + \tau_{k+1}\mathbf{I}_n$$

where $\mathbf{B}''_{k+1} \geq 0$. By induction hypothesis we may write

$$\mathbf{C}_k = \mathbf{D}_k + \mathbf{D}'_k$$

with $\mathbf{D}'_k \geq 0$ and

$$\mathbf{D}_k = \mathbf{D}''_k + \tau'\mathbf{I}_n$$

where $\mathbf{D}''_k \geq 0$ such that

$$\mathbf{C}_k = \mathbf{D}''_k + \mathbf{D}'_k + \tau'\mathbf{I}_n.$$

The first term of (C.4) is

$$\begin{aligned}
\mathbf{A}_{k+1}\mathbf{C}_k &= (\mathbf{B}''_{k+1} + \mathbf{B}'_{k+1} + \tau_{k+1}\mathbf{I}_n)(\mathbf{D}''_k + \mathbf{D}'_k + \tau'\mathbf{I}_n) \\
&= \mathbf{B}''_{k+1}\mathbf{D}''_k + \mathbf{B}''_{k+1}\mathbf{D}'_k + \tau'\mathbf{B}''_{k+1} + \mathbf{B}'_{k+1}\mathbf{D}''_k \\
&+ \mathbf{B}'_{k+1}\mathbf{D}'_k + \tau'\mathbf{B}'_{k+1} + \tau_{k+1}\mathbf{B}''_1 + \tau'_{k+1}\mathbf{B}'_1 \\
&+ \tau_{k+1}\tau'\mathbf{I}_n, \qquad \text{(C.5)}
\end{aligned}$$

and the second term of (C.4) is

$$\begin{aligned}
\mathbf{B}_{k+1}\mathbf{D}_k &= (\mathbf{B}''_{k+1} + \tau_{k+1}\mathbf{I}_n)(\mathbf{D}''_k + \tau'\mathbf{I}_n) \\
&= \mathbf{B}''_{k+1}\mathbf{D}''_k + \tau'\mathbf{B}''_{k+1} + \tau_{k+1}\mathbf{D}''_k + \tau_{k+1}\tau'\mathbf{I}_n \\
&\geq \tau_{k+1}\tau'\mathbf{I}_n = \prod_{i=1}^{k+1}\tau_i\mathbf{I}_n. \qquad \text{(C.6)}
\end{aligned}$$

Since all the terms from (C.6) can also be found in (C.5), and the rest of the terms found in (C.5) are element-wise non-negative, (C.4) holds true, and the proof of the induction step is complete.

Conclusion By the principle of induction (C.1) is true for all $m \in \mathbb{N}$. ■

Lemma C.0.3. *Consider a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes where $\mathcal{A}$ is the adjacency matrix for $\mathcal{G}$. If ith node is globally reachable, then*

$$\mathbf{1}_{n\times 1} \leq [(\mathbf{I}_n + \mathcal{A})^l]_i \tag{C.7}$$

whenever $l \geq n-1$.

Proof. We will prove (C.7) by showing that

$$1 \leq [(\mathbf{I}_n + \mathcal{A})^l]_{ji}$$

holds for all $j \in \mathcal{V}$ whenever $l \geq n-1$.

1. Consider $j = i$. By Proposition C.0.2,

$$\mathbf{I}_n = \mathbf{I}_n^m \leq (\mathbf{I}_n + \mathcal{A})^m, \;\; m = 1, 2, \ldots,$$

and this implies that $1 \leq [(\mathbf{I}_n + \mathcal{A})^m]_{ii}$ for all $i \in \mathcal{V}$, and $m \in \mathbb{N}$.

2. Consider $j \in \mathcal{V} \setminus \{i\}$. Since the graph $\mathcal{G}$ has globally reachable node i, by the definition of the global reachability, if i is globally reachable for every $j \in \mathcal{V} \setminus \{i\}$, there is a directed path from node j to i. We note that the maximum geodesic distance, i.e., length of the path, between any two nodes is at most $n-1$. Thus, we may express the directed path between any two nodes (j, i) where $j \neq i$ with length $1 \leq k \leq n-1$ as a sequence $j, l_1, l_2, \ldots, l_{k-1}, i$ where $l_0 = j$. Thus, $(j, l_1) \in \mathcal{E}$, $(l_1, l_2) \in \mathcal{E}, \ldots (l_{k-2}, l_{k-1}) \in \mathcal{E}$, $(l_{k-1}, i) \in \mathcal{E}$, and this implies that $[\mathcal{A}]_{jl_1} = 1$, $[\mathcal{A}]_{l_{k-1}i} = 1$, and $[\mathcal{A}]_{l_m, l_{m+1}} = 1$ for all $m = 1, \ldots, k-2$. Hence, for each j there exists $k \leq n-1$ such that the product $[\mathcal{A}]_{jl_1}[\mathcal{A}]_{l_1l_2} \cdots [\mathcal{A}]_{l_{k-2}l_{k-1}}[\mathcal{A}]_{l_{k-1}i} = 1$, and this implies

