UNIVERSITY OF CALIFORNIA
Santa Barbara

Robotic Surveillance and Deployment Strategies

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

Doctor of Philosophy

in

Mechanical Engineering

by

Rushabh Patel

Committee in Charge:

Professor Francesco Bullo, Chair

Professor João Hespanha

Professor Jeff Moehlis

Professor Bassam Bamieh

Professor Yasamin Mostofi

June 2015

The Dissertation of
Rushabh Patel is approved:

Professor João Hespanha

Professor Jeff Moehlis

Professor Bassam Bamieh

Professor Yasamin Mostofi

Professor Francesco Bullo, Committee Chairperson

April 2015

Robotic Surveillance and Deployment Strategies

by

Rushabh Patel

To my sister Tulsi.

Acknowledgements

First and foremost, I would like to thank my advisor, Francesco Bullo. He has undoubtably played a critical role in my development as a researcher, and my accomplishments would not be possible without him. I would especially like to thank him for his support and understanding through the various ups and downs that life can bring; it is through that support which my successes were possible. I know the lessons learned with him, both academic and personal, will have an everlasting impact on all my future endeavors.

I would also like to give a special thanks to Paolo Frasca, who was an incredible asset in the early stages of my research. He was very patient with me and acted as an exceptional sounding board to get my bearings during my first projects. In addition, his attention to detail helped to make our works that much better.

All of the works that appear in this thesis are the result of various rewarding collaborations with other talented scientist. My thanks go to Joey Durham, Andrea Carron, Pushkarini Agharkar and Ruggero Carli for all their help. An additional thanks goes to Pushkarini for her initial discover and discussions of the Kemeny constant (hitting time), a topic which has proven to be fruitful and a major focus of this thesis.

During my stay at UCSB, I have had several opportunities that have allowed me to branch outside my research. My thanks goes to Wendy Ibsen for running

the School for Scientific Thought as well as various other outreach programs which have allowed me to share my love of science with the community. Thank you to Jeff Peters, who played a critical role in teaching our first high-school robotics course as well as in the publication of the text which was a direct result of that course.

I would also like to thank all the present and past members of the Bullo lab. The many stimulating discussions within the lab, and social gatherings outside of it, have made my experience at UCSB a truly great one.

Lastly, I would like to thank my friends and family whose supportive words and understanding have made this achievement possible.

Curriculum Vitæ

Rushabh Patel

Education

2015 **Ph.D. Mechanical Engineering**
University of California, Santa Barbara,
Santa Barbara, CA, USA

2007 **M.S., B.S. Aerospace Engineering**
California Polytechnic State University,
San Luis Obispo, CA, USA

Research Experience

2010 – 2015 **Graduate Student Researcher**, University of California, Santa Barbara.
Two problems in multi-agent control have been investigated. Specifically, problems related to (1) coordinated deployment and partitioning in robotic networks to provide optimal coverage, and (2) unpredictable surveillance and intruder detection strategies in robotic networks.

2006 – 2007 **Master's Thesis**, California Polytechnic State University, San Luis Obispo.
Investigated the design of adaptive controllers for a four reaction wheel spacecraft simulator.

Teaching Experience

Spring 2014 **Teaching Assistant**
Course: ME 179P, *Introduction to Robotics: Planning and Kinematics*
Mechanical Engineering, University of California, Santa Barbara

Spring 2013	**Teaching Assistant** Course: ME 155A, *Control System Design* Mechanical Engineering, University of California, Santa Barbara
Fall/ Winter 2013	**Instructor** Course: *Teaching Robotics* School for Scientific Thought, University of California, Santa Barbara
Winter 2013	**Teaching Assistant** Course: ME/ECE 236, *Nonlinear Control Systems* Mechanical Engineering, University of California, Santa Barbara
Winter 2011	**Teaching Assistant** Course: ME 6, *Basic Electrical and Electronic Circuits* Mechanical Engineering, University of California, Santa Barbara
Fall 2010	**Teaching Assistant** Course: ME 140A, *Numerical Analysis in Engineering* Mechanical Engineering, University of California, Santa Barbara

Mentoring

Summer 2014	Student: Dillon Azzam Program: Research Mentorship Program University of California, Santa Barbara
Summer 2014	Student: Nicolae Christoffersen Program: Research Mentorship Program University of California, Santa Barbara
Summer 2013	Student: Ariana Del Toro Program: RISE Summer Internship University of California, Santa Barbara
Summer 2009	Student: Christopher Walsvick Program: Northrop Grumman High School Summer Internship Northrop Grumman Aerospace Systems

Professional Service

Technical Reviewer

Journal	IEEE Transaction on Automatic Control IEEE Transaction on Control of Network Systems Automatica International Journal of Control
Conference	American Control Conference IEEE Conference on Decision and Control IEEE Multi-conference on Systems and Control AIAA Guidance, Navigation and Control Conference

Publications

Texts and Thesis

1. J. R. Peters and R. Patel. Thinking Robotics: Teaching Robots to Make Decisions. www.teachengineering.org, February 2015. Note: Submitted.
2. R. Patel. Adaptive output feedback control as applied to the rigid body equations of motion. Master's thesis, California Polytechnic State University, San Luis Obispo, December 2007.

Journal Articles

1. R. Patel, A. Carron, and F. Bullo. The Hitting Time of Multiple Random Walks with Application to Robotic Surveillance. SIAM Journal on Matrix Analysis and Applications, March 2015. Note: Submitted.
2. R. Patel, P. Agharkar, and F. Bullo. Robotic Surveillance and Markov Chains with Minimal First Passage Time. IEEE Transactions on Automatic Control, May 2014. Note: To appear.
3. R. Patel, P. Frasca, and F. Bullo. Centroidal Area-Constrained Partitioning for Robotic Networks. ASME Journal of Dynamic Systems, Measurement, and Control, 136(3):031024, 2014.

4. R. Patel, P. Frasca, J. W. Durham, R. Carli, and F. Bullo. Dynamic Partitioning and Coverage Control with Asynchronous One-To-Base-Station Communication. IEEE Transactions on Control of Network Systems, January 2014. Note: To appear.

Conference Articles

1. P. Agharkar, R. Patel, and F. Bullo. Robotic surveillance and Markov chains with minimal first passage time. In IEEE Conf. on Decision and Control, Los Angeles, CA, USA, pages 6603-6608, December 2014.
2. R. Patel, P. Frasca, and F. Bullo. Centroidal area-constrained partitioning for robotic networks. In ASME Dynamic Systems and Control Conference, Stanford, CA, USA, October 2013.
3. C. Mau-Song, J.C. Donovan, R.C. Patel, and J.K. McCarthy. VIIRS near-field response, ghosting, and static electrical crosstalk maps methodology. In Calibration Conference, Honolulu, HI, August 2008.
4. R. Patel and E. Mehiel. Adaptive output feedback control as applied to the rigid body equations of motion. In AIAA Guidance Navigation and Control Conference and Exhibit, Logan, UT, August 2008.

Abstract

Robotic Surveillance and Deployment Strategies

Rushabh Patel

Autonomous mobile systems are becoming more common place, and have the opportunity to revolutionize many modern application areas. They include, but are not limited to, tasks such as search and rescue operations, ad-hoc mobile wireless networks and warehouse management; each application having its own complexities and challenging problems that need addressing. In this thesis, we explore and characterize two application areas in particular.

First, we explore the problem of autonomous stochastic surveillance. In particular, we study random walks on a finite graph that are described by a Markov chain. We present strategies that minimize the first hitting time of the Markov chain, and look at both the single agent and multi-agent cases. In the single agent case, we provide a formulation and convex optimization scheme for the hitting time on graphs with travel distances. In addition, we provide detailed simulation results showing the effectiveness of our strategy versus other well-known Markov chain design strategies. In the multi-agent case, we provide the first characterization of the hitting time for multiple random walkers, which we denote the *group hitting time*. We also provide a closed form solution for calculating the hitting

time between specified nodes for both the single and multiple random walker cases. Our results allow for the multiple random walks to be different and, moreover, for the random walks to operate on different subgraphs. Finally, we use sequential quadratic programming to find the transition matrices that generate minimal *group hitting time*.

Second, we consider the problem of optimal coverage with a group of mobile agents. For a planar environment with an associated density function, this problem is equivalent to dividing the environment into optimal subregions such that each agent is responsible for the coverage of its own region. We study this problem for the discrete time and space case and the continuous time and space case. For the discrete time and space case, we present algorithms that provide optimal coverage control in a non-convex environment when each robot has only asynchronous and sporadic communication with a base station. We introduce the notion of coverings, a generalization of partitions, to do this. For the continuous time and space case, we present a continuous-time distributed policy which allows a team of agents to achieve a convex area-constrained partition in a convex workspace. This work is related to the classic Lloyd algorithm, and makes use of generalized Voronoi diagrams. For both cases we provide detailed simulation results and discuss practical implementation issues.

Professor Francesco Bullo
Dissertation Committee Chair

Contents

List of Figures

List of Tables

Chapter 1

Introduction

You must construct additional pylons.

- StarCraft (1998)

The applications of multi-agent systems to accomplish complex tasks in a complex environment are vast. Environmental monitoring [78], search and rescue operations [58] and ad-hoc mobile wireless networks [38] are just a few areas where multi-agent systems can have significant impacts. A common problem which arises in many robotic network applications is that of dividing workload amongst agents, so tasks can be accomplished quickly and efficiently. The way workload is distributed amongst a multi-agent platform poses unique challenges for different application areas. For example, in ocean surveillance the cost of travel is large, so instead of each agent traveling around the entire region of interest, it is more effective for each agent to survey small subsets of the larger region. Placement of agents within such regions can also be of importance. In the case of surveillance or warehouse management, it is desirable to be at the center of your region so tasks

are more easily serviced. Simultaneous placement of agents and specification of those agents' regions is also of general interest, however, this can be complicated as the two problems are often coupled. In this thesis we study two closely related robotic applications; robotic surveillance and robotic partitioning and coverage control.

Robotic surveillance The design of surveillance algorithms for quickest detection of intruders and anomalies appear frequently. Specific examples include the monitoring of oil spills [20], the detection of forest fires [48], the tracking of border changes [81], and the periodic patrolling of an environment [30, 68]. Other applications in single and multi-agent systems include minimizing emergency vehicle response times [7] as well as servicing tasks in robotic warehouse management [86]. In areas of research outside of robotics, applications include, but are not limited to, determining how quickly epidemics spread [85], how information propagates in a social network [5] and how quickly information packets get transferred in a wireless node network [75]. In this paper we propose stochastic surveillance strategies based on Markov chains. More specifically, we look at the analysis, generalization and minimization of the *hitting time* of a random walk governed by a Markov chain for both single and multiple random walkers. This problem is not only of interest in the context of robotic surveillance, but is also of general mathematical

interest in the study of Markov chains and random walks; similar to the fastest mixing Markov chain, the first hitting time is a metric by which to gauge the performance of a random walk.

Coverage control In applications such as environmental monitoring or warehouse logistics a team of robots is asked to perform tasks over a large space. The distributed *environment partitioning problem* consists of designing control and communication laws for individual robots such that the team divides a space into regions in order to optimize the quality of service provided. *Coverage control* additionally optimizes the positioning of robots inside of a region. Coverage control and territory partitioning have applications in many fields. In cyber-physical systems, applications include automated environmental monitoring [33], fetching and delivery [86], and other vehicle routing scenarios [35]. In this thesis we consider two variations of the partitioning problem. In the first problem we provide a method for partitioning a discretized environment using an asynchronous one-to-base station communication law. In the second problem we look a the partitioning of continuous environment in which each agent only has communication with its neighbor. In both cases we consider different multi-center cost functions and determine optimal partitions in relation to those cost functions.

1.1 Literature review

1.1.1 Robotic surveillance and the hitting time of a random walk

In this paper we study strategies to surveil an environment, to provide a desired coverage frequency, and to detect an intruder in minimum time. The surveillance problem has appeared in the literature in various manifestations. The authors of [80] look at minimizing time of detection of noisy anomalies via persistent surveillance strategies, and in [55] wireless sensor networks are utilized for intruder detection in previously unknown environments. In [4], the authors explore strategies for surveillance using a multi-agent ground vehicle system which must maintain connectivity between agents. A non-cooperative game framework is utilized in [19] to determine an optimal strategy for intruder detection, and in [69] a similar framework is used to analyze intruder detection for ad-hoc mobile networks. In our setup we model the environment as a graph and design random walks on this graph imposing restrictions on the stationary distribution of the walk. In other works with a similar setup, Markov chain Monte Carlo methods [36, 79] are used to design surveillance strategies. In [36] convexity results for symmetric matrices are utilized to further optimize those strategies. Deterministic policies have been used to minimize the visit frequencies in a graph [29, 80], however, a

main result of [79] shows that deterministic policies are ill-suited when designing strategies with arbitrary constraints on those visit frequencies. We take an alternate approach and design policies using Markov chains with minimal hitting time.

The *hitting time* of a random walk governed by a Markov chain, is the expected time taken by a random walker to travel between *any* two nodes in a network. For a single finite discrete-time Markov chain, this quantity is also well-known as the *Kemeny constant* of the Markov chain, the *mean first passage time* of a Markov chain or, for the case of reversible Markov chains, the *eigentime* of a Markov chain. We refer to this quantity as the *first-hitting time* or simply *hitting time* of a Markov chain, due both to the descriptive nature of this coinage as well as its prevalence in the literature. The hitting time of a Markov chain for a finite irreducible Markov chain first appeared in [47], however, it was rediscovered for finite *reversible* Markov chains in [11]. Since its original discovery, the hitting time has been further developed by several groups [45, 49, 70]. The authors of [45, 53] give bounds on the hitting time for various graph topologies and in [18] an alternate formulation is explored. Recently, the authors of [70] extended the notion of the hitting time to networks with travel distances and provide a scheme for its optimization, and the authors of [2] provided the first formulation of the hitting time for continuous time reversible Markov chains.

The hitting time is closely related to several other well-studied Markov chain properties. We focus on four quantities that are of particular interest, however, several others exist. The first and most closely related quantity is the *pairwise* hitting time between two nodes, which is the expected time to travel between a *specified* pair of nodes. Clearly, the hitting time is simply the expectation of all possible pairwise hitting times of a Markov chain. Using the relation between reversible Markov chains and electrical networks [25] the authors of [82] give analytic expressions for hitting times in terms of effective resistance. Using the electrical framework, closed form expressions for pairwise hitting times have also been given for special cases of certain Markov chains [67, 66], however, to the best of our knowledge no general closed form expression exists. Second, the *cover time* of a graph is the expected time it takes to reach every node in the graph at least once. This quantity is sometimes interpreted as a function of a single node, or more generally, in the context of an arbitrary set of nodes. There are several works relating the cover time to the hitting time of a transition matrix [61]; many of these works bound the cover time in terms of pairwise hitting time (most often the worst case pairwise hitting time). Third, the *mixing rate* of an irreducible Markov chain is the rate at which an arbitrary distribution converges to the chain's stationary distribution. The influential text [54] provides a detailed review of the mixing rate and of other notions of mixing. Recently, [49] refers to

the hitting time as the "expected time to mixing" and relates it to the mixing rate. Finally, the *Kirchhoff index* [50], also known as the *effective graph resistance* [28], is a related metric quantifying the distance between pairs of vertices in an electric network. The relationship between electrical networks and random walks on graphs is explained elaborately in [25]. For an arbitrary graph, the Kirchoff index and the hitting time can be calculated from the eigenvalues of the conductance matrix and the transition matrix, respectively. The relationship between these two quantities for regular graphs is established in [65].

In this work, we also look at the hitting time of multiple random walkers. More specifically, we analyze the expected first time to reach any single node in a network given that there are an arbitrary number of random walkers in that network. Recently, the authors of [27] look at bounds on hitting times and cover times for multiple random walkers. In [3] alternate bounds on cover time are formulated for reversible Markov chains and tight bounds were explored in [31, 32]. The authors of [22, 21] find solutions for cover times in the limit as the number of nodes in the graph becomes infinite. However, as far as we can discern, a key assumption made by all work presented thus far in the literature is that results are based on k copies of a *simple* random walk over a single graph. We make no such assumptions in our work; every random walker can move according to a different and arbitrary random walk and need not share the same underlying

graph topology. Also, as opposed to prior work, our results are not bounds but exact analytic expressions.

To achieve our results, we utilize the notion of Kronecker graphs. Preliminary results for undirected Kronecker graphs were introduced in [84], showing conditions under which the Kronecker product of two graphs generates a connected graph. In [51, 52] the authors consider and analyze special Kronecker graphs that are created by products of the same $n \times n$ edge matrix in order to model large networks and also introduce the concept of "stochastic" Kronecker graphs to generate large networks. This method of generating networks is further refined for the special case of $n = 2$ in [59, 76]. In our work we also utilize the notion of Kronecker products between stochastic matrices, but this should not be confused with the notion of "stochastic" Kronecker graphs previously mentioned. In this work we are *not* attempting to generate network models, but instead are utilizing novel aspects of Kronecker products and stochastic matrices that have, to the best of our knowledge, not been deeply explored; the ideas presented here would most closely be linked to that of [84] of the previously mentioned works.

1.1.2 Partitioning and coverage control

A broad discussion of partitioning and coverage control is presented in [13] which builds on the classic work of Lloyd [56] on algorithms for optimal quan-

tizer design through "centering and partitioning." The Lloyd-type approach was first adapted for distributed coverage control in [24] and has since seen many variations, including non-convex environments [73, 6] and self-triggered coverage algorithms [62].

Many existing coverage control algorithms assume that robots can communicate peer-to-peer [24], but in some environments this is impractical. For example, underwater acoustic communication between ocean gliders is very low bandwidth and hilly or urban terrain can block radio communication. Instead, we present a coverage control algorithm for a team of robots which collectively maintain complete coverage of the environment and individually have only occasional contact with a central base station. This *one-to-base-station* communication model can represent ocean gliders surfacing to communicate with a tower [72], UAV data mules that periodically visit ground robots [77], or cost-mindful use of satellite or cellular communication. Our algorithm optimizes the response time of the team to service requests in a non-convex environment represented by a graph, with optimality defined by relevant "multi-center" cost functions for overlapping territories. Early work in coverage control of discrete non-convex domains (represented by graphs) is presented in [26]. Discrete coverage problems are closely related to the literature on data clustering and k-means [46], as well as the facility location or k-center problem [83].

This work also utilizes the notion of generalized Voronoi partitions. Results on specific manifestations of generalized Voronoi partitions and partitioning can be found in [64]. Results on the existence of power diagrams (a special generalized Voronoi partition manifestation) with area-constraints, along with a method to determine them are presented in [71]. More detailed results on existence of generalized Voronoi partitions for arbitrary area constraints are presented in [23]. Linear programming is used to handle generalized Voronoi partitions in [17] for fixed agents, showing that generalized Voronoi partitions are optimal for a certain class of multicenter functions. Similar results are obtained in [23], together with a discrete-time algorithm to solve the problem of optimal deployment of agents, while satisfying constraints on the areas.

1.2 Contributions

In this section, we highlight the contributions of the thesis organized by chapter.

Chapter 2 Before stating our contributions, it is worth mentioning that all work to date on the *hitting time* and mixing rate of a Markov chain on a graph has been completed under the assumption of homogeneous travel time along the edges of the graph. The contributions of this chapter are then six fold. First,

we provide a convex optimization framework to minimize the hitting time of a reversible Markov chain given the underlying graph topology of the random walk and the desired stationary distribution. Second, using doubly-weighted graphs we extend the formulation of the *hitting time* to the network environments with non-homogeneous travel times, a generalization not yet looked at in the literature. We denote this extension the *weighted hitting time*. Third, we derive a closed form solution for the *weighted hitting time* and show its relation to the hitting time. Fourth, we provide a convex optimization framework to minimize the weighted hitting time of a Markov chain with desired stationary distribution. Fifth, we provide a semidefinite program (SDP) formulation for the optimization of the hitting time and the weighted hitting time. Finally, we look at two stochastic surveillance scenarios; in the first scenario we provide a setup in which minimizing the weighted hitting time leads to the optimal Markov-chain strategy. In the second surveillance scenario we establish through numerical simulation that the Markov chain with the minimum weighted hitting time performs substantially better compared with other well-known Markov chains (i.e., the fastest mixing chain and the Metropolis-Hastings Markov chain).

Chapter 3 There are several key contributions of this chapter. First, we provide a novel method for the computation of the hitting time of a Markov chain. In

the process, we also provide the first closed form method for calculating pairwise hitting times between any two nodes for an arbitrary irreducible Markov chain. To the best of our knowledge, results for determining pairwise hitting times until now require restriction to reversible Markov chains, where knowledge of the Markov chain stationary distribution is known in advance. Our solution has no such restrictions. Second, we extend the notion of pairwise hitting time to an irreducible Markov chain with travel distances between nodes. Third, we define and provide the first closed form solution for computing the hitting time given multiple (different) Markov chains on the same graph. We denote this extension the 'group' hitting time. Fourth, our results also allow for the extension and calculation of the pairwise hitting time to the hitting time between any set of nodes for multiple random walkers; for any combination of specified starting nodes, we can calculate the first hitting time to a specified desired node. Fifth, we further extend the notion of group hitting time and hitting times between sets of nodes from multiple random walks on the same graph to random walks on multiple subgraphs. Finally, we provide a detailed numerical analysis to build intuition on the transition matrices that generate a minimal group hitting time. Before stating the paper organization, it is worthwhile to note that we achieve our analytical results by introducing a method of proof that utilizes the Kronecker product; thus our

work not only provides results in Markov chain behavior, but also gives general insight into Kronecker graphs and stochastic matrices.

Chapter 4 The contributions of this chapter are four-fold. First, we present the first coverage control algorithm for an asynchronous one-to-base-station communication model. This model is realistic and relevant for a variety of application domains. We handle the time delay between when robots communicate with the base station using overlapping regions instead of a partition. The algorithm can be adapted for various cost-functions and also allows for heterogeneity among agents. Second, we prove that the algorithm converges to a centroidal Voronoi partition in finite time for two relevant cost-functions. Our Lyapunov argument is based on an adaptation of the standard partition-based coverage cost function. Third, we introduce the notion of Pareto-optimal partitions and provide a cost-function to achieve such a partition using our algorithm. Finally, we describe how the algorithm can seamlessly handle changes in the environment as well as unscheduled arrival, departure or change in functionality of robots from the team. This feature leverages overlapping regions, and also eases integration of coverage control with task servicing.

Chapter 5 The contributions of this chapter are four-fold. First, we design a provably correct, spatially-distributed continuous-time algorithm to compute

area-constrained generalized Voronoi partitions of a convex environment. Our approach improves upon the work done in [71] for power diagrams both in generality and in numerical stability. Second, we build on work in [23], by introducing a continuous-time spatially-distributed algorithm to compute centroidal area-constrained generalized Voronoi partitions of a convex environment. Although an approximate method is proposed in [71] to achieve this for power diagrams, here we introduce a simpler generalized method which bridges the gap between [71] and [23] in that it is the continuous-time analogue to [23]. More precisely, the continuous-time algorithm presented in this paper and the discrete-time algorithm in [23] both converge to the set of centroidal area-constrained Voronoi partitions, which shall be formally defined below. Our proposed continuous-time algorithms are not only of academic interest in their own right, but also fill application gaps where a discrete-time algorithm may fall short. One such application is in mobile robotic networks, when there is a need to have continuous adaptive coverage and the understanding of the region's underlying density distribution changes frequently. For such cases, in the discrete time setting it is possible that, before the agents arrive at their optimal configuration, the optimal configuration has changed due to a new understanding of the density distribution. Moreover, the path that is taken may cause coverage cost to increase temporarily, since only the configuration is optimal not the path taken. In the continuous-time setting,

the agents are always moving in a path that improves their coverage cost, and a change in the underlying distribution simply means a correction in their path. Third, we introduce a practical method for implementation of our algorithms, and show their performance in simulation. Finally, as theoretical contributions, we prove that the set of mappings that generate area-constrained partitions are unique up to translation and we compute several novel partial derivatives of relevant operators. Specifically, we generalize a classic result about multi-center optimization: the partial derivative of the multicenter function evaluated at area-constrained Voronoi partitions has the same direction as the vector connecting the robot locations to the centroids of their regions.