$$1 \leq [\mathcal{A}^k]_{ji}. \tag{C.8}$$

By Proposition C.0.2,

$$\mathcal{A}^m \leq (\mathbf{I}_n + \mathcal{A})^m$$

for all $m \in \mathbb{N}$ such that for any pair $i, j \in \mathcal{V}$

$$[\mathcal{A}^m]_{ji} \leq [(\mathbf{I}_n + \mathcal{A})^m]_{ji}. \tag{C.9}$$

Combining (C.8) and (C.9), we have

$$1 \leq [(\mathbf{I}_n + \mathcal{A})^k]_{ji},$$

and again by Proposition C.0.2

$$(\mathbf{I}_n + \mathcal{A})^k \leq (\mathbf{I}_n + \mathcal{A})^m$$

holds for all $m \geq k$. Thus,

$$1 \leq [(\mathbf{I}_n + \mathcal{A})^k]_{ji} \leq [(\mathbf{I}_n + \mathcal{A})^m]_{ji}$$

for all $m \geq k$. Note that the uniform upper-bound for k is $n-1$. Hence, as long as $m \geq n-1$

$$1 \leq [(\mathbf{I}_n + \mathcal{A})^m]_{ji}$$

holds for all $j \in \mathcal{V} \setminus \{i\}$.

Combining two results for the cases $j = i$ and $j \in \mathcal{V} \setminus \{i\}$, whenever $l \geq n-1$,

$$\mathbf{1}_{n\times 1} \leq \left[(\mathbf{I}_n + \mathcal{A})^l\right]_i$$

which completes the proof. ■

Appendix D

Invariance principle for algorithms defined via set-valued maps

The appendix contains materials from Zangwill [46], Luenberger [151] and LaSalle [49].

D.1 Algorithm as a set-valued map

It is beneficial to define an *iterative* algorithm using *set-valued* map, i.e., *point-to-set* map. First, we show explicit difference between *single-valued* map, i.e., *point-to-point* map and point-to-set map. Consider a bounded subset Q of $\mathbb{R}^d$, and a point-to-point map $T : Q \to Q$. For a given point $x(0) \in Q$, an algorithm defined by the map T generates a sequence

$$x(0), x(1), x(2), \ldots$$

by

$$x(l+1) = T(x(l)), \quad l = 0, 1, 2, \ldots$$

If the map T is continuous on Q, then by the definition of continuity for every point $y \in Q$, $y(l) \to y$ as $l \to \infty$ implies $T(y(l)) \to T(y)$.

In a similar manner if T is a point-to-set map given by $T : Q \to 2^Q$ then we may define a more general version of an algorithm that is generated by the set-valued map by

$$x(l+1) \in T(x(l)), \quad l = 0, 1, 2, \ldots \tag{D.1}$$

The *motion*, i.e., *trajectory*, generated by T from $x(0)$ refers to a sequence of states:

$$x(0), x(1), \cdots, x(l), \cdots$$

We shall introduce in a moment the concept for set-valued map similar to that of continuity for

point-to-point map called closedness.

D.2 Closedness of set-valued map

Definition D.2.1 (Zangwill [46]). *Let Q be a bounded subset of $\mathbb{R}^d$, and $T : Q \to 2^Q$ be a set-valued map. The map T is closed at some $x \in Q$ if two there are sequences $(x(l))_{l=0}^{\infty}$, $(y(l))_{l=0}^{\infty}$ satisfying*

$$x(l) \to x,$$

$$y(l) \to y$$

where $y(l) \in T(x(l))$ for all $l \in \mathbb{Z}_{\geq 0}$ implies

$$y \in T(x).$$

We say that the map T is closed, if T is closed at all $x \in Q$.

We shall assume from this point on that T is a set-valued map, and is *closed*.

D.3 Limit set and invariant set

Definition D.3.1 (Birkhoff [157]). *A point y is a limit point of $(x(l))_{l=0}^{\infty}$ if there is a sequence of integers l_i; such that $(x(l_i)) \to y$ and $l_i \to \infty$ as $i \to \infty$. The limit set $\Omega(x(0))$ of $(x(l))_{l=0}^{\infty}$ is the set of all limit points of $(x(l))_{l=0}^{\infty}$.*

According to the definition, every limit set is closed. It is quite often a closed set is defined as a set which contains all its limit points [158].

Definition D.3.2. *A set $S \subset Q$ is said to be weakly positively invariant relative to $T : Q \to 2^Q$ if for each $x \in S$, there is $y \in S$ such that $y \in T(x)$.*

Theorem D.3.1. *Every limit set $\Omega(x(0))$ for (D.1) is closed and weakly positively invariant.*

Proof. Suppose $y \in \Omega(x(0))$. There is a sequence of integers l_i such that $l_i \to \infty$, and $(x(l_i)) \to y$ as $i \to \infty$. Since T is closed, then according to Definition D.2.1, if $x(l_i+1) \to y'$ where $x(l_i+1) \in T(x(l_i))$ for all $l \in \mathbb{Z}_{\geq 0}$, then $y' \in T(y)$. Since $y' \in \Omega(x(0))$ as well, we can conclude that for each $z \in \Omega(x(0))$,

there is $z' \in \Omega(x(0))$ such that $z' \in T(z)$. According to Definition D.3.2, $\Omega(x(0))$ is weakly positively invariant. ■

The proof of the following theorem follows closely to that given in [49].

Theorem D.3.2. *If trajectory $(x(l))$ by T is bounded for $l \in \mathbb{Z}_{\geq 0}$, then $\Omega(x(0))$ is non-empty, compact, weakly positively invariant, and is the smallest closet set that $x((l))$ approaches as $n \to \infty$.*

Proof. The boundedness of the motion $(x(l))$ by T clearly implies that $\Omega(x(0))$ is non-empty and bounded. Thus by Theorem D.3.1, $\Omega(x(0))$ is compact, i.e., closed and bounded, and weakly positively invariant. We shall show that the entire sequence approaches the limit set, i.e., $x(l) \to \Omega(x(0))$, if the motion $(x(l))$ is bounded. Since $\rho(x(l), \Omega(x(0)))$ is bounded, if the motion $(x(l))$ does not approach $\Omega(x(0))$, there is a sequence (l_i) such that as $i \to \infty$, $(x(l_i))$ converges but does not approach $\Omega(x(0))$. This is clearly a contradiction since the limit of $(x(l_i))$ must be in $\Omega(x(0))$ [by the definition of the limit set]. Hence $(x(l)) \to \Omega(x(0))$. For each closed set E that $(x(l)) \to E$, $\Omega(x(0)) \subset E$ such that $\Omega(x(0))$ is the *smallest* closed set that $(x(l))$ approaches as $l \to \infty$. ■

D.4 An extension of Lyapunov direct method

We shall show that, suitably defined, Lyapunov functions give information about the location of limit sets. This is done exploiting the invariance property of limit sets, and for this reason the idea behind what we are about to do is called the *Invariance Principle*.

Let $V : \mathbb{R}^d \to \mathbb{R}$. Relative to the set-valued map T, and if $(x(l))_{l=0}^{\infty}$ is solution of (D.1), we define $\dot{V}(x(l)) := V(x(l+1)) - V(x(l))$ the *difference* of V at l along T, and $\dot{V}(x(l)) \leq 0$ for all $l \in \mathbb{Z}_{\geq 0}$ means that V is *non-increasing* along solutions.

Definition D.4.1. *Let Q be any set in $\mathbb{R}^d$. We say that V is a Lyapunov function of (D.1) on Q^n if (i) V is continuous, and (ii) V is non-increasing along solutions.*

For V a Lyapunov function of (D.1) on $Q \subset \mathbb{R}^d$, we define

$$E = \{x \in \mathrm{cl}(Q) \mid \exists y \in T(x) \text{ such that } V(y) = V(x)\}$$

We use M to denote the largest weakly positively invariant set in E, and $V^{-1}(c) = \{x \in (\mathbb{R}^d)^n \mid V(x) = c\}$.

Theorem D.4.1 (Invariance principle for algorithms defined via point-to-set maps [46, 49, 151])**.** *If V is a Lyapunov function of the system (D.1) on $Q \subset \mathbb{R}^d$, and $(x(l))$ is a solution of (D.1) bounded in G for all l, then there is a number $c \in \mathbb{R}$ such that $x(l) \to M \cap V^{-1}(c)$ as $l \to \infty$.*

Proof. Our assumption implies that $V(x(l))$ is non-increasing with l and is bounded from below. Hence there is a number c such that $V(x(l)) \to c$ as $l \to \infty$. Consider a limit point $y \in \Omega(x(0))$. Then there is a sequence (l_i) such that $l_i \to \infty$ and $x(l_i) \to y$. Since V is continuous, $V(x(l_i)) \to V(y) = c$ such that $\Omega(x(0)) \subset V^{-1}(c)$. Since $\Omega(x(0))$ is weakly positively invariant, there is $z \in T(y)$ such that $V(z) = V(y) = c$. Hence, $\Omega(x(0)) \in M$, and $\Omega(x(0)) \in E$. By Theorem D.3.2, $x(l) \to \Omega(x(0))$, and this implies that $x(l) \to M \cap V^{-1}(c)$. ■

References

[1] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, "Distributed memoryless point convergence algorithm for mobile robots with limited visibility," *Robotics and Automation, IEEE Transactions on*, vol. 15, no. 5, pp. 818–828, 1999.

[2] W. Mulzer and D. Werner, "Approximating tverberg points in linear time for any fixed dimension," *Discrete & Computational Geometry*, vol. 50, no. 2, pp. 520–535, 2013.