Chapter 2

The Weighted Hitting Time of a Random Walk

This chapter is organized as follows. In Section 2.1 we summarize the notation which we use in this chapter and in Chapter 3, and briefly review properties of Markov chains. In Section 2.2 we give relevant background for the hitting time and present our results for its minimization. In Section 2.3 we introduce and provide detailed characterization of the weighted hitting time as well as its minimization. In Section 2.4 we provide practical surveillance applications of the weighted hitting time. In the final section we summarize our findings.

2.1 Notation and Markov chains

We use the notation $A = [a_{i,j}]$ to denote a matrix A with the element $a_{i,j}$ in its i-th row and j-th column and, unless otherwise indicated, use bold-faced letters to

denote vectors. Letting $\delta_{i,j}$ denote the Kronecker delta, $A_{\rm d} = [\delta_{i,j} a_{i,j}]$ represents the diagonal matrix whose diagonal elements are the diagonal elements of the matrix A. The column vector of all ones and length n is denoted by $\mathbb{1}_n \in \mathbb{R}^{n\times 1}$ and I represents the identity matrix of appropriate dimension. We use diag[$\boldsymbol{b}$] to denote the diagonal matrix generated by vector $\boldsymbol{b}$ and $\mathbf{Tr}[A]$ to denote the trace of matrix A.

A *Markov chain* is a sequence of random variables taking value in the *finite* set $\{1, \ldots, n\}$ with the Markov property, namely that, the future state depends only on the present state; see [43, 47] for more details. Let $X_k \in \{1, \ldots, n\}$ denote the location of a random walker at time $k \in \{0, 1, 2, \ldots\}$. We are now ready to summarize some terminology and results for Markov chains. (1) A Markov chain is *time-homogeneous* if $\mathbb{P}[X_{n+1} = j | X_n = i] = \mathbb{P}[X_n = j | X_{n-1} = i] = p_{i,j}$, where $P \in \mathbb{R}^{n\times n}$ is the transition matrix of the Markov chain. (2) The vector $\boldsymbol{\pi} \in \mathbb{R}^{n\times 1}$ is a *stationary distribution* of P if $\sum_{i=1}^{n} \boldsymbol{\pi}_i = 1$, $0 \le \boldsymbol{\pi}_i \le 1$ for all $i \in \{1, \ldots, n\}$ and $\boldsymbol{\pi}^T P = \boldsymbol{\pi}^T$. (3) A time-homogeneous Markov chain is said to be *reversible* if $\boldsymbol{\pi}_i p_{i,j} = \boldsymbol{\pi}_j p_{j,i}$, for all $i, j \in \{1, \ldots, n\}$. For reversible Markov chains, $\boldsymbol{\pi}$ is always a steady state distribution. (4) A Markov chain is *irreducible* if there exists a t such that for all $i, j \in \{1, \ldots, n\}, (P^t)_{i,j} > 0$. (5) If the Markov chain is *irreducible*, then there is a unique stationary distribution $\boldsymbol{\pi}$, and the corresponding eigenvalues of the transition matrix, λ_i for $i \in \{1, \ldots, n\}$, are such that $\lambda_1 = 1$, $|\lambda_i| \le 1$ and

$\lambda_i \neq 1$ for $i \in \{2, \ldots, n\}$. For further details on irreducible matrices and about results (4) and (5) see [60, Chapter 8]. In this thesis we consider finite irreducible time-homogeneous Markov chains.

2.2 The hitting time of a Markov chain and its minimization

Consider a undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P)$ with node set $\mathcal{V} := \{1, \ldots, n\}$, edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, and weight matrix $P = [p_{i,j}]$ with the property that $p_{i,j} \geq 0$ if $(i, j) \in \mathcal{E}$ and $p_{i,j} = 0$ otherwise. We interpret the weight of edge (i, j) as the probability of moving along that edge. Therefore, element $p_{i,j}$ in the matrix represents the probability with which the random walk visits node j from node i. Throughout this document we assume that the underlying undirected graph $(\mathcal{V}, \mathcal{E})$ associated to the transition probabilities P is connected.

In this section we look into a discrete-time random walk defined by a Markov chain on a graph $\mathcal{G}$. At each time step (*hop*) of the random walk we move to a new node or stay at the current node according to the transition probabilities described by a transition matrix P as discussed above. We do this with three objectives in mind. The first objective is to analyze the random walk and characterize the average visit time between nodes in the graph. The second objective is to minimize

the average visit time between any two nodes and the final is to achieve a long term (infinite horizon) visit frequency $\boldsymbol{\pi}_i$ at node i. Here, the frequency $\boldsymbol{\pi}_i$ is the ratio of visits to node i divided by the total number of visits to all nodes in the graph. Throughout the paper, we describe the random walk using realizations of a Markov chain with transition matrix $P = [p_{i,j}]$.

2.2.1 The hitting time for a weighted graph

Let $X_k \in \{1, \ldots, n\}$ denote the location of the random walker at time $k \in \{0, 1, 2, \ldots\}$. For any two nodes $i, j \in \{1, \ldots, n\}$, the *first passage time from i to j*, denoted by $T_{i,j}$, is the first time that the random walker starting at node i at time 0 reaches node j, that is,

$$T_{i,j} = \min\{k \geq 1 \mid X_k = j \text{ given that } X_0 = i\}.$$

It is convenient to introduce the shorthand $m_{i,j} = \mathbb{E}[T_{i,j}]$, and to define the *mean first passage time matrix* M to have entries $m_{i,j}$, for $i, j \in \{1, \ldots, n\}$. The *mean first passage time from start node i*, denoted by $\boldsymbol{h}_i$, is the expected first passage time from node i to a random node selected according to the stationary distribution $\boldsymbol{\pi}$, i.e.,

$$\boldsymbol{h}_i = \sum_{j=1}^{n} m_{i,j} \boldsymbol{\pi}_j.$$

It is well known [45] that the mean first passage time from a start node is independent of the start node, that is, $\boldsymbol{h}_i = \boldsymbol{h}_j$ for all $i, j \in \{1, \ldots, n\}$. Accordingly, we let $H = \boldsymbol{h}_i$, for all $i \in \{1, \ldots, n\}$, denote the *hitting time*, also known as the Kemeny constant, of the Markov chain.

Next, we provide formulas for these quantities. By definition, the first passage time from i to j satisfies the recursive formula:

$$T_{i,j} = \begin{cases} 1, & \text{with probability } p_{i,j}, \\ T_{k,j} + 1, & \text{with probability } p_{i,k}, k \neq j. \end{cases}$$

Taking the expectation, we compute

$$m_{i,j} = p_{i,j} + \sum_{k=1,k\neq j}^{n} p_{i,k}(m_{k,j} + 1) = 1 + \sum_{k=1,k\neq j}^{n} p_{i,k} m_{k,j},$$

or in matrix notation,

$$(I - P)M = \mathbb{1}_n \mathbb{1}_n^T - P M_{\mathrm{d}}, \tag{2.1}$$

where P is the transition matrix of the Markov chain. If the Markov chain is irreducible with stationary distribution $\boldsymbol{\pi}$, then one can show $M_{\mathrm{d}} = \mathrm{diag}[\{1/\boldsymbol{\pi}_1, \ldots, 1/\boldsymbol{\pi}_n\}]$, and

$$\boldsymbol{\pi}^T M \boldsymbol{\pi} = \sum_{i=1}^{n} \boldsymbol{\pi}_i \sum_{j=1}^{n} \boldsymbol{\pi}_j m_{i,j} = \sum_{i=1}^{n} \boldsymbol{\pi}_i \boldsymbol{h}_i = H.$$

Clearly, the hitting time can be written as the function $P \mapsto H(P)$, however, to ease notation we simply write H and use $H(P)$ only when we wish to emphasize its dependence on P.

The hitting time $H = \boldsymbol{\pi}^T M \boldsymbol{\pi}$ can be computed from equation (2.1) or can be expressed as a function of the eigenvalues of the transition matrix P as is stated in the following theorem [45].

Theorem 1. *(Hitting time of an irreducible Markov chain): Consider a Markov chain with an irreducible transition matrix P with eigenvalues $\lambda_1 = 1$ and λ_i, $i \in \{2, \ldots, n\}$. The hitting time of the Markov chain is given by*

$$H = 1 + \sum_{i=2}^{n} \frac{1}{1 - \lambda_i}.$$

Using Theorem 1, we derive the following equivalent expression for *reversible* Markov chains in terms of the trace of a symmetric positive definite matrix. Before stating our result, we first introduce some notation. Given a stationary distribution vector $\boldsymbol{\pi} \in \mathbb{R}^{n \times 1}$ for a Markov chain with transition matrix $P \in \mathbb{R}^{n \times n}$, we define the matrix $\Pi \in \mathbb{R}^{n \times n}$ as $\Pi = \text{diag}[\boldsymbol{\pi}]$ and the vector $\boldsymbol{q} \in \mathbb{R}^{n \times 1}$ as $\boldsymbol{q}^T = (\sqrt{\boldsymbol{\pi}_1}, \ldots, \sqrt{\boldsymbol{\pi}_n})$. We are now ready to state our first result.

Theorem 2. *(Hitting time of a reversible irreducible Markov chain): The hitting time of a reversible irreducible Markov chain with transition matrix P and stationary distribution $\boldsymbol{\pi}$ is given by*

$$H = \mathbf{Tr}\left[(I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)^{-1}\right]. \tag{2.2}$$

Proof. We start by noting that P is an irreducible row-stochastic matrix therefore the eigenvalues of P are $\{\lambda_1 = 1, \lambda_2, \ldots, \lambda_n\}$, where $|\lambda_i| \leq 1$ and $\lambda_i \neq 1$ for $i \in \{2, \ldots, n\}$. It follows that the eigenvalues of $(I-P)$ are $\{0, 1-\lambda_2, \ldots, 1-\lambda_n\}$. Since P is irreducible and reversible, there exists a stationary distribution $\boldsymbol{\pi} \in \mathbb{R}^n_{>0}$ implying Π is invertible and that $\Pi^{1/2}(I-P)\Pi^{-1/2} = I-\Pi^{1/2}P\Pi^{-1/2}$ is symmetric. It can easily be verified that $I-P$ and $I-\Pi^{1/2}P\Pi^{-1/2}$ have the same eigenvalues and that $\boldsymbol{q}$ is the eigenvector associated with the eigenvalue at 0. Next, notice the matrix $(I-\Pi^{1/2}P\Pi^{-1/2}+\boldsymbol{q}\boldsymbol{q}^T)$ is symmetric and that $(I-\Pi^{1/2}P\Pi^{-1/2}+\boldsymbol{q}\boldsymbol{q}^T)\boldsymbol{q} = \boldsymbol{q}$. Therefore, $(I-\Pi^{1/2}P\Pi^{-1/2}+\boldsymbol{q}\boldsymbol{q}^T)$ has an eigenvalue at 1 associated with the vector $\boldsymbol{q}$. Since $(I - \Pi^{1/2}P\Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)$ is symmetric, the eigenvectors corresponding to different eigenvalues are orthogonal; implying for eigenvector $\boldsymbol{v} \neq \boldsymbol{q}$ that $(I - \Pi^{1/2}P\Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)\boldsymbol{v} = (I - \Pi^{1/2}P\Pi^{-1/2})\boldsymbol{v}$ since the eigenvalue at 1 is simple. Therefore, the eigenvalues of $(I - \Pi^{1/2}P\Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)$ are $\{1, 1-\lambda_2, \ldots, 1-\lambda_n\}$. Thus, $H = \mathbf{Tr}\left[(I - \Pi^{1/2}P\Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)^{-1}\right] = 1 + 1/(1-\lambda_2) + \ldots + 1/(1-\lambda_n) = H$. □

Given the above result, we are now ready to state our first problem of interest.

Problem 1. *(Optimizing the hitting time of a reversible Markov chain): Given the stationary distribution $\boldsymbol{\pi}$ and graph $\mathcal{G}$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$, determine*

the transition probabilities $P = [p_{i,j}]$ solving:

$$\begin{aligned} \text{minimize} \quad & \mathbf{Tr}\left[(I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)^{-1}\right] \\ \text{subject to} \quad & \sum_{j=1}^{n} p_{i,j} = 1, \ \textit{for each } i \in \{1, \ldots, n\} \\ & \boldsymbol{\pi}_i p_{i,j} = \boldsymbol{\pi}_j p_{j,i}, \ \textit{for each } (i,j) \in \mathcal{E} \\ & 0 \leq p_{i,j} \leq 1, \ \textit{for each } (i,j) \in \mathcal{E} \\ & p_{i,j} = 0, \ \textit{for each } (i,j) \notin \mathcal{E}. \end{aligned} \tag{2.3}$$

Remark 3. *All feasible solutions P to Problem 1 are inherently irreducible transition matrices: when P is not irreducible, the matrix $(I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)$ is not invertible. Moreover, a feasible point always exists since the Metropolis-Hastings algorithm applied to any irreducible transition matrix associated with $\mathcal{G}$, generates a reversible transition matrix which is irreducible and satisfies the stationary distribution constraint [39].*

The following theorem establishes the convexity of the hitting time for transition matrices with fixed stationary distribution.

Theorem 4 (Convexity of Problem 1)**.** *Let $\mathcal{P}_{\boldsymbol{\pi}}$ denote the set of matrices associated to irreducible reversible Markov chains with stationary distribution $\boldsymbol{\pi}$. Then, $\mathcal{P}_{\boldsymbol{\pi}}$ is a convex set and $P \mapsto H(P)$ is a convex function over $\mathcal{P}_{\boldsymbol{\pi}}$.*

Proof. Let $\mathcal{S}$ denote the set of symmetric positive definite matrices, for any stationary distribution $\boldsymbol{\pi} \in \mathbb{R}^n_{>0}$, denote the set $\mathcal{S}_{\mathcal{P},\boldsymbol{\pi}} := \{I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T \mid P \in$

$\mathcal{P}_{\boldsymbol{\pi}}\}$. We begin by showing that $\mathcal{P}_{\boldsymbol{\pi}}$ is a convex set. Let $P_1, P_2 \in \mathcal{P}_{\boldsymbol{\pi}}$, then $\mathcal{P}_{\boldsymbol{\pi}}$ is convex if for an arbitrary $\theta \in [0,1]$ that

$$\theta P_1 + (1-\theta)P_2 = P_3 \in \mathcal{P}_{\boldsymbol{\pi}}. \tag{2.4}$$

Pre and post multiplying (2.4) by $\Pi^{1/2}$ and $\Pi^{-1/2}$, respectively, we have that $\theta\Pi^{1/2}P_1\Pi^{-1/2} + (1-\theta)\Pi^{1/2}P_2\Pi^{-1/2} = \Pi^{1/2}P_3\Pi^{-1/2}$. Then $\Pi^{1/2}P_3\Pi^{-1/2}$ is symmetric since $\Pi^{1/2}P_1\Pi^{-1/2}$ and $\Pi^{1/2}P_2\Pi^{-1/2}$ are symmetric. Pre multiplying (2.4) by $\boldsymbol{\pi}^T$ we easily verify that the stationary distribution P_3 is $\boldsymbol{\pi}^T$ and similarly, post multiplying by $\mathbb{1}_n$ verifies that P_3 is row stochastic. Finally taking the left hand side of (2.4) to the n-th power gives $(\theta P_1 + (1-\theta)P_2)^n = \theta^n P_1^n + (1-\theta)^n P_2^n + \zeta$, where ζ denotes the sum of all other terms in the expansion and has the property $\zeta_{i,j} \geq 0$ for all i, j since P_1 and P_2 are non-negative element-wise matrices. Moreover from irreducibility, there exists a sufficiently large N such that for $n > N$, $(P_1^n)_{i,j} > 0$ and $(P_2^n)_{i,j} > 0$ for all i, j, which implies $(P_3^n)_{i,j} > 0$, therefore $P_3 \in \mathcal{P}_{\boldsymbol{\pi}}$ and $\mathcal{P}_{\boldsymbol{\pi}}$ is convex.

Next we show that $\mathcal{S}_{\mathcal{P},\boldsymbol{\pi}} \subset \mathcal{S}$. From the proof of Theorem 2 we have for $P \in \mathcal{P}_{\boldsymbol{\pi}}$ that $I - \Pi^{1/2}P\Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T$ has eigenvalues $\{1, 1-\lambda_2, \ldots, 1-\lambda_n\}$, where λ_i for $i \in \{1, \ldots, n\}$ are the eigenvalues of P, where $\lambda_i \leq |1|$ for all i and $\lambda_i \neq 1$ for $i \in \{2, \ldots, n\}$. Therefore, all eigenvalues of $I - \Pi^{1/2}P\Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T$ are strictly greater than zero. Finally, since P is reversible $\Pi^{1/2}P\Pi^{-1/2}$ is symmetric implying $(I - \Pi^{1/2}P\Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)^T = I - \Pi^{1/2}P\Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T$ and so $\mathcal{S}_{\mathcal{P},\boldsymbol{\pi}} \subset \mathcal{S}$.

Finally, define the mapping $g : \mathcal{P}_{\boldsymbol{\pi}} \mapsto \mathcal{S}_{\mathcal{P},\boldsymbol{\pi}}$ by $g(X) = I - \Pi^{1/2} X \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T$. This is an affine mapping from the convex set $\mathcal{P}_{\boldsymbol{\pi}}$ to a subset of $\mathcal{S}$. From [34] we know that $\mathbf{Tr}[X^{-1}]$ is convex for $X \in \mathcal{S}$, therefore the composition with the affine mapping $g : \mathcal{P}_{\boldsymbol{\pi}} \mapsto \mathcal{S}_{\mathcal{P},\boldsymbol{\pi}} \subset \mathcal{S}$, $\mathbf{Tr}[g(X)^{-1}]$ is also convex [10, Chapter 3.2.2]. □

Problem 1 includes constraints on the stationary distribution of the transition matrix, a notion which has not been looked at in the literature before. The author of [49] provides bounds to determine the set of transition matrices such that H is minimized and [45] gives special matrices for which the optimal hitting time can be found, but these are all approached for the general setting with no constraint on the actual stationary distribution. In real-world settings, constraints on the stationary distribution are important and have many practical interpretations. For example, it is often desirable to visit certain regions more frequently than other based on each region's relative importance.

2.2.2 SDP framework for optimizing the hitting time

Here we show how Problem 1 can be equivalently rewritten as an SDP by introducing a symmetric slack matrix.

Problem 2. *(Optimizing the hitting time of a reversible Markov chain (SDP)): Given the stationary distribution $\boldsymbol{\pi}$ and graph $\mathcal{G}$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$,*

determine $X = [x_{i,j}]$ *and the transition probabilities* $P = [p_{i,j}]$ *solving:*

$$\text{minimize} \quad \mathbf{Tr}[X]$$

subject to

$$\begin{bmatrix} I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T & I \\ I & X \end{bmatrix} \succeq 0$$

$$\sum_{j=1}^{n} p_{i,j} = 1, \ \textit{for each } i \in \{1, \ldots, n\}$$

$$\boldsymbol{\pi}_i p_{i,j} = \boldsymbol{\pi}_j p_{j,i}, \ \textit{for each } (i,j) \in \mathcal{E}$$

$$0 \leq p_{i,j} \leq 1, \ \textit{for each } (i,j) \in \mathcal{E}$$

$$p_{i,j} = 0, \ \textit{for each } (i,j) \notin \mathcal{E}.$$

The first inequality constraint in Problem 2 represents a linear matrix inequality (LMI) and denotes that the matrix is positive semidefinite. Since the matrix in the LMI has off-diagonal entries equal to the identity matrix, it holds true if and only if X is positive definite and $(I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T) - X^{-1}$ is positive semidefinite [1, Theorem 1]. This implies $(I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)$ is positive definite and that $X \succeq (I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)^{-1}$. Therefore, the SDP given by Problem 2 minimizes the hitting time.

2.3 The weighted hitting time of Markov chain and its minimization

In most practical applications, distance/time traveled and service costs/times are important factors in the model of the system. We incorporate these concepts by allowing for an additional set of weighted edges in our graph in addition to the edge weights which describe the transition probabilities. Such a system can be represented by the doubly-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P, W)$, where $W = [\omega_{i,j}]$ is a weight matrix with the properties that: if $(i,i) \in \mathcal{E}$, then $\omega_{i,i} \geq 0$; if $(i,j) \in \mathcal{E}$, $i \neq j$, then $\omega_{i,j} > 0$; and if $(i,j) \notin \mathcal{E}$, then $\omega_{i,j} = 0$. The weighted adjacency matrix $P = [p_{i,j}]$ has the same interpretation as before as an irreducible row-stochastic transition matrix P which governs the random walk on the graph. An example of a doubly-weighted graph is shown in Figure 2.1. In the following, we will interpret $\omega_{i,j}$, $i \neq j$ as the time to travel between two nodes, i and j, in the graph and $\omega_{i,i}$ as the service time at node i. We discuss another motivating example and interpretation for $\omega_{i,j}$ in a later section.

Recall that $X_k = i$ denotes that the random walker is at node i at time k. If a sample trajectory of the random walk is $X_0 = i$, $X_1 = j$, $X_2 = k$, then the time instant at which a random walker arrives in state X_2 is $\omega_{i,j} + \omega_{jk}$. Thus the

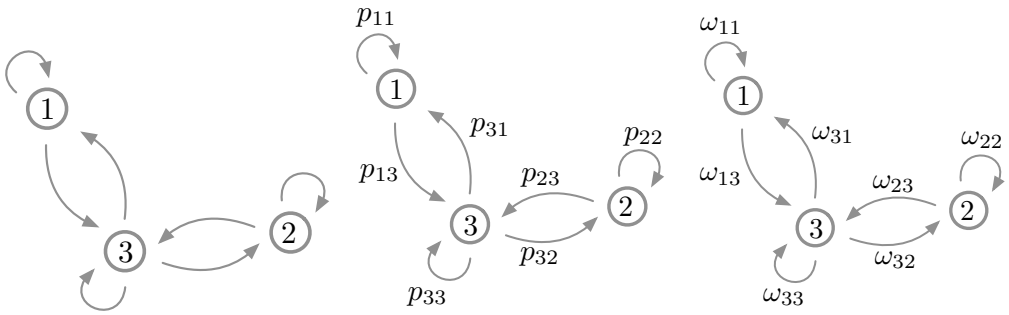

Figure 2.1: Example of a doubly-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P, W)$ with three nodes: (a) shows the edge set, $\mathcal{E}$, allowed for the graph with three nodes, (b) shows the probabilities, $p_{i,j}$ to move along each edge, and (c) shows the time (i.e., distance traveled), $\omega_{i,j}$ to move along each edge.

time interval between two consecutive steps of this random walk depends on the weighted adjacency matrix, W, of the graph and is not constant.

In the following analysis, we look at several characterization and optimization objectives: The first involves extending the notion of the hitting time to doubly-weighted graphs and providing a detailed characterization of this extension. The second involves the minimization of the hitting time of a doubly-weighted graph and the third involves characterization and minimization of the mean time to execute a single *hop*. The first and second objectives are motivated by the need to minimize visit times to nodes in the graph, and the third is motivated by the desire to minimize resource consumption when moving between nodes. We seek to design transition matrices P with stationary distribution $\boldsymbol{\pi}$ which optimize each problem. We start with the first objective.

2.3.1 The hitting time for a doubly-weighted graph

The mean first passage time for the Markov chain on a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P)$ by definition, is simply its hitting time. Recall that the mean first passage time for node i, defined by $\boldsymbol{h}_i$, is determined by taking the expectation over the first passage times $m_{i,j}$, from node i to all other nodes j. We present an analogous notion of the first passage time between two nodes on a doubly-weighted graph. We start with defining the first passage time random variable for a random walk on a doubly-weighted graph and provide a recursive formulation for its expectation.

As in Section 2.2.1, for any two nodes $i, j \in \{1, \ldots, n\}$, the *first passage time from i to j* is the first time that the random walker starting at node i at time 0 reaches node j, that is,

$$T_{i,j} = \min\Big\{ \sum_{n=0}^{k-1} w_{X_n, X_{n+1}},\ \text{for } k \geq 1 \mid X_k = j \text{ given that } X_0 = i \Big\}.$$

Lemma 5. *(First passage time for a doubly-weighted graph): Let $m_{i,j}(W) = \mathbb{E}[T_{i,j}]$ denote the mean first passage time to go from i to j for a graph with weight matrix W and transition matrix P. Then*

$$m_{i,j}(W) = p_{i,j}(\omega_{i,j}) + \sum_{k \neq j} p_{i,k}(m_{k,j}(W) + \omega_{i,k}), \tag{2.5}$$

or, in matrix notation,

$$(I - P)\mathcal{M} = (P \circ W)\mathbb{1}_n \mathbb{1}_n^T - P\mathcal{M}_{\mathrm{d}}, \tag{2.6}$$

where $(P \circ W)$ *is the element-wise product between* P *and* W *and where* $\mathcal{M}_{\mathrm{d}} = [\delta_{i,j} m_{i,j}(W)]$.

Proof. By its definition, the first passage time satisfies the recursive formula:

$$T_{i,j} = \begin{cases} \omega_{i,j}, & \text{with probability } p_{,ij}, \\ \omega_{i,k} + T_{k,j}, & \text{with probability } p_{i,k}, k \neq j. \end{cases} \tag{2.7}$$

Therefore, the results follows from taking the expectation:

$$\mathbb{E}[T_{i,j}] = \omega_{i,j} p_{i,j} + \sum_{k \neq j} p_{i,k} (\mathbb{E}[T_{k,j}] + \omega_{i,k}).$$

□

The matrix $\mathcal{M}$, which we call the *mean first passage time matrix for a doubly-weighted graph* thus satisfies an equation similar to (2.1) for the passage time matrix M of a graph with a single weight matrix, the transition matrix P. In fact, we see that equation (2.6) is equivalent to (2.1) when $w_{i,j} = 1$ for all $(i, j) \in \mathcal{E}$ (i.e., for an unweighted graph).

The random variable tracking the time interval between consecutive visits to a node has been referred to as the *refresh time* of that node [68] and $m_{i,i}(W)$ is the expected value of the refresh time for the random walk. We now obtain a relation between $\boldsymbol{\pi}$ and the refresh times $m_{i,i}(W)$.

Theorem 6 (Refresh times for doubly-weighted graphs)**.** *Consider a Markov chain on a doubly-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P, W)$ with stationary distribution $\boldsymbol{\pi}$ and associated mean first passage time matrix $\mathcal{M}$. The refresh time for node i is given by $m_{i,i}(W) = (\boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n)/\boldsymbol{\pi}_i$, implying that*

$$\mathcal{M}_{\mathrm{d}} = \boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n M_{\mathrm{d}}.$$

Proof. The stationary distribution of the transition matrix P is $\boldsymbol{\pi} \in \mathbb{R}^{n\times 1}$. Therefore, premultiplying equation (2.6) by $\boldsymbol{\pi}^T$, we obtain

$$0 = \boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n\mathbb{1}_n^T - \boldsymbol{\pi}^T\mathcal{M}_{\mathrm{d}},$$

where the left hand side of equation (2.6) is zero since $\pi^T(I - P) = \pi^T - \pi^T = 0$. Now we have that $(\boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n)\mathbb{1}_n^T = \boldsymbol{\pi}^T\mathcal{M}_{\mathrm{d}}$. Since $\mathcal{M}_d$ is a diagonal matrix and $\boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n$ is a scalar, we get that $\boldsymbol{\pi}_i m_{i,i}(W) = \boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n$. Thus, dividing by $\boldsymbol{\pi}_i$ we have that $m_{i,i}(W) = \boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n/\boldsymbol{\pi}_i$, and in matrix form $\mathcal{M}_d = \boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n\mathrm{diag}(\{1/\boldsymbol{\pi}_1, \ldots, 1\ \boldsymbol{\pi}_n\}) = \boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n M_d$. □

This theorem implies that the refresh time $m_{i,i}(W)$ of the random walk is directly proportional to the visit frequencies $1/\boldsymbol{\pi}_i$. Therefore, the relative visit frequencies of one node compared to another are not a function of the weight matrix W and only depend on the stationary distribution of the transition matrix P.

We now investigate the properties of the hitting time of the weighted random walk. The hitting time for a doubly-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P, W)$ with associated passage times matrix $\mathcal{M}$ is given by $H_W = \boldsymbol{\pi}^T \boldsymbol{h}_W$, where $\boldsymbol{h}_W = \mathcal{M}\boldsymbol{\pi}$ is the vector of *first passage times* and the $i-$th entry $\boldsymbol{h}_{W,i}$ in $\boldsymbol{h}_W$ denotes the mean time to go from i to any other node. We refer to the hitting time for a doubly-weighted graph, H_W, as the *weighted hitting time*. We now provide an analytic expression for the vector $\boldsymbol{h}_W \in \mathbb{R}^{n\times 1}$.

Lemma 7. *(First passage times for a doubly-weighted graph): Given a Markov chain on a doubly-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P, W)$ with stationary distribution $\boldsymbol{\pi}$ and associated first passage time matrix $\mathcal{M}$, the following equality holds:*

$$(I - P)\boldsymbol{h}_W = (P \circ W)\mathbb{1}_n - \mathbb{1}_n \boldsymbol{\pi}^T (P \circ W)\mathbb{1}_n, \tag{2.8}$$

where $\boldsymbol{h}_W = \mathcal{M}\boldsymbol{\pi}$.

Proof. Post multiplying equation (2.6) on both sides by $\boldsymbol{\pi}$ gives

$$\begin{aligned}(I - P)\mathcal{M}\boldsymbol{\pi} =& (P \circ W)\mathbb{1}_n \mathbb{1}_n^T \boldsymbol{\pi} - P\mathcal{M}_{\mathrm{d}}\boldsymbol{\pi},\\ (I - P)\boldsymbol{h}_W =& (P \circ W)\mathbb{1}_n - P(\boldsymbol{\pi}^T (P \circ W)\mathbb{1}_n)\mathbb{1}_n\\ =& (P \circ W)\mathbb{1}_n - \mathbb{1}_n \boldsymbol{\pi}^T (P \circ W)\mathbb{1}_n.\end{aligned}$$

□

The right hand side of (2.8) gives the insight that, in general, $\boldsymbol{h}_{W,i} \neq \boldsymbol{h}_{W,j}$ on the doubly-weighted graph, unlike the counterpart for the single-weighted graph (where $\boldsymbol{h}_i = H$ for all $i \in \{1, \ldots, n\}$). Interestingly enough however, there does exist a relation between the weighted hitting time H_W and the hitting time H as is stated in the following theorem, whose proof is postponed to Section 8.

Theorem 8. *(Weighted hitting time of a Markov chain): For the doubly-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P, W)$, the weighted hitting time H_W of the Markov chain is given by*

$$H_W = \boldsymbol{\pi}^T (P \circ W) \mathbb{1}_n H, \tag{2.9}$$

where H is the hitting time associated with the irreducible transition matrix P with stationary distribution $\boldsymbol{\pi}$.

Remark 9. *The expected number of hops to go from one node to another in a Markov chain with transition matrix P is its hitting time. The expected distance travelled (and hence time taken) executing one hop is $\sum_i \boldsymbol{\pi}_i \sum_j p_{ij}\omega_{i,j} = \boldsymbol{\pi}(P \circ W)\mathbb{1}_n$. Hence, it is consistent with intuition that the expected time to travel from one node to another should be $H\boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n$ as is formally shown in Section 8.*

Given the above results, we are now ready to state another problem of interest.

Problem 3. *(Optimizing the weighted hitting time of a reversible Markov chain): Given the stationary distribution $\boldsymbol{\pi}$ and graph $\mathcal{G}$ with vertex set $\mathcal{V}$, edge set $\mathcal{E}$ and*

weight matrix W, determine the transition probabilities $P = [p_{i,j}]$ solving:

$$\begin{aligned}
&\text{minimize} \\
&\qquad \left(\boldsymbol{\pi}^T (P \circ W) \mathbb{1}_n\right) \left(\mathbf{Tr}\left[(I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)^{-1}\right]\right) \\
&\text{subject to} \quad \sum_{j=1}^{n} p_{i,j} = 1, \ \textit{for each } i \in \{1, \ldots, n\} \\
&\qquad\qquad \boldsymbol{\pi}_i p_{i,j} = \boldsymbol{\pi}_j p_{j,i}, \ \textit{for each } (i,j) \in \mathcal{E} \\
&\qquad\qquad 0 \leq p_{i,j} \leq 1, \ \textit{for each } (i,j) \in \mathcal{E} \\
&\qquad\qquad p_{i,j} = 0, \ \textit{for each } (i,j) \notin \mathcal{E}.
\end{aligned}$$

The following theorem establishes the convexity of the weighted hitting time for transition matrices with fixed stationary distribution.

Theorem 10 (Convexity of Problem 3)**.** *Given the graph $\mathcal{G}$ with vertex set $\mathcal{V}$, edge set $\mathcal{E}$ and weight matrix W, let $\mathcal{P}_{\mathcal{G},\boldsymbol{\pi}}$ denote the set of matrices associated with $\mathcal{G}$ that are irreducible reversible Markov chains with stationary distribution $\boldsymbol{\pi}$. Then, $\mathcal{P}_{\mathcal{G},\boldsymbol{\pi}}$ is a convex set and $P \mapsto \boldsymbol{\pi}^T (P \circ W) \mathbb{1}_n H(P)$ is a convex function over $\mathcal{P}_{\mathcal{G},\boldsymbol{\pi}}$.*

Proof. Let $\mathcal{S}$ denote the set of symmetric positive definite matrices, for any stationary distribution $\boldsymbol{\pi} \in \mathbb{R}^n_{>0}$, denote the set $\mathcal{S}_{\mathcal{G},\mathcal{P},\boldsymbol{\pi}} := \{I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T \mid P \in \mathcal{P}_{\mathcal{G},\boldsymbol{\pi}}\}$. The proof of convexity of the set $\mathcal{P}_{\mathcal{G},\boldsymbol{\pi}}$ is similar to that of the proof of $\mathcal{P}_{\boldsymbol{\pi}}$ in Theorem 4 and so is omitted for brevity. Then from the proof of Theorem 4 we know there exists an affine mapping $g(X) : \mathcal{P}_{\mathcal{G},\boldsymbol{\pi}} \mapsto \mathcal{S}_{\mathcal{G},\mathcal{P},\boldsymbol{\pi}}$ given

by $g(X) = I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T$. We know from [34] that $f(X) = \mathbf{Tr}[X^{-1}]$ is convex, therefore the perspective function $h(X,t) = \{tf(X/t) \mid t > 0\}$ is also convex [10, Chapter 3.2.6]. Moreover the composition of $h(X,t)$ with the affine mapping $g(X)$, $h(g(X),t)$ is also convex. Let $t = (\boldsymbol{\pi}^T (X \circ W) \mathbb{1}_n)^{1/2}$, and notice that $\boldsymbol{\pi}^T (X \circ W) \mathbb{1}_n > 0$ for $X \in \mathcal{P}_{\mathcal{G},\boldsymbol{\pi}}$ and therefore $t > 0$. Also notice that for a constant $k \in \mathbb{R}^n_{>0}$ and matrix $X \in \mathbb{R}^{n \times n}$ that $\mathbf{Tr}[(\frac{X}{k})^{-1}] = k\mathbf{Tr}[X^{-1}]$. Then $h(g(X),t) = t\mathbf{Tr}[(\frac{g(X)}{t})^{-1}] = t^2 \mathbf{Tr}[(g(X)^{-1}] = \boldsymbol{\pi}^T (X \circ W) \mathbb{1}_n \mathbf{Tr}[(I - \Pi^{1/2} X \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)^{-1}]$ for $X \in \mathcal{P}_{\mathcal{G},\boldsymbol{\pi}}$. □

2.3.2 SDP framework for optimizing the weighted hitting time

In a similar fashion to Problem 1, we can formulate Problem 3 as an SDP by introducing the symmetric slack matrix $X \in \mathbb{R}^{n \times n}$ and the scalar variable t as is shown in the following.

Problem 4. *(Optimizing the weighted hitting time of a reversible Markov chain (SDP)): Given the stationary distribution $\boldsymbol{\pi}$ and graph $\mathcal{G}$ with vertex set $\mathcal{V}$, edge*

set $\mathcal{E}$ and weight matrix W, determine $Y = [y_{i,j}]$, X and t solving:

$$\text{minimize} \quad \mathbf{Tr}[X]$$

$$\text{subject to}$$

$$\begin{bmatrix} t(I + \boldsymbol{q}\boldsymbol{q}^T) - \Pi^{1/2} Y \Pi^{-1/2} & I \\ I & X \end{bmatrix} \succeq 0$$

$$\sum_{j=1}^{n} y_{i,j} = t, \ \textit{for each } i \in \{1, \ldots, n\}$$

$$\boldsymbol{\pi}_i y_{i,j} = \boldsymbol{\pi}_j y_{j,i}, \ \textit{for each } (i,j) \in \mathcal{E}$$

$$0 \leq y_{i,j} \leq t, \ \textit{for each } (i,j) \in \mathcal{E}$$

$$y_{i,j} = 0, \ \textit{for each } (i,j) \notin \mathcal{E}$$

$$\boldsymbol{\pi}^T (Y \circ W) \mathbb{1}_n = 1$$

$$t \geq 0.$$

Then, the transition matrix P is given by $P = Y/t$.

As in Problem 2, the first inequality constraint in Problem 4 represents an LMI. Before noting when the LMI holds, first note that the constraints in Problem 4 imply that $Pt = Y$ and that $t = \frac{1}{\boldsymbol{\pi}^T (P \circ W) \mathbb{1}_n}$. Hence, using a similar argument as in Problem 2, the LMI constraint holds true if and only if $X \succeq \boldsymbol{\pi}^T (P \circ W) \mathbb{1}_n (I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)^{-1}$ where X and $\pi^T (P \circ W) \mathbb{1}_n (I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)^{-1}$ are both positive definite, and therefore the SDP given by Problem 4 minimizes the weighted hitting time.

2.3.3 Minimizing single hop distance

We now look at the objective of minimizing the mean time for a single *hop* of a random walk. At time k, let $S_{i,j}$ be the time required to transition from i to j in a single hop along an edge of length $\omega_{i,j}$. Then,

$$\begin{aligned}\mathbb{E}[S] &= \sum_{i=1}^{n}\sum_{j=1}^{n} p_{i,j}S_{i,j} \\ &= \sum_{i=1}^{n}\mathbb{P}[X_k = i]\sum_{j=1}^{n}\omega_{i,j}\mathbb{P}[X_{k+1} = j] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\boldsymbol{\pi}_i\omega_{i,j}p_{i,j} = \boldsymbol{\pi}^T(P\circ W)\mathbb{1}_n. \end{aligned} \tag{2.10}$$

The function $\boldsymbol{\pi}^T(P\circ W)\mathbb{1}_n$ is clearly convex in P. If one assumes that $\omega_{i,i} = 0$ for all $i \in \{1,\ldots,n\}$, then minimizing (2.10) over P yields the trivial solution $P = I$. We can take into account both the single hop distance as well as the hitting time to design a Markov chain as follows.

Problem 5. *(Optimizing hitting time and mean distance): Given the stationary distribution $\boldsymbol{\pi}$ and graph $\mathcal{G}$ with vertex set $\mathcal{V}$, edge set $\mathcal{E}$ and weight matrix W, and given user specified constant $\alpha \in [0,1]$, determine the transition probabilities*

$P = [p_{i,j}]$ *solving:*

$$\begin{aligned} \text{minimize} \quad & \alpha \mathbf{Tr}\left[(I - \Pi^{1/2} P \Pi^{-1/2} + \boldsymbol{q}\boldsymbol{q}^T)^{-1}\right] \\ & + (1-\alpha)\boldsymbol{\pi}^T (P \circ W)\mathbb{1}_n \\ \text{subject to} \quad & \sum_{j=1}^{n} p_{i,j} = 1, \ \textit{for each } i \in \{1,\ldots,n\} \\ & \boldsymbol{\pi}_i p_{i,j} = \boldsymbol{\pi}_j p_{j,i}, \ \textit{for each } (i,j) \in \mathcal{E} \\ & 0 \leq p_{i,j} \leq 1, \ \textit{for each } (i,j) \in \mathcal{E} \\ & p_{i,j} = 0, \ \textit{for each } (i,j) \notin \mathcal{E}. \end{aligned} \tag{2.11}$$

This problem is convex since the sum of two convex problems is convex, moreover, it can be extended to an SDP utilizing the LMI defined in Problem 2. In the context where $\omega_{i,i} = 0$ for all $i \in \{1,\ldots,n\}$, varying the parameter α can be used to control the connectivity of the Markov chain; the choice $\alpha = 1$ ensures connectivity, and the choice $\alpha = 0$ minimizes the single hop distance while making the graph disconnected.

2.4 Applications of the hitting time to surveillance

The results on hitting time for doubly-weighted graphs (i.e., the weighted hitting time) presented in this work provide a general framework which can po-

tentially be applied to the analysis and design in a myriad of fields. We focus on one application in particular; the intruder detection and surveillance problem. We look at two variations of this problem:

Scenario I In practical stochastic intruder detection and surveillance scenarios, there is often a desire to surveil some regions more than others (i.e.,have a pre- specified stationary distribution) while simultaneously minimizing the time any one region has to wait before it is serviced. For this setup, in every step of the random walk, the agent must move to a new region and execute its surveillance task. Assuming we are working on a doubly-weighted graph described by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P, W)$, let us also assume there is a fixed persistent intruder in the environment and it takes s_i time for an agent to determine if the intruder is in region $i \in \mathcal{V}$. Denote the time to move from region i to region j by $d_{i,j}$, where we can assume, without loss of generality, that $d_{i,i} = 0$. Then, we can define the weight corresponding to the edge from i to j as $\omega_{i,j} = d_{i,j} + s_j$. In this scenario we wish to minimize the expected time to capture the persistent intruder when no prior knowledge of their position is known.

Scenario II In this scenario we consider the intruder detection problem and adopt a similar setup to Scenario I, however, we now assume a set of intruders are distributed throughout the environment. Each intruder performs a malicious

activity in its host region for a fixed duration of time, which we call the *intruder life-time*, followed instantaneously by another intruder. The intruder is caught only if the agent is in the same region as the intruder for any duration of the *intruder life-time*. For simplicity only a single intruder appears at a time.

In the following subsections we analyze the performance of various stochastic surveillance policies as applied to Scenario I and Scenario II described above. More specifically, we gauge the performance of other well-known Markov chain design algorithms against the algorithms presented in this paper.

2.4.1 Optimal strategy for Scenario I

In the context of Scenario I the weighted hitting time of the agent's transition matrix, P, corresponds to the average time it takes to capture an intruder regardless of where the agent and intruder are in the environment. Therefore by definition of the hitting time, we have the following immediate corollary for Scenario I.

Corollary 11 (Optimal surveillance and service strategy)**.** *The transition matrix P which has minimal hitting time is the optimal strategy for the intruder detection problem described by* Scenario I.

This tells us that if we restrict ourselves to reversible Markov chains, then not only is the chain with minimal hitting time optimal, but given the results from Section 2.2 and 2.3, we can also optimally design this chain.

2.4.2 Numerical analysis of Scenario II

In Scenario II the transition matrix with minimum hitting time is not guaranteed to be the optimal policy, and thus to gauge its performance compared to other policies we analyze both homogeneous (uniform service/travel times) and heterogeneous environment cases. To compare performance we generate a random walk for the environment using the Metropolis-Hastings, fastest mixing Markov chain (FMMC) [9], and hitting time algorithms. While game theoretic frameworks [19, 8] also generate stochastic policies, they are based on assumptions on the intruder behavior. We avoid such assumptions here and, therefore, omit them from our comparative analysis.

We first look at the homogeneous case which is described by the discretized environment shown in Figure 2.2. We assume that a single surveillance agent executes a random walk in the environment, spending 1 time unit in each region, and that the agent transitions between two connected regions instantaneously. Also, we assume a uniform stationary distribution on the transition matrix (each node in the region must be visited with equal likelihood). The Markov chain generated

by the Metropolis-Hastings algorithm is generated by applying the algorithm to the random walk described by $p_{i,j} = 1/d_i$ for $i \neq j$ and $p_{i,j} = 0$ for $i = j$, where d_i is the degree of a node i (excluding self-loops) [39]. The *intruder life-time* is set to 66 time units and 500 intruders appear per simulation run (the sequence in which the intruders appear is determined before each simulation run), for a total simulation run of $33,000$ time units. As stated in the scenario description, the intruder is caught if the surveillance agent is in the same region as the intruder for any portion of the *intruder life-time*. Table 2.1 summarizes the statistical performance of each algorithm after 200 runs of the simulation and justifies our use of the hitting time algorithm as a valid surveillance strategy; the hitting time algorithm captures intruders more frequently than the other two algorithms, and its worst case performance is still better than the worst case performance of the other two algorithms. Although results for an intruder life-time of only 66 time units are presented here, we have found that the hitting time algorithm always outperforms the other two algorithms or is equivalent; the algorithms become equivalent in the limiting case, when the intruder life-times are so low that no intruder can be caught, or when the intruder-life times are so large that the intruder is always caught.

For the heterogeneous case, we work with the environment shown in Figure 2.3. In this environment the time taken by the agent to travel an edge is no longer

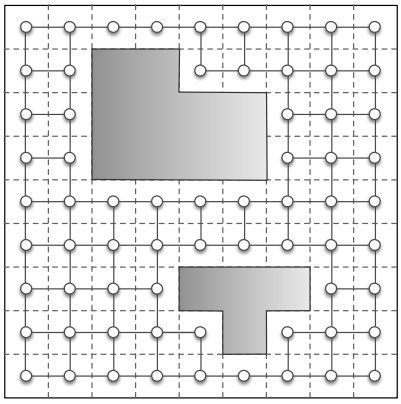

Figure 2.2: Environment with two obstacle represented by an unweighted graph.