[3] M. Bernardine Dias, M. Zinck, R. Zlot, and A. Stentz, "Robust multirobot coordination in dynamic environments," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4. IEEE, 2004, pp. 3435–3442.

[4] H. Park and S. Hutchinson, "Worst-case performance of rendezvous networks in the presence of adversarial nodes," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 5579–5585.

[5] H. Park and S. Hutchinson, "Worst-case performance of a mobile sensor network under individual sensor failure," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 895–900.

[6] R. Isaacs, "Differential games I," 1954.

[7] T. Basar, G. J. Olsder, G. Clsder, T. Basar, T. Baser, and G. J. Olsder, *Dynamic noncooperative game theory*. SIAM, 1995, vol. 200.

[8] J. Nash, "Non-cooperative games," *Annals of mathematics*, pp. 286–295, 1951.

[9] H. Ando, I. Suzuki, and M. Yamashita, "Formation and agreement problems for synchronous mobile robots with limited visibility," in *Intelligent Control, 1995., Proceedings of the 1995 IEEE International Symposium on*. IEEE, 1995, pp. 453–460.

[10] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Physical Review Letters*, vol. 75, no. 6, p. 1226, 1995.

[11] H. Park and S. Hutchinson, "A distributed robust convergence algorithm for multi-robot systems in the presence of faulty robots," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 2980–2985.

[12] H. Park and S. Hutchinson, "An efficient algorithm for fault-tolerant rendezvous of multi-robot systems with controllable sensing range," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, To appear.

[13] H. Park and S. Hutchinson, "A distributed optimal strategy for rendezvous of multi-robots with random node failures," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 1155–1160.

[14] H. Park and S. Hutchinson, "Robust optimal deployment in mobile sensor networks with peer-to-peer communication," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2144–2149.

[15] Y. Liu, K. M. Passino, and M. M. Polycarpou, "Stability analysis of m-dimensional asynchronous swarms with a fixed communication topology," *Automatic Control, IEEE Transactions on*, vol. 48, no. 1, pp. 76–95, 2003.

[16] J. Cortés, S. Martínez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *Automatic Control, IEEE Transactions on*, vol. 51, no. 8, p. 12891298, 2006.

[17] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley & Sons, 2009, vol. 501.

[18] H. Attiya and J. Welch, *Distributed computing: fundamentals, simulations, and advanced topics*. John Wiley & Sons, 2004, vol. 19.

[19] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[20] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.

[21] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, "Gathering of asynchronous robots with limited visibility," *Theoretical Computer Science*, vol. 337, no. 1, pp. 147–168, 2005.

[22] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed autonomous robotic systems 5*. Springer, 2002, pp. 299–308.

[23] J. Lin, A. Morse, and B. Anderson, "The multi-agent rendezvous problem," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 2. IEEE, 2003, pp. 1508–1513.

[24] J. Lin, A. S. Morse, and B. D. Anderson, "The multi-agent rendezvous problem. part 1: The synchronous case," *SIAM Journal on Control and Optimization*, vol. 46, no. 6, pp. 2096–2119, 2007.

[25] T. Izumi, S. Souissi, Y. Katayama, N. Inuzuka, X. Défago, K. Wada, and M. Yamashita, "The gathering problem for two oblivious robots with unreliable compasses," *SIAM Journal on Computing*, vol. 41, no. 1, pp. 26–46, 2012.

[26] I. Suzuki and M. Yamashita, "Distributed anonymous mobile robots: Formation of geometric patterns," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1347–1363, 1999.

[27] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro, "Solving the robots gathering problem," in *Automata, Languages and Programming*. Springer, 2003, pp. 1181–1196.

[28] X. Défago, M. Gradinariu, S. Messika, and P. Raipin-Parvédy, *Fault-tolerant and self-stabilizing mobile robots gathering*. Springer, 2006.

[29] G. Prencipe, "Impossibility of gathering by a set of autonomous mobile robots," *Theoretical Computer Science*, vol. 384, no. 2, pp. 222–231, 2007.

[30] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *Automatic Control, IEEE Transactions on*, vol. 50, no. 2, pp. 169–182, 2005.

[31] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *Automatic Control, IEEE Transactions on*, vol. 48, no. 6, pp. 988–1001, 2003.

[32] J. N. Tsitsiklis, D. P. Bertsekas, M. Athans et al., "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE transactions on automatic control*, vol. 31, no. 9, pp. 803–812, 1986.

[33] P. Flocchini, G. Prencipe, and N. Santoro, "Distributed computing by oblivious mobile robots," *Synthesis Lectures on Distributed Computing Theory*, vol. 3, no. 2, pp. 1–185, 2012.

[34] J. Lin, A. S. Morse, and B. D. Anderson, "The multi-agent rendezvous problem. part 2: The asynchronous case," *SIAM Journal on Control and Optimization*, vol. 46, no. 6, pp. 2120–2147, 2007.

[35] J. Cortés, S. Martínez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *Automatic Control, IEEE Transactions on*, vol. 51, no. 8, pp. 1289–1298, 2006.

[36] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, "Gathering of asynchronous oblivious robots with limited visibility," in *STACS 2001*. Springer, 2001, pp. 247–258.

[37] R. Olfati-Saber and R. Murray, "Consensus protocols for networks of dynamic agents," 2003.

[38] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, p. 15201533, 2004.

[39] R. Sepulchre, D. Paley, and N. Leonard, "Collective motion and oscillator synchronization," in *Cooperative control*. Springer, 2005, pp. 189–205.

[40] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *Control Systems, IEEE*, vol. 27, no. 2, pp. 71–82, 2007.

[41] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1465–1476, 2004.

[42] Z. Lin, B. Francis, and M. Maggiore, "Necessary and sufficient graphical conditions for formation control of unicycles," *Automatic Control, IEEE Transactions on*, vol. 50, no. 1, pp. 121–127, 2005.

[43] L. Moreau, "Leaderless coordination via bidirectional and unidirectional time-dependent communication," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 3. IEEE, 2003, pp. 3070–3075.

[44] D. J. Elzinga and D. W. Hearn, "The minimum covering sphere problem," *Management Science*, vol. 19, no. 1, pp. 96–104, 1972.

[45] L. Blumenthal and G. Wahlin, "On the spherical surface of smallest radius enclosing a bounded subset of n dimensional Euclidean space," *Bulletin of the American Mathematical Society*, vol. 47, no. 10, pp. 771–777, 1941.

[46] W. I. Zangwill, *Nonlinear programming: a unified approach*. Prentice-Hall Englewood Cliffs, NJ, 1969.

[47] J. Cortés, S. Martínez, and F. Bullo, "Spatially-distributed coverage optimization and control with limited-range interactions," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 11, no. 04, pp. 691–719, 2005.

[48] B. Degener, B. Kempkes, T. Langner, F. Meyer auf der Heide, P. Pietrzyk, and R. Wattenhofer, "A tight runtime bound for synchronous gathering of autonomous robots with limited visibility," in *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*. ACM, 2011, pp. 139–148.

[49] J. P. LaSalle, J. Hale, and K. Meyer, "The stability and control of discrete processes," *Applied mathematical sciences*, vol. 62, 1986.

[50] J. Lorenz, "Repeated averaging and bounded confidence-modeling, analysis and simulation of continuous opinion dynamics," Ph.D. dissertation, University of Bremen, Germany, available at http://nbn-resolving. de/urn: nbn: de: gbv: 46-diss000106688, 2007.

[51] J. Lorenz and D. A. Lorenz, "On conditions for convergence to consensus," *Automatic Control, IEEE Transactions on*, vol. 55, no. 7, pp. 1651–1656, 2010.

[52] N. H. Vaidya, "Iterative Byzantine vector consensus in incomplete graphs," in *Distributed Computing and Networking*. Springer, 2014, pp. 14–28.

[53] H. Mendes and M. Herlihy, "Multidimensional approximate agreement in Byzantine asynchronous systems," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013, pp. 391–400.

[54] A. Aviziens, "Fault-tolerant systems," *IEEE Transactions on Computers*, vol. 25, no. 12, pp. 1304–1312, 1976.

[55] N. H. Vaidya, L. Tseng, and G. Liang, "Iterative approximate Byzantine consensus in arbitrary directed graphs," in *Proceedings of the 2012 ACM symposium on Principles of distributed computing*. ACM, 2012, pp. 365–374.

[56] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *Selected Areas in Communications, IEEE Journal on*, vol. 31, no. 4, pp. 766–781, 2013.

[57] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl, "Reaching approximate agreement in the presence of faults," *Journal of the ACM (JACM)*, vol. 33, no. 3, p. 499516, 1986.

[58] A. Doudou and A. Schiper, "Muteness detectors for consensus with byzantine processes," in *in Proceedings of the 17th ACM Symposium on Principle of Distributed Computing,(Puerto*. Citeseer, 1997.

[59] M. Zhu and S. Martínez, "On distributed constrained formation control in operator–vehicle adversarial networks," *Automatica*, vol. 49, no. 12, pp. 3571–3582, 2013.

[60] M. Zhu and S. Martinez, "On resilient consensus against replay attacks in operator-vehicle networks," in *American Control Conference (ACC), 2012*. IEEE, 2012, pp. 3553–3558.

[61] H. Zhang and S. Sundaram, "A simple median-based resilient consensus algorithm," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*. IEEE, 2012, pp. 1734–1741.

[62] A. Khanafer, B. Touri, and T. Basar, "Robust distributed averaging on networks with adversarial intervention," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*. IEEE, 2013, pp. 7131–7136.