Algorithm	**Min**	**Mean**	**Max**	**StdDev**	H
Hitting time	26.6	32.4	38.2	2.1	207
FMMC	24.6	29.8	34.4	1.9	236
Metropolis-Hastings	24.8	31.1	37	2.1	231

Table 2.1: Statistics on the percentage of intruders caught in 200 simulation runs for the environment in Fig. 2.2.

instantaneous and has travel weights as shown in the figure. Once in a new region, the agent is required to spend 1 time unit examining the region for malicious activities. We again assume that each node in the region must be visited with equal likelihood. We again also assume an intruder is caught if the surveillance agent is in the same region as a intruder for any portion of the *intruder life-time*, but now set the *intruder life-time* to 11 time units with a intruder appearing 500 times (total of 5500 time units per simulation run). Since the design of the FMMC and Metropolis-Hastings algorithms do not inherently account for non-uniform travel and service times, we also compare the performance of the random walk generated

by the weighted hitting time algorithm against the random walk generated by solving Problem 5 with $\alpha = 0.1$ (smaller α emphasizes minimizing the length of the average edge traveled in the graph). Table 2.2 summarizes the statistical performance of each algorithm after 200 runs of the simulation. The weighted hitting time algorithm's performance compared to the FMMC and Metropolis-Hastings stochastic policies in this scenario is significantly better than what was seen in the first scenario. We also note that the random walk policy determined by solving Problem 5 performs comparably to the weighted hitting time policy. This is to be expected since the Metropolis-Hastings and FMMC stochastic policies do not account for heterogeneous travel/service times on the graph. To get a better understanding of each algorithm's performance in this intruder scenario, the simulation is run for different intruder life-times, the results of which can be seen in Figure 2.4. There are several key items worth noting from the simulation. First, we see that the weighted hitting time algorithm significantly outperforms the other algorithms for a large range of intruder life-times. This matches our intuition since the algorithm inherently minimizes average travel time between nodes. Second, notice that the Markov chain generated by solving Problem 5 (with $\alpha = 0.1$) performs well for small intruder life-times but its performance plateaus quickly. This is due to the fact that the transition matrix generated by solving Problem 5 forces agents to stay at a given node rather than jump nodes; as one

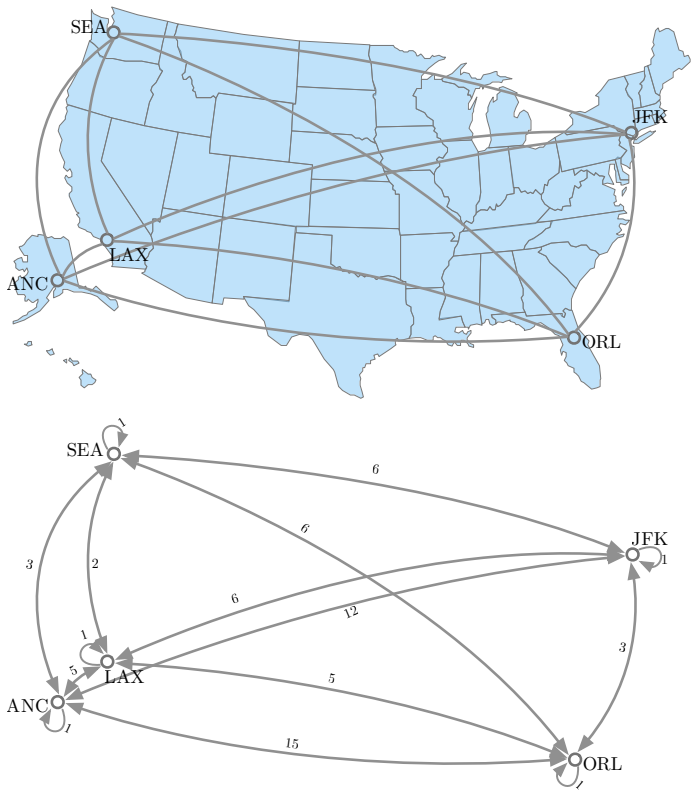

Figure 2.3: Various airport hub locations (top), and the corresponding weight map (bottom). Edge weights between two hubs account for travel time between hubs plus required service time once at hub. Self loops have no travel time so encompass only service time required at hub.

would suspect, once intruder life-times increase, a strategy which places emphasis on an agent that moves between regions will begin to perform relatively better. Finally, observe that as intruder life-time increases, the algorithms' capture rates start to converge. As in the homogeneous case, this is due to the fact that once the intruder's life-time is long enough, the agent will almost surely reach the intruder regardless of the policy it follows.

Algorithm	**Min**	**Mean**	**Max**	**StdDev**	H_W
Weighted Hitting Time	44	50.1	56	2.2	19.5
Hitting Time+Mean Dist.	40.6	47.1	53	2.2	23.1
FMMC	29.8	35.4	40.4	2.2	26.2
Metropolis-Hastings	30.4	36	41.6	2.1	26.5

Table 2.2: Statistics on the percentage of intruders caught in 200 simulation runs for the environment in Fig. 2.3.

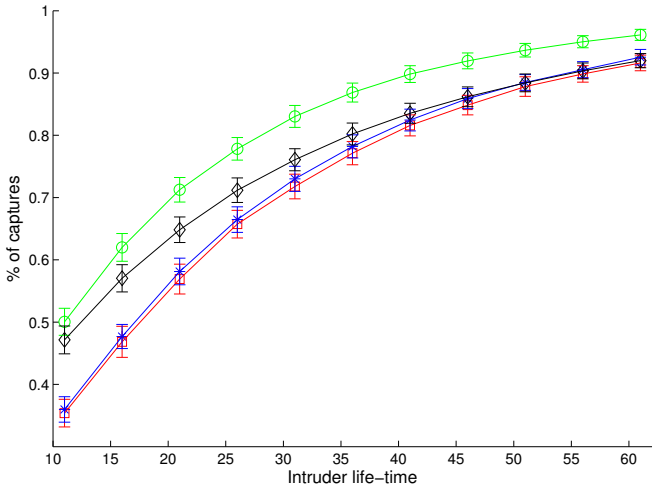

Figure 2.4: Percentage of intruders detected for varying intruder life-times by a surveillance agent executing a random walk according to the Markov chain generated by the hitting time algorithm (circle), FMMC algorithm (square), M-H algorithm (asterisk), and the Markov chain generated by solving Problem 5 (diamond). Average points and standard deviation error bars are taken over 200 runs, where the intruder appears 500 times for each run.

To solve for the Markov chains with minimal hitting time (Problem 2 and Problem 4) and with fastest mixing rate, we use CVX, a Matlab-based package for convex programs [37]. The execution time to solve each Markov chain for the

examples described above takes on the order of a couple seconds using a computer with a 1.3 GHz processor.

2.5 Summary

We have studied the problem of how to optimally design a Markov chain which minimizes the hitting time to go from one region to any other region in a connected environment. We have presented the first formulation of the hitting time for a doubly-weighted graph, which we refer to as the weighted hitting time, and have also provided a provably correct convex formulation for the minimization of both the hitting time and the weighted hitting time. Finally, we have shown that both problems can be written as SDPs and, moreover, have demonstrated the effectiveness of using a Markov chain with minimal hitting time as a surveillance policy as compared to other well-known Markov chain policies.

In the next chapter we look at extending the hitting time of a Markov chain to multiple random walkers.

2.6 Proofs and supplemental material

2.6.1 Proof of Theorem 8

Proof of Theorem 8. Let $\beta = \boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n$, then from Theorem 6 we have that $\mathcal{M}_{\mathrm{d}}$ from (2.6) can be written as βM_{d}. Now from Theorem 13 the general solution to (2.6) is

$$\mathcal{M} = G((P \circ W)\mathbb{1}_n\mathbb{1}_n^T - \beta P M_{\mathrm{d}}) + (I - G(I - P))U, \tag{2.12}$$

where G is a generalized inverse of $(I-P)$ (see Theorem 15) and U is an arbitrary matrix as long as the consistency condition

$$\big(I - (I - P)G\big)\big((P \circ W)\mathbb{1}_n\mathbb{1}_n^T - \beta P M_{\mathrm{d}}\big) = 0 \tag{2.13}$$

holds. Substituting (2.18) from Lemma 16 in for $(I - P)G$ in (2.13) gives that

$$\begin{aligned}
&\big(I - (I - P)G\big)\big((P \circ W)\mathbb{1}_n\mathbb{1}_n^T - \beta P M_{\mathrm{d}}\big) \\
&\qquad = \frac{\boldsymbol{t}\boldsymbol{\pi}^T}{\boldsymbol{\pi}^T\boldsymbol{t}}((P \circ W)\mathbb{1}_n\mathbb{1}_n^T - \beta P M_{\mathrm{d}}), \\
&\qquad = \frac{\boldsymbol{t}}{\boldsymbol{\pi}^T\boldsymbol{t}}(\boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n\mathbb{1}_n^T - \beta\boldsymbol{\pi}^T P M_{\mathrm{d}}), \\
&\qquad = \frac{\boldsymbol{t}}{\boldsymbol{\pi}^T\boldsymbol{t}}(\beta\mathbb{1}_n^T - \beta\mathbb{1}_n^T) = 0,
\end{aligned}$$

and so we have that the system of equations is consistent. This implies we can design U to reduce (2.12). We start by seeing how the second term in (2.12) can be reduced. Using (2.19) from Lemma 16 we have that $(I - G(I - P))U =$

$\frac{\mathbb{1}_n \boldsymbol{u}^T}{\boldsymbol{u}^T \mathbb{1}_n} U = \mathbb{1}_n \boldsymbol{k}^T$, where $\boldsymbol{k}^T = \frac{\boldsymbol{u}^T U}{\boldsymbol{u}^T \mathbb{1}_n}$. Hence, we can re-write (2.12) as

$$\mathcal{M} = G((P \circ W)\mathbb{1}_n \mathbb{1}_n^T - \beta P M_{\mathrm{d}}) + \mathbb{1}_n \boldsymbol{k}^T, \tag{2.14}$$

designing U reduces to designing the n elements of $\boldsymbol{k}$. Let $K = \mathrm{diag}[\boldsymbol{k}]$, then $\mathbb{1}_n \boldsymbol{k}^T = \mathbb{1}_n \mathbb{1}_n^T K$. Also, let $\Xi = \mathbb{1}_n \boldsymbol{\pi}^T$, where $\Xi_{\mathrm{d}} = \mathrm{diag}[\boldsymbol{\pi}]$. Utilizing these expressions in (2.14) and taking the diagonal elements gives

$$\begin{aligned}
&\big(\mathcal{M} = G((P \circ W)\mathbb{1}_n \mathbb{1}_n^T - \beta P M_{\mathrm{d}}) + \mathbb{1}_n \mathbb{1}_n^T K\big)_{\mathrm{d}}, \\
\implies &\beta M_{\mathrm{d}} = (G(P \circ W)\Xi)_{\mathrm{d}} M_{\mathrm{d}} - \beta (GP)_{\mathrm{d}} M_{\mathrm{d}} + K, \\
\implies &K = \beta M_{\mathrm{d}} - (G(P \circ W)\Xi)_{\mathrm{d}} M_{\mathrm{d}} + \beta (GP)_{\mathrm{d}} M_{\mathrm{d}},
\end{aligned}$$

where we use Lemma 14 to get the initial diagonal reduction. Substituting the expression for K into (2.14), and recalling that $\mathbb{1}_n \boldsymbol{k}^T = \mathbb{1}_n \mathbb{1}_n^T K$ gives

$$\begin{aligned}
\mathcal{M} = \big(&G(P \circ W)\Xi - \mathbb{1}_n \mathbb{1}_n^T (G(P \circ W)\Xi)_{\mathrm{d}} \\
&+ \beta(\mathbb{1}_n \mathbb{1}_n^T (GP)_{\mathrm{d}} - GP + \mathbb{1}_n \mathbb{1}_n^T)\big) M_{\mathrm{d}},
\end{aligned} \tag{2.15}$$

where we use the fact that $\mathbb{1}_n \mathbb{1}_n^T = \Xi M_{\mathrm{d}}$. Now from (2.19) we have that $I - G - GP = \frac{\mathbb{1}_n \boldsymbol{u}^T}{\boldsymbol{u}^T \mathbb{1}_n}$. Notice that $\mathbb{1}_n \mathbb{1}_n^T (I - G - GP)_{\mathrm{d}} = \mathbb{1}_n \mathbb{1}_n^T (\frac{\mathbb{1}_n \boldsymbol{u}^T}{\boldsymbol{u}^T \mathbb{1}_n})_{\mathrm{d}} = \frac{\mathbb{1}_n \boldsymbol{u}^T}{\boldsymbol{u}^T \mathbb{1}_n}$ and so this implies that $\mathbb{1}_n \mathbb{1}_n^T - \mathbb{1}_n \mathbb{1}_n^T G_{\mathrm{d}} + \mathbb{1}_n \mathbb{1}_n^T (GP)_{\mathrm{d}} = I - G + GP$, which implies that $\mathbb{1}_n \mathbb{1}_n^T + \mathbb{1}_n \mathbb{1}_n^T (GP)_{\mathrm{d}} - GP = I - G + \mathbb{1}_n \mathbb{1}_n^T G_{\mathrm{d}}$. Substituting this into (2.15) gives

the following reduced form.

$$\mathcal{M} = \big(G(P \circ W)\Xi - \mathbb{1}_n\mathbb{1}_n^T(G(P \circ W)\Xi)_{\mathrm{d}} \tag{2.16}$$
$$+ \beta(\mathbb{1}_n\mathbb{1}_n^T G_{\mathrm{d}} + I - G)\big)M_{\mathrm{d}}.$$

Observing the definition of the generalized inverse, G, given by Theorem 15 part (ii) and recalling that $\Xi = \mathbb{1}_n\boldsymbol{\pi}^T$, we can rewrite the first term on the right hand side of (16) as $G(P \circ W)\Xi = (I - P + \boldsymbol{t}\boldsymbol{u}^T)^{-1}(P \circ W)\mathbb{1}_n\boldsymbol{\pi}^T$. Substituting (2.20) in for the right hand side with $\boldsymbol{t} = (P \circ W)\mathbb{1}_n$ gives $G(P \circ W)\Xi = \frac{\mathbb{1}_n\boldsymbol{\pi}^T}{\boldsymbol{u}^T\mathbb{1}_n} = \frac{1}{\boldsymbol{u}^T\mathbb{1}_n}\Xi$, and so $\mathbb{1}_n\mathbb{1}_n^T(G(P \circ W)\Xi)_{\mathrm{d}} = \mathbb{1}_n\mathbb{1}_n^T(\frac{1}{\boldsymbol{u}^T\mathbb{1}_n}\Xi)_{\mathrm{d}} = \frac{1}{\boldsymbol{u}^T\mathbb{1}_n}\Xi = G(P \circ W)\Xi$. Therefore, the first two terms in (2.16) cancel giving the equality

$$\mathcal{M} = \beta(\mathbb{1}_n\mathbb{1}_n^T G_{\mathrm{d}} + I - G)M_{\mathrm{d}}. \tag{2.17}$$

We have already defined $\boldsymbol{t}$ in the generalized inverse G but not $\boldsymbol{u}$. Let $\boldsymbol{u} = \boldsymbol{\pi}$ and multiply the right hand side of (2.17) by $\boldsymbol{\pi}$ and the left hand side by $\boldsymbol{\pi}^T$. Utilizing equality (2.21) from Lemma 16 gives

$$\begin{aligned}
\boldsymbol{\pi}^T\mathcal{M}\boldsymbol{\pi} &= \boldsymbol{\pi}^T\beta(\mathbb{1}_n\mathbb{1}_n^T G_{\mathrm{d}} + I - G)M_{\mathrm{d}}\boldsymbol{\pi} \\
&= \beta(\mathbb{1}_n^T G_{\mathrm{d}} + \boldsymbol{\pi}^T - \frac{\boldsymbol{\pi}^T}{\beta})\mathbb{1}_n \\
&= \beta(\mathbb{1}_n^T G_{\mathrm{d}}\mathbb{1}_n + 1 - \frac{1}{\beta}) \\
&= \beta(\mathbf{Tr}[G] + 1) - 1.
\end{aligned}$$

Noting that the eigenvalue at 1 for an irreducible row-stochastic matrix is unique, it can be easily verified using the orthogonality property of left and right eigenvectors that the eigenvalues of G^{-1} are $\bar{\lambda}_i = (1 - \lambda_i)$ for $i \in \{2, \ldots, n\}$, where λ_i are eigenvalues of P and $\lambda_i \neq 1$. Therefore, it only remains to find $\bar{\lambda}_1$. Taking the trace of G^{-1} gives $\mathbf{Tr}[I - P + (P \circ W)\mathbb{1}_n\boldsymbol{\pi}^T] = \mathbf{Tr}[I - P] + \mathbf{Tr}[(P \circ W)\mathbb{1}_n\boldsymbol{\pi}^T] = \sum_{i=1}^{n}(1-\lambda_i)+\boldsymbol{\pi}^T(P\circ W)\mathbb{1}_n$, which implies that $\bar{\lambda}_1 = \boldsymbol{\pi}^T(P\circ W)\mathbb{1}_n = \beta$. Therefore, $\beta(\mathbf{Tr}[G] + 1) - 1 = \beta(\frac{1}{\beta} + \sum_{i=2}^{n} \frac{1}{1-\lambda_i} + 1) - 1 = \beta(1 + \sum_{i=2}^{n} \frac{1}{1-\lambda_i})$. □

2.6.2 Supplemental Material

For completeness, we include the following results which are needed in the proof of Theorem 8. We begin with some standard results on generalized inverses. For more details refer to [43, Chapter 4] or [42] .

Definition 12 (Generalized inverse)**.** *A* generalized inverse *of an $m \times n$ matrix A is defined as any $n \times m$ matrix A^- that has the property*

$$AA^-A = A.$$

It should be noted that a generalized inverse always exists, although it is not always unique. However, for non-singular matrices the generalized inverse is unique and corresponds to the usual notion of a matrix inverse. The following theorems summarize practical considerations when working with generalized inverses.

Theorem 13. *The equation $A\boldsymbol{x} = \boldsymbol{b}$ admits a solution if and only if every generalized inverse A^- satisfies*

$$AA^-\boldsymbol{b} = \boldsymbol{b}.$$

Then, we say $A\boldsymbol{x} = \boldsymbol{b}$ is consistent *and all its general solutions are given by*

$$\boldsymbol{x} = A^-\boldsymbol{b} + (A^-A - I)\boldsymbol{z},$$

where $\boldsymbol{z}$ is an arbitrary vector. Moreover, a necessary and sufficient condition for the equation $AX = C$ to be consistent is that $(I - AA^-)C = 0$, in which case the general solution is given by

$$X = A^-C + (I - A^-A)U,$$

where U is an arbitrary matrix.

The next two results come from [44, Chapter 7].

Lemma 14 (Diagonal matrix properties)**.** *For $\boldsymbol{\pi}$ with positive non-zero elements, let $\mathbb{1}_n\boldsymbol{\pi}^T = \Xi$, where $\Xi_{\mathrm{d}} = diag[\boldsymbol{\pi}]$. Also, let Λ be any diagonal matrix, X any square matrix of same dimensions as Λ, and $D = (\Xi_{\mathrm{d}})^{-1}$, then*

(i.) $(X\Lambda)_{\mathrm{d}} = (X_{\mathrm{d}})\Lambda$, *and*

(ii.) $(X\mathbb{1}_n\mathbb{1}_n^T)_{\mathrm{d}} = (X\Xi)_{\mathrm{d}}D$, *and*

(iii.) $\mathbb{1}_n \mathbb{1}_n^T \Xi_{\mathrm{d}} = \Xi$.

Theorem 15 (Generalized inverse of $I-P$)**.** *Let $P \in \mathbb{R}^{n\times n}$ be the transition matrix of a irreducible Markov Chain with stationary distribution $\boldsymbol{\pi}$. Let $\boldsymbol{u}, \boldsymbol{t} \in \mathbb{R}^n$ be any vectors such that $\boldsymbol{u}^T \mathbb{1}_n \neq 0$ and $\boldsymbol{\pi}^T \boldsymbol{t} \neq 0$, then*

(i.) $I - P + \boldsymbol{t}\boldsymbol{u}^T$ *is nonsingular, and*

(ii.) $(I - P + \boldsymbol{t}\boldsymbol{u}^T)^{-1}$ *is a generalized inverse of* $I - P$.

Lemma 16 (Properties of the generalized inverse of $I-P$)**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P, W)$ be a doubly-weighted graph with associated weight matrix W and irreducible transition matrix P with stationary distribution $\boldsymbol{\pi}$. Also let $G = (I - P + \boldsymbol{t}\boldsymbol{u}^T)^{-1}$ denote the generalized inverse of $(I - P)$, then the following relations hold.*

$$(I - P)G = I - \frac{\boldsymbol{t}\boldsymbol{\pi}^T}{\boldsymbol{\pi}^T \boldsymbol{t}}, \tag{2.18}$$

$$G(I - P) = I - \frac{\mathbb{1}_n \boldsymbol{u}^T}{\boldsymbol{u}^T \mathbb{1}_n}, \text{ and} \tag{2.19}$$

$$\frac{\mathbb{1}_n}{\boldsymbol{u}^T \mathbb{1}_n} = G\boldsymbol{t}. \tag{2.20}$$

If $\boldsymbol{t} = (P \circ W)\mathbb{1}_n$ and $\boldsymbol{u}^T = \boldsymbol{\pi}^T$ then

$$\boldsymbol{\pi}^T G = \frac{\boldsymbol{\pi}^T}{\boldsymbol{\pi}^T (P \circ W)\mathbb{1}_n} \tag{2.21}$$

Proof. First, notice that $(I - P + \boldsymbol{t}\boldsymbol{u}^T)(I - P + \boldsymbol{t}\boldsymbol{u}^T)^{-1} = I$ implies that

$$(I - P)G = I - \boldsymbol{t}\boldsymbol{u}^T G. \tag{2.22}$$

Multiplying both sides on the left by $\boldsymbol{\pi}^T$ and noting that $\boldsymbol{\pi}^T(I - P) = 0$ gives that $\boldsymbol{\pi}^T = (\boldsymbol{\pi}^T\boldsymbol{t})\boldsymbol{u}^T G$. Dividing through by $(\boldsymbol{\pi}^T\boldsymbol{t})$ gives

$$\frac{\boldsymbol{\pi}^T}{\boldsymbol{\pi}^T\boldsymbol{t}} = \boldsymbol{u}^T G, \tag{2.23}$$

and substituting (2.23) into (2.22) gives (2.18).

Following a similar procedure as before with $(I-P+\boldsymbol{t}\boldsymbol{u}^T)^{-1}(I-P+\boldsymbol{t}\boldsymbol{u}^T) = I$, where we now multiply both sides on the right by $\mathbb{1}_n$ and note that $(I-P)\mathbb{1}_n = 0$ results in (2.20), which after appropriate substitution gives (2.19).

For the proof of equality (2.21), first we check that $\boldsymbol{t} = (P\circ W)\mathbb{1}_n$ and $\boldsymbol{u}^T = \boldsymbol{\pi}^T$ satisfy the conditions of Theorem 15. The definition of W guarantees that $P \circ W$ has at least one non-zero element which implies $\boldsymbol{\pi}^T\boldsymbol{t} = \boldsymbol{\pi}^T(P \circ W)\mathbb{1}_n \neq 0$. Also, $u^T\mathbb{1}_n = \boldsymbol{\pi}^T\mathbb{1}_n = 1$. Now substituting $\boldsymbol{u}$ and $\boldsymbol{t}$ into (2.23) gives (2.21). □

Chapter 3

The Hitting Time of Multiple Random Walks

This chapter further extends notions of the hitting time described in Chapter 2 and is organized as follows. In Section 3.1 we introduce special tensor notation that is specific to this chapter only and review useful concepts of Kronecker products and Markov Chains. In Section 3.2 we provide an alternate formulation for the hitting time and weighted hitting time. In Section 3.3 we introduce the 'group' hitting time and hitting time between sets of nodes for a Markov chain, and provide a detailed characterization. In Section 3.4 we provide insight into optimal group hitting times through numerical simulation, and finally in Section 3.5 we summarize our findings.

As in the previous chapter, we work with finite irreducible time-homogeneous Markov chains (see Section 2.1 for details).

3.1 Tensor notation and the Kronecker product

In this section we define various useful concepts and notation. In particular, we introduce the notation that will be used throughout this chapter to deal with tensors, and conclude with a brief summary of the Kronecker product (also known as a tensor product) and some of its properties.

We use the notation $A = [a_{i_1 \dots i_k, j}]$ to denote the matrix generated by elements $a_{i_1 \dots i_k, j}$, where the j index denotes the j-th column of A and the rows of A are determined by cycling through index i_k, then i_{k-1}, and so forth. For example, consider $i_1, i_2, j \in \{1, \dots, n\}$, then

$$A = [a_{i_1 i_2, j}] = \begin{bmatrix} a_{11,1} & a_{11,2} & \dots & a_{11,n} \\ a_{12,1} & a_{12,2} & \dots & a_{12,n} \\ \vdots & \vdots & \dots & \vdots \\ a_{1n,1} & a_{1n,2} & \dots & \vdots \\ a_{21,1} & a_{21,2} & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \dots \\ a_{nn,1} & \dots & \dots & a_{nn,n} \end{bmatrix}.$$

For the case where $A = [a_{i,j}]$ this corresponds to the classic interpretation with element $a_{i,j}$ in the i-th row and j-th column of A. To avoid ambiguity, especially in cases when $A = [a_{i_1 \dots i_k, j}]$, at times we will refer to the (i, j) element of A by

$A(i,j)$. Unless otherwise mentioned, vectors will be denoted by bold-faced letters (i.e., $\boldsymbol{a}$). We use the notation diag$\boldsymbol{a}$ to indicate the diagonal matrix generated by vector $\boldsymbol{a}$ and vec$[A]$ to indicate the vectorization of a matrix $A \in \mathbb{R}^{n\times m}$ where

$$\text{vec}[A] = [A(1,1), \ldots, A(n,1), A(1,2), \ldots, A(n,2), \ldots, A(m,1), \ldots, A(n,m)]^T.$$

In other words, even if we define A as $A = [a_{i_1\ldots i_k,j}]$, the vector vec$[A]$ = vec$[[a_{i_1\ldots i_k,j}]]$ is simply a stacking of the columns of A. We also define the special matrix $[\mathcal{I}^{h_1\ldots h_n,k}_{i_1\ldots i_n,j}]$ as the matrix whose entries are all zero except for a single entry at $(h_1 \ldots h_n, k)$ which has a value of 1, where $h_1, \ldots, h_n, k$ can only take values within the range of values that $i_1, \ldots, i_n, j$ take. With this matrix definition, it is easy to verify for $A = [a_{i_1\ldots i_n,j}]$ that $a_{h_1\ldots h_n,k} = \text{vec}[[\mathcal{I}^{h_1\ldots h_n,k}_{i_1\ldots i_n,j}]]^T \text{vec}[A]$. This enables us to go back and forth between the vectorized notation to the individual matrix elements. We denote $I_n \in \mathbb{R}^{n\times n}$ as the identity matrix of size n, $\mathbb{1}_n$ as the vector of ones of size n and $\mathbb{0}_{n\times n}$ as the matrix zeros of size $n \times n$. We define a generalized Kronecker delta function $\delta_{i_1 i_2\ldots i_n,j}$, by

$$\delta_{i_1\ldots i_n,j} = \begin{cases} 1, & \text{if } i_k = j \text{ for any } k \in \{1,\ldots,n\}, \\ 0, & \text{otherwise .} \end{cases}$$

Then, with a slight abuse of notation, $A_d = [\delta_{i_1 i_2\cdots n,j} a_{i_1\ldots i_k,j}]$ represents the "diagonal" matrix generated by the elements of A. In reality, only when $A = [a_{i,j}]$ is the matrix truly diagonal. Finally, since the subscript operator is already in use,

we use the superscript operation in parenthesis to delineate between two variables with the same name. For example we write $A^{(1)}$ and $A^{(2)}$ to distinguish that the matrices are different. A superscript without parenthesis denotes a matrix raised to that power (i.e., $A^2 = AA$ and $(A^{(2)})^2 = A^{(2)}A^{(2)}$).

We are now ready to review some useful facts about Kronecker products. The Kronecker product, represented by the symbol $\otimes$, of two matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{q\times r}$ is a $nq \times mr$ matrix given by

$$A \otimes B = \begin{bmatrix} a_{1,1}B & \dots & a_{1,m}B \\ \vdots & \ddots & \vdots \\ a_{n,1}B & \ddots & a_{n,m}B \end{bmatrix}.$$

To build some intuition, notice for $A \in \mathbb{R}^{n\times n}$, that $I_n \otimes A$ is the block diagonal matrix with n copies of A on the diagonal:

$$I_n \otimes A = \begin{bmatrix} A & \mathbb{0}_{n\times n} & \dots & \mathbb{0}_{n\times n} \\ \mathbb{0}_{n\times n} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0_{n\times n} \\ \mathbb{0}_{n\times n} & \dots & \mathbb{0}_{n\times n} & A \end{bmatrix}. \tag{3.1}$$

This implies for $A = I_n$, that $I_n \otimes A = I_n \otimes I_n = I_{n^2}$. The Kronecker product is bilinear and has many useful properties, two of which are summarized in the following lemma; see [41] for more information.

Lemma 17. *Given the matrices A, B, C and D, the following relations hold for the Kronecker product.*

(i) $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$,

(ii) $(B^T \otimes A)\,\mathrm{vec}[C] = \mathrm{vec}[ACB]$,

where it is assumed that the matrices are of appropriate dimension when matrix multiplication or addition occurs. In addition, for matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times m}$ with respective eigenvalues λ_i^A, $i \in \{1, \ldots, n\}$ and λ_j^B, $j \in \{1, \ldots, m\}$,

(iii) the eigenvalues of $A \otimes B$ are $\lambda_i^A \lambda_j^B$ for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$.

3.2 The hitting time of a Markov chain

We begin by recalling some properties of the single random walker hitting time from Chapter 2, and then provide a new formulation for determining this quantity. We show that this alternate formulation gives rise to new results for the pairwise hitting time between two specified nodes, and, in the subsequent section, utilize it to extend the notion of the hitting time and pairwise hitting time to the multiple random-walker case.

3.2.1 The hitting time of a Markov chain

Consider a connected undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P)$ with node set $\mathcal{V} := \{1, \dots, n\}$, edge set $E \subset \mathcal{V} \times \mathcal{V}$, and weight matrix $P = [p_{i,j}]$ with the property that $p_{i,j} \geq 0$ if $(i,j) \in \mathcal{E}$ and $p_{i,j} = 0$ otherwise. We interpret the weight of edge (i,j) as the probability of moving along that edge. Therefore, P is a transition matrix of a Markov chain, where the element $p_{i,j}$ in the matrix represents the probability with which the random walk visits node j from node i.

As done previously, let $X_t \in \{1, \dots, n\}$ denote the location of a random walker at time $t \in \{0, 1, 2, \dots\}$. For any two nodes $i, j \in \{1, \dots, n\}$, the *first passage time from i to j*, denoted by $T_{i,j}$, is the first time that the random walker starting at node i at time 0 reaches node j, that is,

$$T_{i,j} = \min\{t \geq 1 \mid X_t = j \text{ given that } X_0 = i\}.$$

The mean first passage time from i to j is then given by $m_{i,j} = \mathbb{E}[T_{i,j}]$. Denoting the *mean first passage time matrix* M as the matrix whose i,jth entries are given by $m_{i,j}$, recall that the hitting time, $H(P)$, can be written as $H(P) = \boldsymbol{\pi}^T M \boldsymbol{\pi}$. Given the definition of the hitting time, one quickly sees that it can also be determined using the matrix M as follows,

$$\boldsymbol{\pi}^T M \boldsymbol{\pi} = (\boldsymbol{\pi} \otimes \boldsymbol{\pi})^T \operatorname{vec}[M] = \sum_{i=1}^{n} \boldsymbol{\pi}_i \sum_{j=1}^{n} \boldsymbol{\pi}_j m_{i,j} = H, \tag{3.2}$$

where the relation $\boldsymbol{\pi}^T M \boldsymbol{\pi} = (\boldsymbol{\pi} \otimes \boldsymbol{\pi})^T \operatorname{vec}[M]$ follows from identity (ii) in Lemma 17. From Chapter 2 we saw that $m_{i,j}$ is described by the following recursive formula.

$$m_{i,j} = p_{i,j} + \sum_{k=1,k\neq j}^{n} p_{i,k}(m_{k,j}+1) = 1 + \sum_{k=1,k\neq j}^{n} p_{i,k} m_{k,j}.$$

The above formula can be expressed in various vectorized forms. The classical form already seen utilizes the matrix representation of M and is given by

$$(I-P)M = \mathbb{1}_n \mathbb{1}_n^T - PM_{\mathrm{d}}, \tag{3.3}$$

where we recall that $M_{\mathrm{d}} = \operatorname{diag}\{1/\boldsymbol{\pi}_1, \ldots, 1/\boldsymbol{\pi}_n\}$ and the value for M_d is determined by pre-multiplying (3.3) by $\boldsymbol{\pi}^T$. For reasons which will become clear later, we vectorize the matrix M to generate a different representation of (3.3). Applying property (ii) of Lemma 17 to (3.3) gives

$$(I_n \otimes (I_n - P)) \operatorname{vec}[M] = \mathbb{1}_{n^2} - (I_n \otimes P) \operatorname{vec}[M_{\mathrm{d}}]. \tag{3.4}$$

Similar to before, M_d can be determined from (3.4) with appropriate vector pre-multiplication.

We know from Theorem 1 that the hitting time can be written as a function of the eigenvalues of P as

$$H(P) = 1 + \sum_{i=2}^{n} \frac{1}{1-\lambda_i}.$$

The proof for Theorem 1 relies on the fact that M_d is known, however, as will be seen next, there exists an equivalent expression for $H(P)$ which requires no such

knowledge. Before presenting the alternate formulation for $H(P)$, we introduce the following useful result.

Lemma 18. *(Eigenvalue shifting for stochastic matrices): Let $P \in \mathbb{R}^{n\times n}$ be an irreducible row stochastic matrix, and let E be any diagonal matrix with diagonal elements $E_{ii} \in \{0,1\}$, with at least one diagonal element which is zero. Then the eigenvalues λ_i of PE satisfy $|\lambda_i| < 1$ for all $i \in \{1,\ldots,n\}$.*

Proof. Consider the case when the kth diagonal element of E is equal to zero, $E_{kk} = 0$ and all other diagonal elements are equal to 1, $E_{ii} = 1$ for $i \neq k$. This has the affect of making the kth column of P zero, $p_{ik} = 0$; the physical interpretation is that no node can move to node k. Thus the matrix $A = PE$ is reducible and there exists a permutation matrix $S \in \mathbb{R}^{n\times n}$ such that SAS^T is the block upper triangular matrix

$$SAS^T = \begin{bmatrix} B^{(1)}_{r\times r} & C_{r\times(n-r)} \\ \mathbb{0}_{(n-r)\times r} & B^{(2)}_{(n-r)\times(n-r)} \end{bmatrix}, \tag{3.5}$$

where each $B^{(i)}$ is square. Clearly, SAS^T remains sub-stochastic under permutation and the eigenvalues of the matrix A correspond to the eigenvalues of each block $B^{(i)}$. Consider the trivial permutation which moves the column k to the first column. Then, $B^{(1)} = [0] \in \mathbb{R}^{1\times 1}$ and $B^{(2)} \in \mathbb{R}^{n-1\times n-1}$. Clearly $B^{(2)}$ is sub-stochastic since P is irreducible and there exists a path from every node to

every other node. Now, if $B^{(2)}$ is also irreducible then $\lambda_{\max}[B^{(2)}] < 1$, [60]. If it is not, then there exists another permutation matrix,$\bar{S} \in \mathbb{R}^{n-1\times n-1}$, such that $\bar{S}B^{(2)}\bar{S}^T$ is upper block triangular similar to (3.5). Due to the irreducibility of P each of the diagonal blocks of $\bar{S}B^{(2)}\bar{S}^T$ are sub-stochastic. Therefore, if they are irreducible then $\lambda_{\max} < 1$ for each block. If not, then we repeat the previous argument until we are left with diagonal blocks which are irreducible or zero. The case when more than one column is zero follows from noting that if $a_{i,j} \leq b_{i,j}$ for each i and j then $\rho[A] \leq \rho[B]$, [60, Chapter 7.10]. □

We are almost ready to present our alternate representation of the hitting time, but first we must introduce the following equality: notice that $\text{vec}[M_d] = E\,\text{vec}[M]$, when E is defined by $E = \text{diag}[\delta_{i,j}]$. Using this interpretation of $\text{vec}[M_d]$ we are ready to state our result.

Theorem 19. *(Hitting times of an irreducible Markov chain): Consider a Markov chain with an irreducible transition matrix $P \in \mathbb{R}^{n\times n}$, then the following properties hold:*

(i) the hitting time of the Markov chain is given by

$$H(P) = (\boldsymbol{\pi} \otimes \boldsymbol{\pi})^T \,\text{vec}[M], \quad \textit{where}$$

$$\text{vec}[M] = (I_{n^2} - (I_n \otimes P)(I_{n^2} - E))^{-1}\mathbb{1}_{n^2}, \quad \textit{and}$$

(ii) the pairwise hitting time between nodes h and k, denoted $m_{h,k}$, of the Markov chain is given by

$$m_{h,k} = \text{vec}[[\mathcal{I}_{i,j}^{h,k}]]^T \text{vec}[M].$$

Proof. First notice that by rearranging (3.4) and substituting $E\,\text{vec}[M]$ for $\text{vec}[M_d]$ gives that

$$(I_{n^2} - (I_n \otimes P)(I_{n^2} - E))\,\text{vec}[M] = \mathbb{1}_{n^2}. \tag{3.6}$$

From (3.2) we know that $H = (\boldsymbol{\pi} \otimes \boldsymbol{\pi})\,\text{vec}[M]$, therefore it only remains to show that $(I_{n^2} - I_n \otimes P(I_{n^2} - E))$ is in fact invertible. First, recall from (3.1) that $I_n \otimes P$ results in the block diagonal matrix, whose diagonal blocks consist of copies of P. Second, notice that $(I_{n^2} - E)$ is simply the identity matrix with some diagonal entries set to zero. It can be easily verified that $(I_n \otimes P)(I_{n^2} - E)$ results in the block diagonal matrix where each block contains the matrix P with one column

set to zero. For example, for $P \in \mathbb{R}^{3\times 3}$ we have that

$$(I_n \otimes P)(I_{n^2} - E) = \left[\begin{array}{ccc|ccc|ccc} 0 & p_{12} & p_{13} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & p_{22} & p_{23} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & p_{32} & p_{33} & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & p_{11} & 0 & p_{13} & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{21} & 0 & p_{23} & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{31} & 0 & p_{33} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & p_{11} & p_{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_{21} & p_{22} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_{31} & p_{32} & 0 \end{array}\right].$$

Notice that each diagonal block will have at least one column set to zero. Hence, using Lemma 18, we have that maximum eigenvalue of each block is strictly less than one in magnitude, and thus $\left|\lambda_{\max}[(I_n \otimes P)(I_{n^2} - E)]\right| < 1$. Let λ_i denote the eigenvalues of $(I_n \otimes P)(I_{n^2} - E)$, then since the eigenvalues of $(I_{n^2} - (I_n \otimes P)(I_{n^2} - E))$ are simply $1 - \lambda_i$ and $|\lambda_i| < 1$ for all $i \in \{1, \ldots, n^2\}$, this implies the matrix $(I_{n^2} - (I_n \otimes P)(I_{n^2} - E))$ is invertible and $\text{vec}[M]$ is the unique solution to (3.6).

□

Remark 20. *Notice that when the transition matrix P is reducible, the hitting time $H(P)$ is infinite (i.e., due to $I_{n^2} - (I_n \otimes P)(I_{n^2} - E)$ being singular). In other words, the hitting time gives a notion of the connectivity of the graph.*

3.2.2 Hitting time for doubly-weighted graphs

Leveraging results from Chapter 2 , Theorem 19 can be easily extended to account for "travel distances" between nodes. Define the doubly-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P, W)$, where $W = [\omega_{i,j}]$ is the weight matrix with the properties that: if $(i,i) \in \mathcal{E}$, then $\omega_{i,i} \geq 0$; if $(i,j) \in \mathcal{E}$, $i \neq j$, then $\omega_{i,j} > 0$; and if $(i,j) \notin \mathcal{E}$, then $\omega_{i,j} = 0$. Let $P = [p_{i,j}]$ be the transition matrix associated with $\mathcal{G}$ with the property that $p_{i,j} \geq 0$ if $(i,j) \in \mathcal{E}$ and $p_{i,j} = 0$ otherwise.

Recall that the first passage time is now denoted by

$$T_{i,j}(W) = \min\Big\{ \sum_{n=0}^{k-1} w_{X_n, X_{n+1}}, \text{ for } k \geq 1 \mid X_k = j \text{ given that } X_0 = i \Big\}.$$

Letting $m_{i,j}(W) = \mathbb{E}[T_{i,j}]$ denote the mean first passage time from i to j, or in matrix notation $\mathcal{M} = [m_{i,j}(W)]$, the following matrix relationship can be determined (see Section 2.3.1 equation (2.6)).

$$(I - P)\mathcal{M} = (P \circ W)\mathbb{1}_n \mathbb{1}_n^T - P\mathcal{M}_{\mathrm{d}}, \tag{3.7}$$

where $P \circ W$ denotes the Hadamard product between P and W. Similar to before, applying property (ii) of Lemma 17 to (3.7) gives

$$(I_n \otimes (I_n - P))\,\mathrm{vec}[\mathcal{M}] = \mathrm{vec}[(P \circ W)\mathbb{1}_n^T \mathbb{1}_n] - (I_n \otimes P)\,\mathrm{vec}[\mathcal{M}_{\mathrm{d}}].$$

Given the above expression, the following Lemma follows from Theorem 19.

Lemma 21. *(hitting time and pairwise hitting times of a doubly-weighted graph): Given the doubly-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, P, W)$, the following properties hold:*

(i) the weighted hitting time $H_W(P)$ of the Markov chain is given by

$$H_W(P) = (\boldsymbol{\pi} \otimes \boldsymbol{\pi})^T \operatorname{vec}[\mathcal{M}], \quad \textit{where}$$

$$\operatorname{vec}[\mathcal{M}] = (I_{n^2} - (I_n \otimes P)(I_{n^2} - E))^{-1} \operatorname{vec}[(P \circ W)\mathbb{1}_n^T \mathbb{1}_n], \ \textit{and}$$

(ii) the pairwise hitting time between nodes h and k, denoted $m_{h,k}(W)$, of the Markov chain is given by

$$m_{h,k}(W) = \operatorname{vec}[[\mathcal{I}_{i,j}^{h,k}]]^T \operatorname{vec}[\mathcal{M}].$$

Now that the single random walker case has been explored in detail, we explore the case of multiple random-walkers in the following section.

3.3 Group hitting time of multiple Markov chains

In the following sections, we will expand on the single agent hitting time to the N-agent group hitting time. To build intuition, we initially assume that every agent can reach all nodes in the graph, and then move to the case where each agent only needs to have access to a subset of nodes in the graph.

3.3.1 Random-walkers covering the full graph

Consider $h \in \{1, \dots, N\}$ connected undirected weighted graphs $\mathcal{G}^{(h)} = (\mathcal{V}, E^{(h)}, P^{(h)})$ with same node sets $\mathcal{V} := \{1, \dots, n\}$ and different edge sets $E^{(h)} \subset \mathcal{V} \times \mathcal{V}$ with corresponding transition matrices $P^{(h)} = [p_{i,j}^{(h)}]$ for $h \in \{1, \dots, N\}$ satisfying the property $p_{i,j}^{(h)} \geq 0$ if $(i,j) \in E^{(h)}$ and $p_{i,j}^{(h)} = 0$ otherwise. As before, each matrix $P^{(h)}$ describes a Markov chain on the graph.

Let $X_t^{(1)}, X_t^{(2)}, \dots, X_t^{(N)} \in \{1, \dots, n\}$ denote the location of N random walkers at time $t \in \{0, 1, 2, \dots\}$. For any $N+1$ nodes $i_1, \dots, i_N, j \in \{1, \dots, n\}$, the *first passage time from any i_h, $h \in \{1, \dots, N\}$ to j*, denoted by $T_{i_1 \dots i_N, j}$, is the first time that any random walker reaches node j, when starting from nodes i_h, $h \in \{1, \dots, N\}$. More formally,

$$T_{i_1 \dots i_N, j} = \min\{t \geq 1 \mid \text{For } h \in \{1, \dots, N\}, \text{ any } X_t^{(h)} = j \text{ given that } X_0^{(h)} = i_h\}. \tag{3.8}$$

With this definition, we are ready to state our first result for the N-random-walker system.

Lemma 22. *(Recursive formulation of first passage time for multiple random-walkers): Let $m_{i_1 \dots i_N, j} = \mathbb{E}[T_{i_1 \dots i_N, j}]$ denote the first passage time from any i_h, $h \in \{1, \dots, N\}$ to j. Also, let $P^{(h)}$ be the transition matrix associated with $\mathcal{G}^{(h)}$,*

then

$$m_{i_1\dots i_N,j} = 1 + \sum_{k_1,\dots,k_N \neq j} m_{k_1\dots k_N,j} p^{(1)}_{i_1,k_1} \dots p^{(N)}_{i_N,k_N},$$

or, in matrix notation,

$$M = \mathbb{1}_{n^N}\mathbb{1}_n^T + (P^{(1)} \otimes \dots \otimes P^{(N)})M - (P^{(1)} \otimes \dots \otimes P^{(N)})M_d, \tag{3.9}$$

where $M = [m_{i_1\dots i_N,j}]$ *and* $M_d = [\delta_{i_1\dots i_N,j} m_{i_1\dots i_N,j}]$.