[63] S. Sundaram, S. Revzen, and G. Pappas, "A control-theoretic approach to disseminating values and overcoming malicious links in wireless networks," *Automatica*, vol. 48, no. 11, pp. 2894–2901, 2012.

[64] A. Gusrialdi, Z. Qu, and M. A. Simaan, "Robust design of cooperative systems against attacks," in *American Control Conference (ACC), 2014*. IEEE, 2014, pp. 1456–1462.

[65] G. Parlangeli, "A fault compensation strategy for consensus networks subject to transient and intermittent faults," in *Control & Automation (MED), 2013 21st Mediterranean Conference on*. IEEE, 2013, pp. 46–53.

[66] F. Pasqualetti, A. Bicchi, and F. Bullo, "Consensus computation in unreliable networks: A system theoretic approach," *Automatic Control, IEEE Transactions on*, vol. 57, no. 1, pp. 90–104, 2012.

[67] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation via linear iterative strategies in the presence of malicious agents," *Automatic Control, IEEE Transactions on*, vol. 56, no. 7, pp. 1495–1508, 2011.

[68] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation via linear iterations in the presence of malicious agents-part II: Overcoming malicious behavior," in *American Control Conference, 2008*. IEEE, 2008, pp. 1356–1361.

[69] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation via linear iterations in the presence of malicious agents-part I: Attacking the network," in *American Control Conference, 2008*. IEEE, 2008, pp. 1350–1355.

[70] W. Zeng, M.-Y. Chow, and P. Ning, "Secure distributed control in unreliable D-NCS," in *Industrial Electronics (ISIE), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 1858–1863.

[71] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Finite-time average consensus in a Byzantine environment using set-valued observers."

[72] N. Agmon and D. Peleg, "Fault-tolerant gathering algorithms for autonomous mobile robots," *SIAM Journal on Computing*, vol. 36, no. 1, pp. 56–82, 2006.

[73] J. Clement, X. Défago, M. G. Potop-Butucaru, S. Messika, P. Raipin-Parvédy et al., "Fault and Byzantine tolerant self-stabilizing mobile robots gathering," 2012.

[74] Z. Bouzid, S. Das, and S. Tixeuil, "Gathering of mobile robots tolerating multiple crash faults," in *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*. IEEE, 2013, pp. 337–346.

[75] Z. Bouzid, M. G. Potop-Butucaru, and S. Tixeuil, "Optimal Byzantine-resilient convergence in uni-dimensional robot networks," *Theoretical Computer Science*, vol. 411, no. 34, pp. 3154–3168, 2010.

[76] Z. Bouzid, M. G. Potop-Butucaru, and S. Tixeuil, "Byzantine-resilient convergence in oblivious robot networks," in *Distributed Computing and Networking*. Springer, 2009, pp. 275–280.

[77] Z. Bouzid, M. G. Potop-Butucaru, and S. Tixeuil, "Optimal Byzantine resilient convergence in asynchronous robots networks," in *Stabilization, Safety, and Security of Distributed Systems*. Springer, 2009, pp. 165–179.

[78] C. Auger, Z. Bouzid, P. Courtieu, S. Tixeuil, and X. Urbain, "Certified impossibility results for Byzantine-tolerant mobile robots," in *Stabilization, Safety, and Security of Distributed Systems*. Springer, 2013, pp. 178–190.

[79] M. Franceschelli, A. Giua, and A. Pisano, "Finite-time consensus on the median value by discontinuous control," in *American Control Conference (ACC), 2014*. IEEE, 2014, pp. 946–951.

[80] H. J. LeBlanc, H. Zhang, S. Sundaram, and X. Koutsoukos, "Consensus of multi-agent networks in the presence of adversaries using only local information," in *Proceedings of the 1st international conference on High Confidence Networked Systems*. ACM, 2012, pp. 1–10.

[81] G. Bracha, "An asynchronous [(n-1)/3]-resilient consensus protocol," in *Proceedings of the third annual ACM symposium on Principles of distributed computing.* ACM, 1984, pp. 154–162.

[82] H. Zhang, E. Fata, and S. Sundaram, "A notion of robustness in complex networks," *Control of Network Systems, IEEE Transactions on*, vol. 2, no. 3, pp. 310–320, 2015.

[83] A. Weber and C. J. Friedrich, "Alfred weber's theory of the location of industries," 1929.

[84] H. Tverberg, "A generalization of radon's theorem," *J. London Math. Soc*, vol. 41, no. 1, pp. 123–128, 1966.

[85] J. R. Reay et al., "An extension of radon's theorem," *Illinois Journal of Mathematics*, vol. 12, no. 2, pp. 184–189, 1968.

[86] J.-P. Roudneff, "Partitions of points into intersecting tetrahedra," *Discrete Mathematics*, vol. 81, no. 1, pp. 81–86, 1990.

[87] J.-P. Roudneff, "New cases of reays conjecture on partitions of points into simplices with k-dimensional intersection," *European Journal of Combinatorics*, vol. 30, no. 8, pp. 1919–1943, 2009.