Proof. For clarity, we first study the 2-agent case and then generalize. By definition, the 2-agent first passage time satisfies the recursive formula

$$T_{i_1 i_2,j} = \begin{cases} 1, & \text{with probability } p^{(1)}_{i_1,j} + p^{(2)}_{i_2,j} - p^{(1)}_{i_1,j} p^{(2)}_{i_2,j}, \\ T_{k_1 k_2,j} + 1, & \text{with probability } p^{(1)}_{i_1,k_1} p^{(2)}_{i_2,k_2} \text{ such that } k_1, k_2 \neq j. \end{cases}$$

Taking the expectation we have that

$$\begin{aligned} \mathbb{E}[T_{i_1 i_2,j}] &= p^{(1)}_{i_1,j} + p^{(2)}_{i_2,j} - p^{(1)}_{i_1,j} p^{(2)}_{i_2,j} + \sum_{k_1,k_2 \neq j} (\mathbb{E}[T_{k_1 k_2,j}] + 1) p^{(1)}_{i_1,k_1} p^{(2)}_{i_2,k_2} \\ &= p^{(1)}_{i_1,j} + p^{(2)}_{i_2,j} - p^{(1)}_{i_1,j} p^{(2)}_{i_2,j} + \sum_{k_1,k_2 \neq j} m_{k_1 k_2,j} p^{(1)}_{i_1,k_1} p^{(2)}_{i_2,k_2} + \sum_{k_1,k_2 \neq j} p^{(1)}_{i_1,k_1} p^{(2)}_{i_2,k_2}. \end{aligned} \tag{3.10}$$

Utilizing the row-stochastic property of $P^{(1)}$ and $P^{(2)}$ expand $(-p^{(1)}_{i_1,j}p^{(2)}_{i_2,j})$ above to get

$$\begin{aligned}
-p^{(1)}_{i_1,j}p^{(2)}_{i_2,j} &= -(1-\sum_{k_1\neq j} p^{(1)}_{i_1,k_1})(1-\sum_{k_2\neq j} p^{(2)}_{i_2,k_2}) \\
&= -1+\sum_{k_1\neq j} p^{(1)}_{i_1,k_1}+\sum_{k_2\neq j} p^{(2)}_{i_2,k_2}-(\sum_{k_1\neq j} p^{(1)}_{i_1,k_1})(\sum_{k_2\neq j} p^{(2)}_{i_2,k_2}) \\
&= -1+\sum_{k_1\neq j} p^{(1)}_{i_1,k_1}+\sum_{k_2\neq j} p^{(2)}_{i_2,k_2}-\sum_{k_1,k_2\neq j} p^{(1)}_{i_1,k_1}p^{(2)}_{i_2,k_2},
\end{aligned}$$

substituting this back into (3.10) gives the result

$$m_{i_1i_2,j} = 1+\sum_{k_1,k_2\neq j} m_{k_1k_2,j}p^{(1)}_{i_1,k_1}p^{(2)}_{i_2,k_2},$$

or, in matrix notation,

$$M = \mathbb{1}_{n^2}\mathbb{1}_2^T + (P^{(1)}\otimes P^{(2)})M - (P^{(1)}\otimes P^{(2)})M_d,$$

where $M=[m_{i_1i_2,j}]$ and $M_d=[\delta_{i_1i_2,j}m_{i_1i_2,j}]$.

Similar to the 2-agent case, the N-agent first passage time satisfies the recursive formula

$$T_{i_1\ldots i_N,j} = \begin{cases} 1, & \text{with probability } 1-(1-p^{(1)}_{i_1,j})(1-p^{(2)}_{i_2,j})\ldots(1-p^{(N)}_{i_N,j}), \\ T_{k_1\ldots k_N,j}+1, & \text{with probability } p^{(1)}_{i_1,k_1}p^{(2)}_{i_2,k_2}\ldots p^{(N)}_{i_N,k_N} \text{ such that } k_1,\ldots,k_N\neq j. \end{cases}$$

Similar to before, we take expectations and utilize the row-stochastic properties of each $P^{(i)}$ to reach the result. □

Given the formulation for the N-agent first passage time matrix we can define a quantity similar to the mean first passage time given by (3.2). In order to do this, we first need to define the frequency of being at any given node in the graph. This quantity should take into account the probability of being at one node instead of another in the limit of the random walks. Since the random walks are evolving in parallel, the relative frequency of being at a specific node is simply the average of the N random walkers visit frequency at that node. More explicitly, $\boldsymbol{\pi}_{\text{ave}} = \sum_{h=1}^{N}(\boldsymbol{\pi}^{(h)})/N$. Then, similar to the single agent case, the *N-agent mean first passage time from start nodes i_h, $h \in \{1, \ldots, N\}$*, denoted $\boldsymbol{h}_{i_1 \ldots i_N}$, is given by

$$\boldsymbol{h}_{i_1 \ldots i_N} = \sum_{j=1}^{n} m_{i_1 \ldots i_N, j} \boldsymbol{\pi}_{\text{ave},j}.$$

Therefore, the average time to go from any set of N nodes to a single node in a graph is given by

$$\begin{aligned} H_N &= \sum_{i_1=1}^{n} \boldsymbol{\pi}_{i_1}^{(1)} \cdots \sum_{i_N=1}^{n} \boldsymbol{\pi}_{i_N}^{(N)} \sum_{j=1}^{n} m_{i_1 \ldots i_N, j} \boldsymbol{\pi}_{\text{ave},j} \\ &= (\boldsymbol{\pi}^{(1)} \otimes \cdots \otimes \boldsymbol{\pi}^{(N)}) M \boldsymbol{\pi}_{\text{ave}} \\ &= (\boldsymbol{\pi}_{\text{ave}} \otimes \boldsymbol{\pi}^{(1)} \otimes \cdots \otimes \boldsymbol{\pi}^{(N)}) \operatorname{vec}[M], \end{aligned}$$

where we denote H_N as being the N-agent *group* hitting time. It is clear the group hitting time can be written as the function $P^{(1)} \times \cdots \times P^{(N)} \mapsto H_N(P^{(1)}, \ldots, P^{(N)})$, but to ease notation we simply write H_N.

Given the definition of H_N we are now ready to state our next result.

Theorem 23. *(Group hitting time for irreducible Markov chains): Consider N multiple Markov chains, each with an irreducible transition matrix $P^{(h)} \in \mathbb{R}^{n\times n}$. Let $\boldsymbol{\pi}_{\text{ave}} = (\sum_{h=1}^{N} \boldsymbol{\pi}^{(h)})/N$, and let $E \in \mathbb{R}^{n^{N+1}\times n^{N+1}}$ be the diagonal matrix which satisfies the equality $E\,\text{vec}[M] = \text{vec}[M_d]$. Then the following properties hold:*

(i) the group hitting time of the Markov chain is given by

$$H_N = (\boldsymbol{\pi}_{\text{ave}} \otimes \boldsymbol{\pi}^{(1)} \otimes \cdots \otimes \boldsymbol{\pi}^{(N)})^T \,\text{vec}[M], \quad \textit{where}$$
$$\text{vec}[M] = (I_{n^{N+1}} - (I_n \otimes P^{(1)} \otimes \cdots \otimes P^{(N)})(I_{n^{N+1}} - E))^{-1}\mathbb{1}_{n^{N+1}}, \textit{ and} \tag{3.11}$$

(ii) the hitting time between nodes $h_1, \ldots, h_N$ and k, denoted $m_{h_1\ldots h_N,k}$, of the Markov chain is given by

$$m_{h_1\ldots h_N,k} = \text{vec}[[\mathcal{I}^{h_1\ldots h_N,k}_{i_1\ldots i_N,j}]]^T \,\text{vec}[M].$$

Proof. Letting $P = P^{(1)} \otimes \cdots \otimes P^{(N)}$, notice that (3.9) can be written in the vectorized form

$$\text{vec}[M] = \mathbb{1}_{n^{N+1}} + (I_n \otimes P)\,\text{vec}[M] - (I_n \otimes P)\,\text{vec}[M_d].$$

Rearranging terms and substituting $E\,\text{vec}[M]$ for $\text{vec}[M_d]$ gives

$$(I_{n^{N+1}} - (I_n \otimes P)(I_{n^{N+1}} - E))\,\text{vec}[M] = \mathbb{1}_{n^{N+1}}. \tag{3.12}$$

To ease the complexity of our notation, we move forward by looking at the 2-agent case and then generalizing from there. For the two agent case with $P = P^{(1)} \otimes P^{(2)}$ the system (3.12) becomes

$$(I_{n^3} - (I_n \otimes P)(I_{n^3} - E))\,\mathrm{vec}[M] = \mathbb{1}_{n^3},$$

indicating a unique solution exists if $(I_{n^3} - (I_n \otimes P)(I_{n^3} - E))$ is invertible. Let the eigenvalues $\lambda_i^{(1)}$ and $\lambda_i^{(2)}$ be associated with transition matrices $P^{(1)}$ and $P^{(2)}$, respectively. Then, from property (iii) of Lemma 17 the eigenvalues of $P^{(1)} \otimes P^{(2)}$ are $\lambda_j^{(1)}\lambda_k^{(2)}$ for $j, k \in \{1, \ldots, n\}$. This means, when each $P^{(i)}$ is periodic, the Kronecker product can results in a Markov chain which has multiple eigenvalues at 1, making this chain reducible. Therefore, we must show that irreducible blocks can be constructed that allow the application of Lemma 18 as before. As before we require that $(I_n \otimes P)(I_{n^3} - E)$ has $\lambda_{\max} < 1$. Recall that $I_n \otimes P$ is a block diagonal matrix consisting of copies of P. In each block, different columns of P are set to zero when multiplied by $(I_{n^3} - E)$, however, unlike the single agent case (Theorem 19), now multiple columns are set to zero in each block by definition of E. Therefore, we need only show that in every block, each irreducible component of P has at least one column set to zero when multiplied by $(I_{n^3} - E)$. Thus, we first examine the structure of P. Since $P = P^{(1)} \otimes P^{(2)}$ is reducible we can apply a series of permutation matrices $S^{(i)}$ for $i \in \{1, \ldots, m\}$ to P ([60, Chapter 8.3])

such that

$$\bar{P} = \begin{bmatrix} A^{(1)} & * & \dots & * \\ \mathbb{0} & A^{(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ \mathbb{0} & \dots & \mathbb{0} & A^{(k)} \end{bmatrix}. \tag{3.13}$$

Where $\bar{P} = SPS^T$ with permutations $S = (S^{(m)}) \dots (S^{(1)})$, and each $A^{(i)}$ is irreducible. Since P is row-stochastic then each irreducible component is either sub-stochastic ($\rho[A^{(i)}] < 1$) or stochastic ($\rho[A^{(i)}] = 1$). Since we need only worry about the stochastic $A^{(i)}$, let's denote $B \in \mathbb{R}^{r \times r}$ as a irreducible stochastic matrix $A^{(i)} \in \mathbb{R}^{r \times r}$ in (3.13). For B irreducible, $\bar{P}$ has the form

$$\bar{P} = \begin{bmatrix} A^{(1)} & * & \dots & \dots & \dots & * \\ \mathbb{0} & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & B & \mathbb{0} & \dots & \mathbb{0} \\ \vdots & \ddots & \ddots & A^{(l)} & \dots & * \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \mathbb{0} & \dots & \dots & \dots & \mathbb{0} & A^{(k)} \end{bmatrix}.$$

Notice from the definition of vec$[M]$ and the block diagonal structure of $(I_n \otimes P)$, that the j-th block in $(I_n \otimes P)$ corresponds to mean first passage times, $m_{i_1 i_2, j}$, to node j. Since a permutation matrix simply acts as a relabeling of elements, assume without a loss of generality, that the elements associated with B vary from

$i_1, i_2 \in \{1, \ldots, r\}$. Therefore, the equations associated with B have the form

$$\begin{aligned} m_{i_1 i_2, j} = 1 &+ \sum_{k_1, k_2} m_{k_1 k_2, j} p^{(1)}_{i_1, k_1} p^{(2)}_{i_2, k_2} - \sum_{\substack{k_1 \\ k_2 \neq j}} m_{j k_2, j} p^{(1)}_{i_1, j} p^{(2)}_{i_2, k_2} \\ &- \sum_{\substack{k_2 \\ k_1 \neq j}} m_{k_1 j, j} p^{(1)}_{i_1, k_1} p^{(2)}_{i_2, j} - m_{jj, j} p^{(1)}_{i_1, j} p^{(2)}_{i_2, j} \\ = 1 &+ \sum_{k_1, k_2 \neq j} m_{k_1 k_2, j} p^{(1)}_{i_1, k_1} p^{(2)}_{i_2, k_2}, \end{aligned}$$

where the subtracted terms in the expression above are associated with entries of E (i.e., to the columns of P that are set to zero). If all subtracted terms are zero for each $m_{i_1 i_2, j}$, $i_1, i_2 \in \{1, \ldots, r\}$, this implies that there exists no $i_1, i_2 \in \{1, \ldots, r\}$ such that $p^{(1)}_{i_1 j} > 0$ or $p^{(2)}_{i_2 j} > 0$; that can only be true if there exists no path to node j from any node i_1 or i_2, which is impossible by definition of each $P^{(i)}$. Therefore, for each irreducible row-stochastic component of $\bar{P}$, there is at least one non-zero element E such that a column of that component is set to zero, allowing us to apply Lemma 18.

Now, proceeding to the N-agent case, similar to the 2 agent case, we require that a unique solution exists for the N agent case if $(I_{n^{N+1}} - (I_n \otimes P)(I_{n^{N+1}} - E))$ from (3.12) is invertible. For this to hold true it must be that $(I_n \otimes P)(I_{n^{N+1}} - E)$ has $|\lambda_{\max}| < 1$. Similar to before, given that $I_n \otimes P$ and hence $(I_n \otimes P)(I_{n^{N+1}} - E)$ is a block diagonal matrix, we need only show that each block has $|\lambda_{\max}| < 1$. The proof continues a parallel line of argument to the 2-agent case shown in

Theorem 23. More explicitly, each P can be deconstructed into block upper triangular matrix composed of square matrices along the diagonal, each of which is irreducible. Since P is row-stochastic, then each irreducible block is either row-stochastic or sub-stochastic. If the irreducible block is row-stochastic then a column of that block is necessarily set to zero due to the connectivity of the graph, and hence by Lemma 18 has $|\lambda_{\max}| < 1$. If the irreducible block is not row-stochastic then it is necessarily sub-stochastic, and hence $|\lambda_{\max}| < 1$. □

Given this representation of the N-agent group hitting time, a natural question is whether one can determine a simplified expression for this quantity which is a function of the eigenvalues of $P^{(h)}$, similar to the expression in Theorem 1. As mentioned earlier, proof of that theorem relies on the ability to extort knowledge of M_d. Consider for example the 2-agent case; if we try to find the entries of M_d in a similar fashion to the single agent case by pre-multiplying (3.9) with $\boldsymbol{\pi}^{(1)} \otimes \boldsymbol{\pi}^{(2)}$ we have that

$$\mathbb{1}_n^T = (\boldsymbol{\pi}^{(1)} \otimes \boldsymbol{\pi}^{(2)})^T M_d.$$

This is a system of n equations and $2n-1$ unknowns, and thus the solution of M_d is under determined. Therefore, even though one may be able to express the group hitting time as a function of the eigenvalues, it is currently not well understood how.

3.3.2 Random-walkers covering subgraphs

The group hitting time as stated thus far is of interest in its own right. However, in many applications it is often desirable to have a notion of the same quantity when multiple agents don't have access to the entire graph. In the following section we tackle this problem by utilizing reducible graphs. We will leverage the framework of the N-agent group hitting time for irreducible Markov chains in order to generalize our results to reducible Markov chains.

Consider $h \in \{1, \dots, N\}$ undirected irreducible weighted graphs $\mathcal{G}^{(h)} = (\mathcal{V}^{(h)}, E^{(h)}, P^{(h)})$ with node sets $\mathcal{V}^{(h)} \subset \{1, \dots, n\}$ such that $\cup_{h=1}^{N} \mathcal{V}^{(h)} = \{1, \dots, n\}$. The edge sets satisfy $E^{(h)} \subset \mathcal{V}^{(h)} \times \mathcal{V}^{(h)}$ and have corresponding weight matrices $P^{(h)} = [p_{i,j}^{(h)}]$ for $h \in \{1, \dots, N\}$ with the property $p_{i,j}^{(h)} \geq 0$ if $(i, j) \in E^{(h)}$ and $p_{i,j}^{(h)} = 0$ otherwise. As before, each matrix $P^{(h)}$ describes a Markov chain on the graph.

Let $X_t^{(h)} \in \mathcal{V}^{(h)}$ denote the location of N random walkers at time $t \in \{0, 1, 2, \dots\}$. For any $N+1$ nodes $i_h \in \mathcal{V}^{(h)}$ and $j \in \{1, \dots, n\}$, the *first passage time from any* i_h, $h \in \{1, \dots, N\}$ *to* j, denoted by $T_{i_1 \dots i_N, j}$, is the first time that any random walker reaches node j, when starting from nodes i_h, $h \in \{1, \dots, N\}$ and is given by

$$T_{i_1 \dots i_N, j} = \min\{t \geq 1 \mid \text{For } h \in \{1, \dots, N\}, \text{ any } X_t^{(h)} = j \text{ given that } X_0^{(h)} = i_h\}.$$

Similar to case where each agent has access to the entire graph, we have the following Lemma, whose proof is equivalent to that of Lemma 22.

Lemma 24. *(Recursive formulation of first passage time for multiple random-walkers over subgraphs): Consider the graphs* $\mathcal{G}^{(h)} = (\mathcal{V}^{(h)}, E^{(h)}, P^{(h)})$ *satisfying the property* $\cup_{h=1}^{N} \mathcal{V}^{(h)} = \{1, \ldots, n\}$ *and let* $P^{(h)}$ *be the transition matrix associated with* $\mathcal{G}^{(h)}$*. Also, let* $|V^{(h)}|$ *denote the cardinality of each node set and let* $m_{i_1 i \ldots i_N, j} = \mathbb{E}[T_{i_1 \ldots i_N, j}]$ *denote the first passage time from any* $i_h \in \mathcal{V}^{(h)}$ *to* $j \in \{1, \ldots, n\}$*, then*

$$m_{i_1 \ldots i_N, j} = 1 + \sum_{k_1, \ldots, k_N \neq j} m_{k_1 \ldots k_N, j} p^{(1)}_{i_1, k_1} \ldots p^{(N)}_{i_N, k_N},$$

or, in matrix notation,

$$M = \mathbb{1}_\alpha \mathbb{1}_n^T + (P^{(1)} \otimes \cdots \otimes P^{(N)}) M - (P^{(1)} \otimes \cdots \otimes P^{(N)}) M_d, \tag{3.14}$$

where $\alpha = \prod_{h=1}^{N} |V^{(h)}|$*,* $M = [m_{i_1 \ldots i_N, j}]$ *and* $M_d = [\delta_{i_1 \ldots i_N, j} m_{i_1 \ldots i_N, j}]$*.*

Proof. The formulation of this system follows in a similar fashion the N agent case, with the exception that now, if node i_h has the property that $i_h \notin V^{(h)} \subset \{1, \ldots, n\}$ then the corresponding $m_{i_1 \ldots i_N, j}$ value is zero. □

Given the formulation for the N-agent first passage time matrix over multiple subgraphs, we now determine the average first visit time to any node in the full graph. Like before, first we calculate the relative frequency of being at any given node. Unlike before, however, each agent operates over a subset of the nodes in

the graph. Let $\boldsymbol{\pi}^{(h)}$ be the stationary distribution associated with Markov chain $P^{(h)} \in \mathbb{R}^{r\times r}$, where $r \leq n$, and for convenience assume that $V^{(h)} \subset \{1,\ldots,n\}$ is an ordered ascending set (i.e., $V^h = \{3, 9, 20\}$). Then, let $\tilde{\boldsymbol{\pi}}^{(h)}$ be the vector whose entries are given by $\tilde{\boldsymbol{\pi}}^{(h)}_{V_i^{(h)}} = \boldsymbol{\pi}_i^{(h)}$ for $i \in \{1,\ldots,|V^{(h)}|\}$ and $\tilde{\boldsymbol{\pi}}_i^{(h)} = 0$ otherwise. In other words, $\tilde{\boldsymbol{\pi}}^{(h)}$ corresponds to the stationary distribution of each agent over the entire graph, not just its subgraph. Therefore if an agent never visits a node, its visit frequency to that node is 0. Given the padded vector $\tilde{\boldsymbol{\pi}}^{(h)}$, we write the average visit frequency as $\tilde{\boldsymbol{\pi}}_{\text{ave}} = \sum_{h=1}^{N}(\tilde{\boldsymbol{\pi}}^{(h)})/N$. Notice that this interpretation of average visit frequency takes into account that multiple Markov chains are running in parallel.

Now, the *N-agent mean first passage time from start nodes i_h, $h \in \{1,\ldots,N\}$*, denoted $\boldsymbol{h}_{i_1\ldots i_N}$, is given by

$$\boldsymbol{h}_{i_1\ldots i_N} = \sum_{j=1}^{n} m_{i_1\ldots i_N,j}\tilde{\boldsymbol{\pi}}_{\text{ave},j}.$$

Therefore, the average time to go from any set of N nodes to a single node in a graph is given by

$$\begin{aligned} H_N &= \sum_{i_1=1}^{|V^{(1)}|} \boldsymbol{\pi}_{i_1}^{(1)} \cdots \sum_{i_N=1}^{|V^{(N)}|} \boldsymbol{\pi}_{i_N}^{(N)} \sum_{j=1}^{n} m_{i_1\ldots i_N,j}\tilde{\boldsymbol{\pi}}_{\text{ave},j} \\ &= (\boldsymbol{\pi}^{(1)} \otimes \cdots \otimes \boldsymbol{\pi}^{(N)}) M \tilde{\boldsymbol{\pi}}_{\text{ave}} \\ &= (\tilde{\boldsymbol{\pi}}_{\text{ave}} \otimes \boldsymbol{\pi}^{(1)} \otimes \cdots \otimes \boldsymbol{\pi}^{(N)})\,\text{vec}[M], \end{aligned} \tag{3.15}$$

where as before, we denote H_N as the N-agent group hitting time.

We are now ready to state our main result.

Theorem 25. *(Group hitting time for irreducible subgraphs): Consider the N graphs $\mathcal{G}^{(h)} = (\mathcal{V}^{(h)}, E^{(h)}, P^{(h)})$ satisfying the property $\cup_{h=1}^{N} \mathcal{V}^{(h)} = \{1, \ldots, n\}$ and let $P^{(h)} \in \mathbb{R}^{|V^{(h)}| \times |V^{(h)}|}$ be the* irreducible *transition matrices associated with $\mathcal{G}^{(h)}$. Also, let $\tilde{\boldsymbol{\pi}}_{\text{ave}} = (\sum_{i=1}^{N} \tilde{\boldsymbol{\pi}}^{(i)})/N$, and let $E \in \mathbb{R}^{\alpha n \times \alpha n}$ be the diagonal matrix which satisfies the equality $E \operatorname{vec}[M] = \operatorname{vec}[M_d]$. Then the following hold:*

(i) the group hitting time of the Markov chain is given by

$$\begin{aligned} H_N &= (\tilde{\boldsymbol{\pi}}_{\text{ave}} \otimes \boldsymbol{\pi}^{(1)} \otimes \cdots \otimes \boldsymbol{\pi}^{(N)})^T \operatorname{vec}[M], \quad \textit{where} \\ \operatorname{vec}[M] &= (I_{\alpha n} - (I_n \otimes P^{(1)} \otimes \cdots \otimes P^{(N)})(I_{\alpha n} - E))^{-1} \mathbb{1}_{\alpha n}, \end{aligned} \tag{3.16}$$

and $\alpha = \prod_{h=1}^{N} |V^{(h)}|$, and

(ii) the hitting time between nodes $h_1, \ldots, h_N$ and k, denoted $m_{h_1 \ldots h_N, k}$, of the Markov chain is given by

$$m_{h_1 \ldots h_N, k} = \operatorname{vec}[[\mathcal{I}^{h_1 \ldots h_N, k}_{i_1 \ldots i_N, j}]]^T \operatorname{vec}[M].$$

Proof. The proof of this theorem follows the exact same logic as the proof of Theorem 23 □

This theorem immediately leads to the following corollary.

Corollary 26. *(Group hitting time for reducible Markov chains): Consider N multiple Markov chains, each with a transition matrix $P^{(h)} \in \mathbb{R}^{n \times n}$ satisfying*

the property that if $P^{(h)}$ *is reducible, then there exists a permutation matrix* $S^{(i)}$ *such that* $(S^{(h)})^T P^{(h)} S^{(h)}$ *is block diagonal with exactly one irreducible component. Let* $\boldsymbol{\pi}_{\text{ave}} = (\sum_{h=1}^{N} \boldsymbol{\pi}^{(h)})/N$*, with the property that* $\boldsymbol{\pi}_{\text{ave},j} \neq 0$ *for all* j*. Also, let* $E \in \mathbb{R}^{n^{N+1} \times n^{N+1}}$ *be the diagonal matrix which satisfies the equality* $E \operatorname{vec}[M] = \operatorname{vec}[M_d]$*. Then the following properties hold:*

(i) the group hitting time of the Markov chain is given by

$$H_N = (\boldsymbol{\pi}_{\text{ave}} \otimes \boldsymbol{\pi}^{(1)} \otimes \cdots \otimes \boldsymbol{\pi}^{(N)})^T \operatorname{vec}[M], \quad \textit{where}$$

$$\operatorname{vec}[M] = (I_{n^{N+1}} - (I_n \otimes P^{(1)} \otimes \cdots \otimes P^{(N)})(I_{n^{N+1}} - E))^{-1} \mathbb{1}_{n^{N+1}}, \textit{ and}$$

(ii) the hitting time between nodes $h_1, \ldots, h_N$ *and* k*, denoted* $m_{h_1 \ldots h_N, k}$*, of the Markov chain is given by*

$$m_{h_1 \ldots h_N, k} = \operatorname{vec}[[\mathcal{I}^{h_1 \ldots h_N, k}_{i_1 \ldots i_N, j}]]^T \operatorname{vec}[M].$$

Proof. First, note that the Kronecker product of two block diagonal matrices generates a block diagonal matrix. Second, notice from Definition 17 property (i) that

$$(S^{(1)})^T P^{(i)} S^{(1)} \otimes \cdots \otimes (S^{(N)})^T P^{(i)} S^{(N)} =$$

$$((S^{(1)})^T \otimes \cdots \otimes (S^{(N)})^T)(P^{(1)} \otimes \cdots \otimes P^{(N)})(S^{(1)} \otimes \cdots \otimes S^{(N)}),$$

and so there exists a permutation matrix $(S^{(1)} \otimes \cdots \otimes S^{(N)})$ that makes $(P^{(1)} \otimes \cdots \otimes P^{(N)})$ block diagonal. Since exactly one block from each matrix is not exactly zero, the

same is true of $(S^{(1)})^T P^{(i)} S^{(1)} \otimes \cdots \otimes (S^{(N)})^T P^{(i)} S^{(N)}$. This block corresponds to matrix P in Theorem 25. The rest of the proof follows by noticing that each node in the graph is reached if and only if $\boldsymbol{\pi}_{\text{ave}}, j \neq 0$ for all j. This is due to the fact that $\boldsymbol{\pi}_k^{(h)} \neq 0$ when k denotes a persistent reachable node in the graph, and $\boldsymbol{\pi}_k^{(h)} = 0$ otherwise. □

3.3.3 Computational Complexity

Due to the extensive use of Kronecker products, it is important to verify the memory and computational costs of the group hitting time. Looking at equation (3.11), we can assert that the group hitting time is affected by the curse of dimensionality; with n nodes and N agents, the matrix $I_n \otimes P^{(1)} \otimes \cdots \otimes P^{(N)}$ contains n^{2N+2} elements. For example, given $n = 100$ nodes and $N = 10$ agents the size of that matrix is $10^{44} \times 10^{44}$. The most intense operation is the inversion of $I_n \otimes P^{(1)} \otimes \cdots \otimes P^{(N)}$, which requires a cost of $O(k^3)$ where k is the number of elements in the matrix [15], thus in our case this becomes $O(n^{6N+6})$. Noticing that the first Kronecker product in (3.12) is between the identity matrix and the $P = P^{(1)} \otimes \cdots \otimes P^{(N)}$ matrix, this implies $I_n \otimes P$ is block diagonal and therefore we can store and invert single blocks, reducing memory cost to $O(n^{2N})$ and inversion cost to $O(n^{6N})$. For the more general hitting time formula described by equation (3.15), the complexity can be further reduced. Given (3.15)

describes random-walks on subgraphs, then $|V^{(h)}| = n\beta_h$ for some $\beta_h \in (0, 1]$, and therefore the number of elements in the matrix $I_{\alpha n} \otimes P^{(1)} \otimes \cdots \otimes P^{(N)}$ is $(\prod_{h=1}^{N} \beta_h^2) n^{2N+2}$. This leads to a computational complexity for the inversion equal to $O\left((\prod_{h=1}^{N} \beta_h^6) n^{6N+6}\right)$. Similar to before, we can take advantage of the fact that the first Kronecker product in $I_{\alpha n} \otimes P^{(1)} \otimes \cdots \otimes P^{(N)}$ is the identity matrix, reducing memory and inversion costs to $(\prod_{h=1}^{N} \beta_h^2) n^{2N}$ and $O\left((\prod_{h=1}^{N} \beta_h^6) n^{6N}\right)$, respectively.

In special circumstances, the above computational complexity can be dramatically reduced. This happens when the intersection between a subgraph of a single agent, does not intersect with any other agents' subgraph. In this case, the disjoint agent's hitting time over its subgraph can be calculated independently of the group hitting time of the other $N - 1$ agents. The single agent hitting time can then be averaged with the $N-1$ agent group hitting time to generate the N agent group hitting time. In the case where every agent owns its own disjoint region, the computational complexity scales to $O(\sum_{h=1}^{N} |V^{(h)}|^{12})$, or to $O(\sum_{h=1}^{N} |V^{(h)}|^{6})$ when exploiting the Kronecker product between the identity matrix and a transition matrix as was done previously. It is clear that uniformly partitioning the graph between agents, or partitioning as close to uniform as is possible, we get the lowest computational complexity.

In the next section we compute an optimized group hitting time for various graph topologies.

3.4 Numerical optimization of the group hitting time

In the following sections we study the transition matrices that arise from the numerical optimization of the group hitting time. In particular, we look at the minimization of (3.16) described by Problem 6 below. The problem is numerically solved using a *sequential quadratic programming* solver as implemented by MATLAB's `fmincon` optimization algorithm, details of which are discussed in section 3.4.3.

Problem 6. *(Group hitting time minimization): Let* $H_1^{(i)}$ *denote the single agent hitting time for random-walker* $i \in \{1, \ldots, N\}$*. Given the stationary distributions* $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_N$ *and Graph* $\mathcal{G}$ *with vertex set* V *and edge set* E*, find the transitions*

matrices $P^{(1)}, P^{(2)}, \ldots, P^{(N)}$ *solving:*

$$\begin{aligned}
\text{minimize} \quad & (\tilde{\boldsymbol{\pi}}_{\text{ave}} \otimes \boldsymbol{\pi})^T (I_{\alpha n} - (I_n \otimes (P^{(1)} \otimes P^{(2)} \cdots \otimes P^{(n)}))(I_{\alpha n} - E))^{-1} \mathbb{1}_{\alpha n} \\
\text{subject to} \quad & P^{(i)} \mathbb{1}_{|V^{(i)}|} = \mathbb{1}_{|V^{(i)}|}, \text{ for each } i \in \{1, \ldots, N\} \\
& (\boldsymbol{\pi}^{(i)})^T P^{(i)} = (\boldsymbol{\pi}^{(i)})^T, \text{ for each } i \in \{1, \ldots, N\} \\
& 0 \leq p_{h,k}^{(i)} \leq 1, \text{ for each } (h,k) \in E \text{ and } i \in \{1, \ldots, N\} \\
& p_{h,k}^{(i)} = 0, \text{ for each } (h,k) \notin E \text{ and } i \in \{1, \ldots, N\} \\
& P^{(i)} \text{ is irreducible for } i \in \{1, \ldots, N\}.
\end{aligned}$$

The constraints in Problem 6, including the final one on the irreducibility of $P^{(i)}$, guarantee that the conditions of Theorem 25 are satisfied. In practice, it is hard to enforce the irreducibility constraint during each step of an iterative optimization algorithm; our approach is to relax the constraint and verify a posteriori that the iteratively-computed solution satisfies the irreducibility constraint. In all the computational settings we considered, we never encountered a solution that violated the irreducibility constraint.

In order to build intuition on the Markov chain combinations that generate optimal group hitting time values, we present numerical results for the ring graphs, complete graph and lattice graph shown in Figure 3.1. We look at two cases in particular; one in which every random walker is required to visit all nodes in the graph, and one in which random walkers are allowed to visit subgraphs. For simplicity, we always restrict the stationary distribution $\tilde{\boldsymbol{\pi}}_{\text{ave}}$ to be uniform.

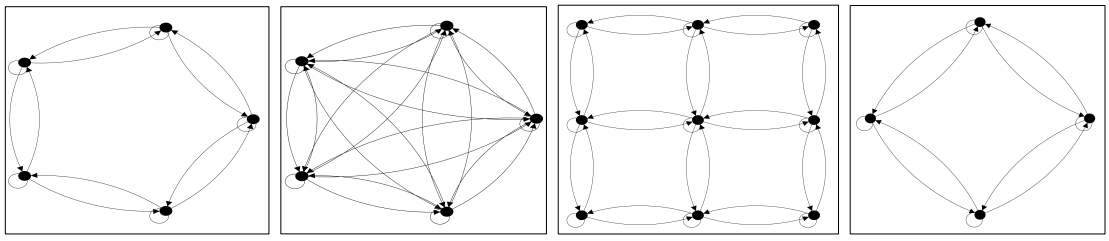

Figure 3.1: From left to right, example of a 5 node ring, 5 node complete, 9 node lattice and 4 node ring graph with self-loops.

3.4.1 Random-walkers covering the full graph

In the following we study which Markov chains generate optimal group hitting time values when every random walker must visit every node in the graph. Surprisingly in fact, we will observe the random walks that generate optimal group hitting times can be different. In each example we define individual agent's stationary distribution as the uniform distribution. It is easily verified that with this choice of stationary distribution, the condition $\tilde{\boldsymbol{\pi}}_{\text{ave},j} = \tilde{\boldsymbol{\pi}}_{\text{ave},k}$ for all j, k is always met no matter how many agents are added to the system.

We begin with the ring graph. For a singe random walker, the transition matrix which generates the minimal hitting time is simply the one describing a cycle (i.e, moving to a neighboring node with probability 1), and for the 5 node ring graph shown in Figure 3.1. This is as expected as a cycle describes the fastest time to reach any node from any other node. It turns out, for the multi-agent case the optimal group hitting time occurs when every agent performs its own cycle; an

example of this for three random-walkers is shown in Figure 3.2. Moreover, since the group hitting time averages over all potential initial conditions, the direction of cycles does not matter. In other words, one agent can go clockwise while the other goes counter-clockwise. A summary of the optimal group and individual random walker hitting times on a ring graph shown in Figure 3.1 is given in Table 3.1.

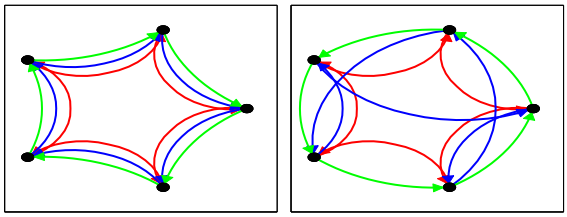

Figure 3.2: Probability to move along each edge of a ring graph (left) and complete graph (right) for 3 random walkers. In the case above, the probability to move along each edge is 1 which indicates a cycle. The group hitting time for the shown trajectories for both ring and complete graph are $H_3 = 1.8$

Random Walker(s)	**Red**	**Blue**	**Green**	H_N
One	3.0	–	–	3
Two	3.0	3.0	–	2.2
Three	3.0	3.0	3.0	1.8

Table 3.1: Group hitting time values for random walks shown in Figure 3.3. The last column indicates the group hitting time for each case whereas the middle three columns indicate each random-walkers individual hitting time. Surprisingly, the ring and complete graph exhibit equivalent hitting time results.

Given the results for a ring graph, one can imply what will happen for the complete graph for a single agent. As it happens, since a cycle exists, this is also

the optimal strategy for the complete graph for both the single and multi-random walker cases. Again direction of cycle does not matter as is seen for the three agent case shown in Figure 3.2. A summary of the optimal group and individual random walker hitting times on a complete graph shown in Figure 3.1 is equivalent to the ring graph and thus is given in Table 3.1.

Next, we look at the lattice graph. Figure 3.3 shows the optimal trajectories found ranging from a single random walker up to 3 random walkers. It is interesting to note that individual agents trajectories are quite different unlike for the ring and complete graphs. This can be more easily seen by observing each agent's individual hitting time as shown in Table 3.2. What is surprising is that the transition matrices that generate the optimal group hitting time for the multi-random waker cases are in fact sub-optimal individually. Interestingly enough, if one substitutes the optimal single agent transition matrix from the single random walker case in for any/all of the multi-waker transition matrices, the group hitting time becomes worse for those multi-waker cases.

Random Walker(s)	**Red**	**Blue**	**Green**	H_N
One	6.8	–	–	6.8
Two	7.7	10.5	–	4.1
Three	7.0	15.9	16.9	2.9

Table 3.2: Group hitting time values for random walks shown in Figure 3.3. The last column indicates the group hitting time for each case whereas the middle three columns indicate each random-walkers individual hitting time.

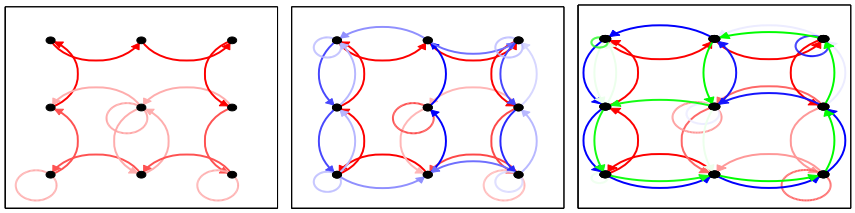

Figure 3.3: Probability to move along each edge of a lattice graph for 1 random walker (left), 2 random walkers (middle) and 3 random walkers (right). In each graph, the opacity of a line indicates the probability to move along an edge.

Thus far we have seen that repeating multiple copies of the fastest random walk is not always the most optimal. In fact, through simulation we've seen that repeating random walks is only optimal when a cycle is the optimal single agent strategy. In the following section we explore in more detail how the optimal group hitting time is affected when working with sub graphs.

3.4.2 Random-walkers covering subgraphs

In the following section we build intuition on how the group hitting time is affected over subgraphs. We will observe that working with subgraphs sometimes improves the group hitting time, but can also cause the group hitting time to worsen. We look at two cases in particular, when the subgraphs overlap and when they do not (i.e., the nodes are partitioned amongst random walkers). The subgraphs are not necessarily allocated in any optimal way, they are simply chosen so that the stationary distribution is uniform over all nodes. For the simple

examples shown, the stationary distributions for each random walker are defined as $\boldsymbol{\pi}_j^{(i)} = \tilde{\boldsymbol{\pi}}_{\text{ave},j}/N_j$ where N_j denotes the number of agents who share node j. For example, $\boldsymbol{\pi}_j^{(i)} = \tilde{\boldsymbol{\pi}}_{\text{ave},j}$ if node j is only owned by one agent. For comparison with results presented in the previous section, we work with the ring and lattice graphs shown in Figure 3.1.

The case when subgraphs overlap is studied for the 5 node ring and 9 node lattice graph, the results of which can be seen in Figure 3.4. For the ring graph shown, the optimal group hitting time is $H_2 = 2.5$ in contrast to $H_2 = 2.2$ from full graph case (Table 3.1). For the lattice graph, the group hitting time is $H_2 = 3.6$ in contrast to $H_2 = 4.1$ (Table 3.2). Therefore, from these two examples, we see that each agent covering less nodes is not always indicative of lower group hitting time.

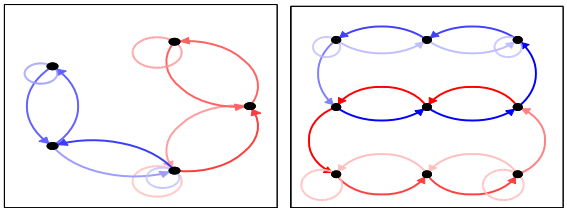

Figure 3.4: Probability to move along each edge of a 5 node ring graph with two agents (left), and 9 node lattice graph with two agents (right). In each graph, the opacity of a line indicates the probability to move along an edge.

The case when subgraphs are partitioned is analyzed for the 9 node lattice and 4 node ring graph shown in Figure 3.5. For the ring graph we see that the group

hitting time is $H_2 = 1.5$ whereas for the case where each agent covers the whole graph $H_2 = 1.9$; we do not show the figure for the latter case, however, recall that the optimal full graph trajectory is simply a cyclic tour for each agent. Now, for the partitioned lattice graph, the group hitting time is $H_3 = 3.7$ in contrast to $H_3 = 2.9$ (Table 3.2). As before, we see that each agent covering less nodes, which are partitioned, is not indicative or lower group hitting time.

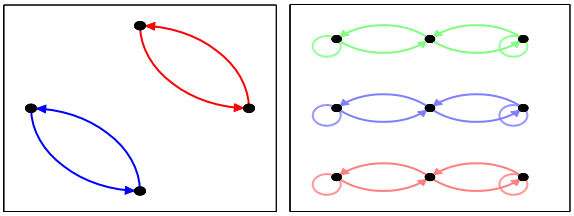

Figure 3.5: Probability to move along each edge of a 4 node ring graph with two agents (left), and 9 node lattice graph with three agents (right). In each graph, the opacity of a line indicates the probability to move along an edge.

Clearly, the small sample study presented here leaves open many future avenues that can be explored when attempting to optimize the group hitting time. For example, notice that one can not always partition a graph and achieve an arbitrary $\tilde{\boldsymbol{\pi}}_{\text{ave}}$. Also, it's unclear what happens when you allow $\tilde{\boldsymbol{\pi}}_{\text{ave}}$ and therefore individual stationary distributions to vary. We leave these, among other questions to future work.

3.4.3 Implementation Notes

In order to generate repeatable and accurate results, we utilize a *sequential quadratic programming* (SQP) solver as implemented in MATLAB's `fmincon` optimization algorithm. Several other solvers are available, however, the SQP solver has the most desirable mathematical properties. More specifically, with the SQP solver, given that the maximum number of iterations is not reached, the minimization algorithm stops when the first-order necessary Karush-Kuhn-Tucker conditions are approximately satisfied; conditions are satisfied in the norm-sense with a tolerance of 10^{-6}. For all results used, the first-order optimality stopping criteria were satisfied.

The group hitting time results presented were taken as the minimum of 1000 optimization runs, each starting from a random initial conditions. Surprisingly, with the exception of the lattice, every initial condition converged to the minimum group hitting time value (within a 10^{-6} to 10^{-11} tolerance). For the two and three agent results presented in Table 3.2, we found that solutions converged to the minimum group hitting time value within a 10^{-2} tolerance for 96 and 93 percent of samples, respectively. Therefore, we claim with reasonable confidence that a local minimum for each group hitting time value was reached, if not a global minimum. On a desktop computer with an Intel i7-4790 processor and 8 Gb of RAM running MATLAB 2014b, the lattice with 9 nodes and 3 agents shown in

Figure 3.2 took the longest time to run, averaging 10 minutes per run, whereas all other simulations took less than a minute to fractions of a second per case run.

3.5 Summary

We have studied the hitting time of multiple random walkers on a graph and have presented the first formulation of this quantity, which we denote the group hitting time. Moreover, we have presented the first closed form solution for calculating the first hitting time between any specified set of nodes in a graph for both the single and multi-agent cases. Finally, we posed the group hitting time as an optimization problem and provided detailed simulation results which help to build insight into the transition matrices that minimize this quantity.

This concludes our discussion of the hitting time. In the next chapter we begin our discussion on multi-agent coverage control.

Chapter 4

Partitioning with One-to-Base-Station Communication

In this chapter and the following we discuss problems related to partitioning and coverage control. The chapter is organized as follows. In Section 4.1 we setup preliminary notation, introduce the concept of partitions and coverings, introduce the one-to-base station network model and present our problem in technical detail. In Section 4.2 we introduce some novel cost functions and present the solution to our problem. In Section 4.3 we discuss implementation issues and present simulation results. In the final section we summarize our findings.

4.1 Preliminaries and problem statement

The one-to-base-station communication model studied in this paper requires that, in the design of coverage algorithms, we adopt overlapping coverings instead of partitions. In this Section we translate concepts used in partitioning of continuous environments [13] to coverings on graphs. In our notation, $\mathbb{R}_{>0}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{Z}_{\geq 0}$ respectively denote the sets of positive, nonnegative and non-negative integer numbers. Given a set A, $|A|$ denotes the number of elements in A. Given sets A, B, their union and intersections are denoted as $A \cup B$ and $A \cap B$, respectively, and their difference is $A \setminus B = \{a \in A \mid a \notin B\}$.

4.1.1 Graphs and Distances

Let the finite set Q be a set of points in a continuous environment. These points can represent locations or small areas of interest. They are assumed to be the nodes of an (undirected) weighted graph $G(Q) = (Q, \mathcal{E}, \Omega)$ with edge set $\mathcal{E} \subset Q \times Q$ and, with a slight abuse of notation, weight matrix $\Omega = [\Omega_{i,j}]$ with the property that $\Omega_{i,j} = \Omega_{j,i} > 0$ if $(i, j) \in \mathcal{E}$ and $\Omega_{i,j} = 0$ otherwise. We assume that $G(Q)$ is connected and think of the $\Omega_{i,j}$ edge weights as travel distances between nearby nodes.

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

4.1.2 Coverings of Graphs

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

Definition 27 (n-Covering)**.** *Given the graph $G(Q) = (Q, \mathcal{E}, \Omega)$, we define a* n-covering of Q *as a collection $P = \{P_i\}_{i=1}^n$ of subsets of Q such that:*

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

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

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

Note that a vertex in Q may belong to multiple subsets in P, i.e., a vertex may be covered by multiple agents. The above definition is an important change from prior work [26], which was limited to partitions of Q, defined as follows.

Definition 28 (n-Partition)**.** *A* n-partition *is a* n*-covering with the additional property that:*

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

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

Among the ways of covering Q, there is one which is worth special attention. Before we state the partition, let us define the vector of weights $w := \{w_1, \ldots, w_n\}$, such that $w_i > 0$ and $\sum_{j=1}^{n} w_i = 1$. For brevity, we denote $\mathcal{W} = \{w \in \mathbb{R}^n_{>0} \mid \sum_{i=1}^{n} w_i = 1\}$. Then given $w \in \mathcal{W}$ and a vector of distinct points $c \in Q^n$, the partition $P \in \mathrm{Part}_n(Q)$ is said to be a *multiplicatively weighted Voronoi partition* of Q generated by c and weighted by w if, for each P_i and all $k \in P_i$, we have $c_i \in P_i$ and $\frac{1}{w_i} d_G(k, c_i) \leq \frac{1}{w_j} d_G(k, c_j)$, for $j \neq i$. The elements of c are said to be the generators of the Voronoi partition multiplicatively weighted by w. Note that there can be more than one multiplicatively-weighted Voronoi partition generated by c and w since how to assign tied vertices is unspecified. Multiplicatively-weighted Voronoi partition allow us to accommodate heterogeneous agents. For example, if agent i is faster than another agent j (i.e.,

$w_i > w_j$), it would make sense that agent i should control more territory than agent j. Multiplicatively-weighted Voronoi partitions are a subset of generalized Voronoi partitions, which will be discussed in further detail in Chapter 5 From this point forward we refer to multiplicatively-weighted Voronoi partitions simply as *Voronoi partitions* and the vector of weights w is given and fixed. Given that weights are fixed, for the rest of the paper we refer to a Voronoi partition generated by c and w simply as a Voronoi partition generated by c.