[88] B. J. Birch, "On 3 n points in a plane," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 55, no. 04. Cambridge Univ Press, 1959, pp. 289–293.

[89] M. Cao, A. S. Morse, and B. D. Anderson, "Agreeing asynchronously," *Automatic Control, IEEE Transactions on*, vol. 53, no. 8, pp. 1826–1838, 2008.

[90] J. Hajnal, "Weak ergodicity in non-homogeneous markov chains," in *Proc. Cambridge Philos. Soc*, vol. 54, no. 2. Cambridge Univ Press, 1958, pp. 233–246.

[91] J. Wolfowitz, "Products of indecomposable, aperiodic, stochastic matrices," *Proceedings of the American Mathematical Society*, vol. 14, no. 5, pp. 733–737, 1963.

[92] E. Seneta, "Coefficients of ergodicity: structure and applications," *Advances in applied probability*, pp. 576–590, 1979.

[93] S. Chatterjee and E. Seneta, "Towards consensus: some convergence theorems on repeated averaging," *Journal of Applied Probability*, pp. 89–97, 1977.

[94] J. Hajnal and M. Bartlett, "The ergodic properties of non-homogeneous finite markov chains," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 52, no. 01. Cambridge Univ Press, 1956, pp. 67–77.

[95] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications.* ACM, 2002, pp. 32–41.

[96] M. Cardei, J. Wu, M. Lu, and M. O. Pervaiz, "Maximum network lifetime in wireless sensor networks with adjustable sensing ranges," in *Wireless And Mobile Computing 2005.(WiMob'2005), IEEE International Conference on*, vol. 3. IEEE, 2005, pp. 438–445.

[97] C. W. Wu, "Algebraic connectivity of directed graphs," *Linear and Multilinear Algebra*, vol. 53, no. 3, pp. 203–223, 2005.

[98] D. Pickem, L. Wang, P. Glotfelter, Y. Diaz-Mercado, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "Safe, remote-access swarm robotics research on the robotarium," *arXiv preprint arXiv:1604.00640*, 2016.

[99] D. Pickem, M. Lee, and M. Egerstedt, "The gritsbot in its natural habitat-a multi-robot testbed," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4062–4067.

[100] J. Czyzowicz, L. Gasieniec, and A. Pelc, "Gathering few fat mobile robots in the plane," *Theoretical Computer Science*, vol. 410, no. 6, pp. 481–499, 2009.

[101] H. Markowitz, "Portfolio selection*," *The journal of finance*, vol. 7, no. 1, pp. 77–91, 1952.

[102] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, p. 215233, 2007.

[103] J. M. Mulvey, R. J. Vanderbei, and S. A. Zenios, "Robust optimization of large-scale systems," *Operations research*, vol. 43, no. 2, pp. 264–281, 1995.

[104] A. Ben-Tal and A. Nemirovski, "Robust optimization–methodology and applications," *Mathematical Programming*, vol. 92, no. 3, pp. 453–480, 2002.

[105] H.-G. Beyer and B. Sendhoff, "Robust optimization–a comprehensive survey," *Computer methods in applied mechanics and engineering*, vol. 196, no. 33, pp. 3190–3218, 2007.

[106] W. Chen, "Quality utility–a compromise programming approach to robust design," Ph.D. dissertation, University of Illinois, 1998.

[107] C. Daskalakis, I. Diakonikolas, and R. A. Servedio, "Learning poisson binomial distributions," in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. ACM, 2012, pp. 709–728.

[108] A. Shapiro and S. Ahmed, "On a class of minimax stochastic programs," *SIAM Journal on Optimization*, vol. 14, no. 4, pp. 1237–1249, 2004.

[109] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello, "The sample average approximation method for stochastic discrete optimization," *SIAM Journal on Optimization*, vol. 12, no. 2, pp. 479–502, 2002.

[110] S. Ahmed, A. Shapiro, and E. Shapiro, "The sample average approximation method for stochastic programs with integer recourse," *Submitted for publication*, pp. 1–24, 2002.

[111] A. Shapiro and A. Kleywegt, "Minimax analysis of stochastic problems," *Optimization Methods and Software*, vol. 17, no. 3, pp. 523–542, 2002.

[112] Z. Drezner and H. W. Hamacher, *Facility location: applications and theory*. Springer, 2004.

[113] Z. Drezner, *Facility location: a survey of applications and methods*. Springer Verlag, 1995.

[114] A. Kwok and S. Martinez, "Energy-balancing cooperative strategies for sensor deployment," in *Decision and Control, 2007 46th IEEE Conference on*. IEEE, 2007, pp. 6136–6141.

[115] M. Schwager, D. Rus, and J.-J. Slotine, "Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 371–383, 2011.

[116] M. A. Batalin and G. S. Sukhatme, "Spreading out: A local approach to multi-robot coverage," in *Distributed autonomous robotic systems 5*. Springer, 2002, pp. 373–382.

[117] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems*, vol. 26, no. 2-4, pp. 181–196, 2004.

[118] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *Robotics and Automation, IEEE Transactions on*, vol. 20, no. 2, p. 243255, 2004.

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

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

[121] I. Rekleitis, V. Lee-Shue, A. P. New, and H. Choset, "Limited communication, multi-robot team based coverage," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4. IEEE, 2004, pp. 3462–3468.

[122] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1-4, pp. 113–126, 2001.

[123] T. Bretl and S. Hutchinson, "Robust coverage by a mobile robot of a planar workspace," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4582–4587.

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

[125] K. Laventall and J. Cortés, "Coverage control by multi-robot networks with limited-range anisotropic sensory," *International Journal of Control*, vol. 82, no. 6, pp. 1113–1121, 2009.

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

[127] S. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[128] H. M. Choset, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[129] W. Li and C. G. Cassandras, "Distributed cooperative coverage control of sensor networks," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*. IEEE, 2005, pp. 2542–2547.

[130] S. Hutchinson and T. Bretl, "Robust optimal deployment of mobile sensor networks," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, p. 671676.

[131] F. Aurenhammer, "Voronoi diagramsa survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.

[132] I. Gowda, D. Kirkpatrick, D. Lee, and A. Naamad, "Dynamic voronoi diagrams," *Information Theory, IEEE Transactions on*, vol. 29, no. 5, pp. 724–731, 1983.

[133] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, no. 1-4, pp. 153–174, 1987.

[134] Q. Du, M. Emelianenko, and L. Ju, "Convergence of the lloyd algorithm for computing centroidal voronoi tessellations," *SIAM journal on numerical analysis*, vol. 44, no. 1, pp. 102–119, 2006.

[135] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. ACM, 2002, pp. 88–97.

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

[137] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[138] D. S. Hochbaum and A. Pathria, "Analysis of the greedy approach in problems of maximum k-coverage," *Naval Research Logistics (NRL)*, vol. 45, no. 6, pp. 615–627, 1998.

[139] S. Kumar, T. H. Lai, and J. Balogh, "On k-coverage in a mostly sleeping sensor network," in *Proceedings of the 10th annual international conference on Mobile computing and networking*. ACM, 2004, pp. 144–158.

[140] Z. Zhou, S. Das, and H. Gupta, "Connected k-coverage problem in sensor networks," in *Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on*. IEEE, 2004, pp. 373–378.

[141] M. Hefeeda and M. Bagheri, "Randomized k-coverage algorithms for dense sensor networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE, 2007, pp. 2376–2380.

[142] H. M. Ammari and S. K. Das, "Centralized and clustered k-coverage protocols for wireless sensor networks," *Computers, IEEE Transactions on*, vol. 61, no. 1, pp. 118–133, 2012.

[143] K. Vu and R. Zheng, "Robust coverage under uncertainty in wireless sensor networks," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 2015–2023.

[144] M. I. Shamos and D. Hoey, "Closest-point problems," in *Foundations of Computer Science, 1975., 16th Annual Symposium on*. IEEE, 1975, pp. 151–162.

[145] D.-T. Lee, "On k-nearest neighbor voronoi diagrams in the plane," *IEEE Trans. Computers*, vol. 31, no. 6, pp. 478–487, 1982.

[146] A. Pierson, L. C. Figueiredo, L. C. Pimenta, and M. Schwager, "Adapting to performance variations in multi-robot coverage," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 415–420.

[147] A. Pierson and M. Schwager, "Adaptive inter-robot trust for robust multi-robot sensor coverage," in *International Symposium on Robotics Research*, 2013.

[148] J. Moller, *Lectures on random Voronoi tessellations*. Springer Science & Business Media, 2012, vol. 87.

[149] G. Voronoï, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs," 1908.

[150] D. P. Bertsekas, "Nonlinear programming," 1999.

[151] D. G. Luenberger, "Linear and nonlinear programming. 1984," *New York*.

[152] A. L. Peressini, F. E. Sullivan, and J. J. Uhl, *The mathematics of nonlinear programming*. Springer-Verlag New York, 1988.

[153] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[154] J.-P. Aubin and H. Frankowska, *Set-valued analysis*. Springer Science & Business Media, 2009.

[155] R. B. Wilson, "A simplicial algorithm for concave programming," Ph.D. dissertation, Graduate School of Business Administration, George F. Baker Foundation, Harvard University, 1963.

[156] J. Nocedal and S. Wright, "Numerical optimization, series in operations research and financial engineering," *Springer, New York*, 2006.

[157] G. D. Birknoff, "Dynamical systems," in *American Mathematical Society Colloquium Publications*, 1960.

[158] A. N. Kolmogorov and S. V. Fomin, *Introductory real analysis*. Courier Corporation, 2012.