Given the above, we are now ready to state the one-to-base station network model.

4.1.3 One-to-Base-Station Robotic Network Model

Given a team of n robotic agents and a central base station, each agent $i \in \{1, \ldots, n\}$ is required to have the following basic computation capabilities:

(C1) agent i can identify itself to the base station; and

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

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

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

The base station must have the following capabilities:

(C4) it can store an arbitrary n-covering of Q, $P = \{P_i\}_{i=1}^n$, a list of locations $c \in Q^n$ and weights $w \in \mathcal{W}$;

(C5) it can perform computations on subgraphs of $G(Q)$; and

(C6) it can store and operate on multiple n-coverings of Q, $P = \{P_i\}_{i=1}^n$ and a list of locations $c \in Q^n$.

4.1.4 Problem Statement

We are now ready to state our problem of interest.

Given weights $w \in \mathcal{W}$ assume that, for all $t \in \mathbb{R}_{\geq 0}$, each agent $i \in \{1, \ldots, n\}$ maintains in memory a subset $S_i(t)$ of environment Q and a vertex $s_i(t) \in S_i(t)$. Our goal is to iteratively update the covering $S(t) = \{S_i\}_{i=1}^n$ and the centers $s(t) = \{s_i\}_{i=1}^n$ while solving the optimization problem:

$$\min_{s \in Q^n} \min_{S \in \mathrm{Cov}_n(Q)} U(s, S), \tag{4.1}$$

for some cost function $U(s, S)$ subject to the constraint that every node in the environment Q maintains coverage from some agent, and subject to the constraint

imposed by the robot network model with asynchronous one-to-base-station communication.

4.2 Proposed Solution

In this section we present our proposed solution to the problem described in Section4.1.4. We begin by first introducing some useful cost functions and their properties. Then, we present an algorithm that uses these cost functions and show that it solves (4.1).

4.2.1 Cost Functions

Let *weight function* $\phi : Q \to \mathbb{R}_{>0}$ be a bounded positive function which assigns a relative weight to each element of Q. The weight assigned to a node by ϕ can be thought of as the "service time" or importance of that node. The *one-center function* $\mathcal{H}_1$ gives the cost for a robot to cover a subset $A \subset Q$ from a vertex $h \in A$ with relative prioritization given by ϕ:

$$\mathcal{H}_1(h; A) = \sum_{k \in A} d_G(h, k)\phi(k).$$

This cost function leads us to the following definition.

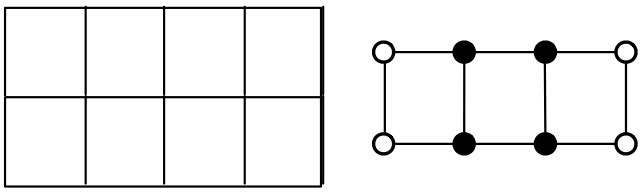

Figure 4.1: The left image shows a grid environment whose corresponding graph representation is shown in the right image. Each cell in the grid represents a node in the graph and if two cells are adjacent, then there is an unit-weight edge between those nodes. The black nodes in the graph denote the set of generalized centroids for the corresponding grid environment.

Definition 29 (Centroid)**.** *We define the set of generalized centroids of $A \subset Q$ as the set of vertices in A which minimize $\mathcal{H}_1$, i.e.,*

$$\mathrm{C}(A) := \operatorname*{argmin}_{h \in A} \mathcal{H}_1(h; A).$$

In what follows, we drop the word "generalized" for brevity. Note that the centroid of a set always belongs to the set. Figure 4.1 shows an illustrative example of the set $\mathrm{C}(A)$ for a simple environment.

With these notions, we are ready to define other useful cost functions. We can define the multi-center function $\mathcal{H} : Q^n \times \mathrm{Cov}_n(Q) \to \mathbb{R}_{\geq 0}$ to measure the cost for n robots to cover a n-covering P from the vertices $c \in Q^n$ by

$$\mathcal{H}(c, P) = \sum_{i=1}^{n} \sum_{k \in P_i} \frac{1}{w_i} d_G(c_i, k)\phi(k).$$

Note that if $w_i = w_j$ for all i, j, then the multi-center cost function above is the same as in [26]. Furthermore, we define the minimum cost-to-cover mapping

$\mathcal{H}_{\min} : Q^n \times \mathrm{Cov}_n(Q) \to \mathbb{R}_{\geq 0}$ by

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

Note that if P is a partition, then $\mathcal{H}_{\min}(c, P) = \mathcal{H}(c, P)$ for any c. We aim to minimize these performance functions with respect to both the covering P and the vertices c. In the motivational scenario we are considering, each robot will periodically be asked to perform a task somewhere in its region with tasks located according to distribution ϕ. When idle, the robots would position themselves at the vertices c. By minimizing the coverage cost, the robot team minimizes the expected distance between a task and the furthest robot which can service the task.

We are almost ready to introduce a notion of optimal partition, the centroidal Voronoi partition. Our discussion begins with the following results, which are direct consequences of the above definitions.

Proposition 30 (Properties of $\mathcal{H}$)**.** *Let $P \in \mathrm{Part}_n(Q)$ and $c \in Q^n$ then the following properties hold:*

(i) If P' is a Voronoi partition generated by c, then

$$\mathcal{H}(c, P') \leq \mathcal{H}(c, P).$$

(ii) Let $\mathcal{I} \subset \{1, \ldots, n\}$. If $c' \in Q^n$ satisfies $c'_i \in C(P_i)$ for $i \in \mathcal{I}$ and $c'_j = c_j$ for $j \notin \mathcal{I}$, then

$$\mathcal{H}(c', P) \leq \mathcal{H}(c, P)$$

with a strict inequality if $c_i \notin C(P_i)$ *for any* $i \in \mathcal{I}$.

Proposition 31 (Properties of $\mathcal{H}_{\min}$)**.** *Let* $P' \in \mathrm{Part}_n(Q)$ *be a Voronoi partition generated by* $c \in Q^n$*, then the following properties hold:*

(i) If $P \in \mathrm{Cov}_n(Q)$ *is a covering such that* $P'_i \subseteq P_i$ *for all* i*, then*

$$\mathcal{H}_{\min}(c, P') = \mathcal{H}_{\min}(c, P).$$

(ii) If $\overline{P} \in \mathrm{Part}_n(Q)$ *is a partition satisfying* $\overline{P}_i \cap P'_i \ni c_i$ *for all* i*, then*

$$\mathcal{H}_{\min}(c, P') \leq \mathcal{H}_{\min}(c, \overline{P}).$$

(iii) Let $\mathcal{I} \subset \{1, \ldots, n\}$*. If* $c' \in Q^n$ *satisfies* $c'_i \in C(P'_i)$ *for* $i \in \mathcal{I}$ *and* $c'_j = c_j$ *for* $j \notin \mathcal{I}$*, then*

$$\mathcal{H}_{\min}(c', P') \leq \mathcal{H}_{\min}(c, P')$$

with a strict inequality if $c_i \notin C(P'_i)$ *for any* $i \in \mathcal{I}$.

Propositions 30 and 31 imply that if $P \in \mathrm{Part}_n(Q)$ and (c, P) minimizes $\mathcal{H}$ (equivalently $\mathcal{H}_{\min}$), then $c_i \in C(P_i)$ for all i and P must be a Voronoi partition generated by c. This motivates the following definition.

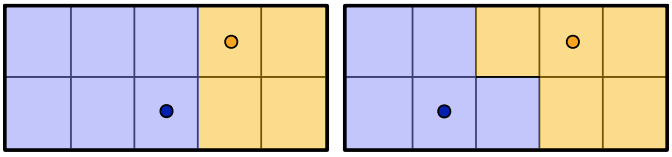

Figure 4.2: The figure shows two environments with two agents. Each cell denotes a node in a graph and if two cells are adjacent, then there is a unit-weight edge between those nodes. The left image shows a Voronoi partition generated by the two agents. Note that the blue agent is not at its region's centroid. The right image is instead a centroidal Voronoi partition.

Definition 32 (Centroidal Voronoi Partition)**.** $P \in \mathrm{Part}_n(Q)$ *is a* centroidal Voronoi partition *of* Q *if there exists a* $c \in Q^n$ *such that* P *is a Voronoi partition generated by* c *and* $c_i \in \mathrm{C}(P_i)$ *for all* i*.*

For a given environment Q, a pair made of a centroidal Voronoi partition and the corresponding vector of centroids is locally optimal in the following sense: the cost functions $\mathcal{H}$ and $\mathcal{H}_{\min}$ cannot be reduced by changing either P or c independently. Figure 4.2 demonstrates the difference between a Voronoi and centroidal Voronoi partition.

4.2.2 The One-to-Base Coverage Algorithm

Given the cost function defined by $U(s, S)$ and the One-to-Base Network model described by (C1)–(C6), we introduce the *One-to-Base Coverage Algorithm* to solve the optimization problem (4.1).

One-to-Base Coverage Algorithm

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

update $\overline{P} := P$, $\overline{c} := c$, $\overline{\overline{P}} := P$, $\overline{\overline{c}} := c$,
compute sets

$$P_{i,+} := \left\{ x \in Q \;\middle|\; \tfrac{1}{w_i} d_G(x, c_i) < \min\left\{ \tfrac{1}{w_j} d_G(x, c_j) \;\middle|\; x \in P_j,\ j \neq i \right\} \right\}$$

$$P_{i,-} := \left\{ x \in P_i \cap \left(\cup_{i \neq j} P_j \right) \;\middle|\; \tfrac{1}{w_i} d_G(x, c_i) \geq \min\left\{ \tfrac{1}{w_j} d_G(x, c_j) \;\middle|\; x \in P_j,\ j \neq i \right\} \right\}$$

update $\overline{\overline{P}}_i := (P_i \backslash P_{i,-}) \cup P_{i,+}$
for $k \in P_i \backslash c$ **do**
    compute sets

$$P_{i,+} := \left\{ x \in Q \;\middle|\; \tfrac{1}{w_i} d_G(x, k) < \min\left\{ \tfrac{1}{w_j} d_G(x, c_j) \;\middle|\; x \in P_j,\ j \neq i \right\} \right\}$$

$$P_{i,-} := \left\{ x \in P_i \cap \left(\cup_{i \neq j} P_j \right) \;\middle|\; \tfrac{1}{w_i} d_G(x, k) \geq \min\left\{ \tfrac{1}{w_j} d_G(x, c_j) \;\middle|\; x \in P_j,\ j \neq i \right\} \right\}$$

    update $\overline{P}_i := (P_i \backslash P_{i,-}) \cup P_{i,+}$
    update $\overline{c}_i := k$
    if $U\left(\overline{c}, \overline{P}\right) < U\left(\overline{\overline{c}}, \overline{\overline{P}}\right)$ **then**
        update $\overline{\overline{P}}_i := \overline{P}_i$
        update $\overline{\overline{c}}_i := \overline{c}_i$
$P_i^+ := \overline{\overline{P}}_i$
$c_i^+ := \overline{\overline{c}}_i$
tell agent i to set $S_i^+ := P_i^+$ and $s_i^+ = c_i^+$

Remark 33 (Constant cost)**.** *Given the constant cost function $U(c, P) = \alpha$ for $\alpha \in \mathbb{R}$, for a set of initial conditions (c, P), the One-to-Base Coverage Algorithm produces a Voronoi partition generated by c.*

Remark 34 (Full coverage)**.** *Notice that the set $P_{i,+}$ adds points to an agents environment from other agent's territory that are closer to it. Also, notice that $P_{i,-}$ only removes points from agent i's territory if another agent is covering that territory. Defining the sets in this way ensures that every point in the environment will always have coverage by some agent.*

We have the following main result on the limit behavior of the algorithm.

Theorem 35 (Convergence of One-to-Base Coverage Algorithm ($\mathcal{H}_{\min}$))**.** *Consider a network consisting of n robots endowed with the computation capacities (C1), (C2) and communication capacity (C3), and a base station with capacities (C4), (C5) and (C6). Assume the network implements the One-to-Base Coverage Algorithm with $U(c, P) = \mathcal{H}_{\min}(c, P)$. Then the resulting evolution*

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

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

Pareto-Optimal Partitions

Using the algorithms described thus far, ties along partitions' boundaries are not handled in any optimal way and can often be improved. The major source of sub-optimal boundary allocation is due to the discrete nature of how centroids of a region are selected. Often times when an agent has more than one "center" location, the overall partition can become better balanced if the agent takes an alternate center value as its centroid. The following definition and proposition make this notion more precise.

Definition 36 (Pareto-Optimal Partition)**.** *Given a vector of positions* $c = \{c_1, \ldots, c_n\}$*, the Voronoi partition* P *generated by* c *is* Pareto-optimal *if for all* $\overline{c} = \{c_1, \ldots, \overline{c}_i, \ldots, c_n\}$ *for* $i \in \{1, \ldots, n\}$ *such that* $\overline{c}_i \neq c_i$ *and* $\overline{c}_i \in P_i$*, the Voronoi partition* $\overline{P}$ *generated by* $\overline{c}$ *satisfies* $\mathcal{H}(c, P) \leq \mathcal{H}(\overline{c}, \overline{P})$*.*

As an immediate consequence of the definition of a Pareto-optimal partition and of Proposition 30 we can conclude that *every* Pareto-optimal *partition is also a centroidal Voronoi partition.* However, the reverse implication does not hold, as shown in the following example.

Example 1 (One Dimensional Pareto-optimal Partition)**.** *Consider the three environments in Figure 4.3, each with two agents denoted by the colored circles. Assume that each cell denotes a node in the graph and that unit-weight edges con-*

nect any adjacent cells. Assume ϕ is constant. If the environment is partitioned according to leftmost image of Figure 4.3, then each agent is at the centroid of its region, and the graph is a centroidal Voronoi partition whose multi-center function cost-to-cover is $\mathcal{H} = 4$. This partition is clearly not well balanced, and unless ties are broken in some non-trivial way, this is a valid (worst-case) partition that the system can reach. If however, the blue agent moves to its other centroid, as shown in the middle image, then the worst case partition must be a variant of the partition shown in the rightmost image whose cost-to-cover is $\mathcal{H} = 3$. There exists no partition with smaller cost-to-cover (w.r.t. $\mathcal{H}$) by moving any single agent, and hence the partition in the rightmost image is Pareto-optimal.

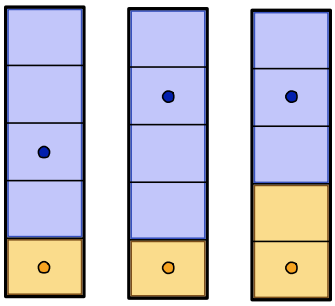

Figure 4.3: *The figure shows three environments with two agents. Each cell denotes a node in a graph, and if two cells are adjacent then there is an unit-weight edge between those nodes.*

The above results give that Pareto-optimal partitions are a subset of centroidal Voronoi partitions. We can define the cost function, $U(c, P)$, in the One-to-Base Coverage Algorithm such that the algorithm converges to a Pareto-optimal

partition. We define the new cost function

$$\mathcal{H}_{\text{inf}}(c) = \sum_{k \in Q} \min_i \big\{ \frac{1}{w_i} d_G(c_i, k) \mid k \in Q \big\} \phi(k). \tag{4.2}$$

Notice that this function is different from $\mathcal{H}_{\text{min}}$ in that it looks for the absolute minimum distance to a point, k. The function $\mathcal{H}_{\text{inf}}$ allows the case when $k \notin P_i$, but $\frac{1}{w_i} d_G(c_i, k) < \frac{1}{w_j} d_G(c_j, k)$, for all $j \neq i$. The $\mathcal{H}_{\text{inf}}$ function is linked to the multicenter function in the following sense.

Proposition 37 (Properties of $\mathcal{H}_{\text{inf}}$)**.** *Given $c \in Q^n$ and $w \in \mathcal{W}$, let P be a Voronoi partition generated by c, then*

$$\mathcal{H}(c, P) = \mathcal{H}_{\text{inf}}(c).$$

Proof. Voronoi partitions are optimal in the sense that $\mathcal{H}(c, P) = \mathcal{H}_{\text{inf}}(c)$ by definition of $\mathcal{H}_{\text{inf}}$. □

We are now ready to state the main result of this subsection. Given the *One-to-Base Coverage Algorithm* with $U(s, S) = \mathcal{H}_{\text{inf}}(c)$ we have the following result.

Theorem 38 (Convergence of One-to-Base Coverage Algorithm ($\mathcal{H}_{\text{inf}}$))**.** *Consider a network consisting of n robots endowed with the computation capacities (C1), (C2) and communication capacity (C3), and a base station with capacities (C4), (C5) and (C6). Assume the network implements the One-to-Base Coverage*

Algorithm with $U(c, P) = \mathcal{H}_{\text{inf}}(c)$. Then the resulting evolution

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

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

Some remarks are in order. First, it is possible for the One-to-base Algorithm with $U = \mathcal{H}_{\text{min}}$ to converge to a Pareto-optimal partition, however, it is not guaranteed as in the case with $U = \mathcal{H}_{\text{inf}}$. Second, if a partition is not Pareto-optimal then the cost to cover a region, in the context of the multi-center function, can be further decreased by making it Pareto-optimal. This point is clarified in Proposition 37, which relates the multi-center function to $\mathcal{H}_{\text{inf}}$. Finally, we emphasize that the difference in the cost-to-cover a region for a Pareto-optimal partition versus a centroidal Voronoi partition decreases as the map defining a region becomes less coarse. This is because the notion of a centroidal Voronoi partition not being Pareto-optimal only exists when a region has more than one centroid, a property of discrete spaces but not of continuous ones. Therefore, as the grid approximating a region becomes less coarse, the more likely it is that a centroidal Voronoi partition is also Pareto-optimal.

Combining $\mathcal{H}_{\min}$ and $\mathcal{H}_{\inf}$

Although the One-to-Base Coverage Algorithm with $U = \mathcal{H}_{\inf}$ is guaranteed to converge to a Pareto-optimal partition whereas the algorithm with $U = \mathcal{H}_{\min}$ is not, the algorithm with $U = \mathcal{H}_{\min}$ is still of practical importance. Computing $\mathcal{H}_{\inf}$ requires that for each node in the graph, all agents compare their relative distances to that node regardless of whether that node exists in the agent's territory or not. The $\mathcal{H}_{\min}$ function, however, only requires a comparison if the node belongs in the agent's territory. Given the computational capabilities of the base station being used, one method may be preferred over the other. A user can take advantage of both algorithm properties by running the algorithm with $U = \mathcal{H}_{\min}$ until it converges to get an initial partition, and then run the algorithm with $U = \mathcal{H}_{\inf}$ to reach a Pareto-optimal solution, if the solution system has not already reached it during the $U = \mathcal{H}_{\min}$ portion of the algorithm. With a slight abuse of notation we will refer to this combined algorithm as the One-to-Base Coverage algorithm with $U = \mathcal{H}_{\min,\inf}$.

4.2.3 Convergence Proofs

In this section we prove Theorems 35 and 38. Any mention to the One-to-Base Coverage Algorithm in this section will refer to the algorithm presented in Section 4.2.2. Their proof is based on the following convergence result for

set-valued algorithms on finite state spaces, which can be recovered as a direct consequence of [12, Theorem 4.3].

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

Lemma 39 (Convergence under persistent switches)**.** *Let (X, d) be a finite metric space. Given a collection of maps $T_1, \ldots, T_m : X \to X$, define the set-valued map $T : X \rightrightarrows X$ by $T(x) = \{T_1(x), \ldots, T_m(x)\}$ and let $\{x_n\}_{n \in \mathbb{Z}_{\geq 0}}$ be an evolution of T. Assume that:*

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

(ii) for all $i \in \{1, \ldots, m\}$, there exists an increasing sequence of times $\{n_k \mid k \in \mathbb{Z}_{\geq 0}\}$ such that $x_{n_k+1} = T_i(x_{n_k})$ and $(n_{k+1} - n_k)$ is bounded.

Let $F_i = \{x \in X \mid T_i(x) = x\}$ be the set of fixed points of T_i. Then, for all $x_0 \in X$ there exist $N \in \mathbb{N}$ and $\bar{x} \in (F_1 \cap \cdots \cap F_m)$ such that $x_n = \bar{x}$ for all $n \geq N$.

Note that the existence of a common fixed point for the collection of maps T_i is guaranteed by this result.

We now apply Lemma 39 to the evolution of One-to-base Coverage Algorithm with $U(c,P) = \mathcal{H}_{\min}(c,P)$ and $U(c,P) = \mathcal{H}_{\inf}(c)$, respectively. To do so, for each function given by $U(c,P)$, we must describe the algorithm as a set-valued map and find a corresponding Lyapunov function. The first step is possible because the One-to-Base Coverage Algorithm is well-posed in the sense of the following immediate result.

Proposition 40 (Well-posedness)**.** *Let $P \in \mathrm{Cov}_n(Q)$ and $c \in Q^n$ such that $c_i \in P_i$ and $c_i \neq c_j$ for all i and all $j \neq i$. Then, P^+ and c^+ produced by the* One-to-Base Coverage Algorithm *meet the same criteria.*

Given this result, the One-to-Base Coverage Algorithm can be written as a set valued map. For any $i \in \{1,\dots,n\}$, we define the map $T_{U,i} : Q^n \times \mathrm{Cov}_n(Q) \to Q^n \times \mathrm{Cov}_n(Q)$ by

$$T_{U,i}(c,P) = \big\{\{c_1,\dots,c_i^+,\dots,c_N\},\\ \{P_1,\dots,P_i^+,\dots,P_n\}\big\},$$

where c_i^+ and P_i^+ are defined per the algorithm when i is the communicating robot, and U is dependent on the cost function we are referring to (i.e., $U = \mathcal{H}_{\min}$ or $U = \mathcal{H}_{\inf}$). Then we can define the set-valued map $T_U : Q^n \times \mathrm{Cov}_n(Q) \mapsto Q^n \times \mathrm{Cov}_n(Q)$ by

$$T_U(c,P) = \{T_{U,1}(c,P),\dots,T_{U,N}(c,P)\}.$$

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

For our Lyapunov arguments we will need to define $M(P)$ as the set of vertices which are owned by multiple agents. We now proceed by stating two useful propositions, which allow us to conclude Theorem 35.

Proposition 41 (Decaying $\mathcal{H}_{\min}$ cost function)**.** *After each iteration of the one-to-base station algorithm if* $(c^+, P^+) \neq (c, P)$ *then one of the following holds:*

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

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

Proof. If $c^+ = c$ then $\mathcal{H}_{\min}(c^+, P^+) \leq \mathcal{H}_{\min}(c, P)$. This is a direct consequence of how the sets $P_{i,+}$ and $P_{i,-}$ are defined. Points are added to P_i if and only if they are strictly closer to c_i than any other center c_j and hence the cost of $\mathcal{H}_{\min}$ must decrease by the addition of these points. Points are removed if and only if they are strictly farther away or tied points and so $\mathcal{H}_{\min}$ must decrease or stay the same. If $c^+ \neq c$ then by lines 8-10 of the algorithm $\mathcal{H}_{\min}(c^+, P^+) < \mathcal{H}_{\min}(c, P)$. For the case $\mathcal{H}_{\min}(c^+, P^+) = \mathcal{H}_{\min}(c, P)$ then for every $x \in P_j \backslash P_i$, for all $j \neq i$, there exists no point that is strictly closer to the center c_i than any other center

$c_j,\ j \neq i$. Therefore, no points can be added to P_i^+, and so if $P^+ \neq P$ it must be the case that $|M(P^+)| < |M(P)|$. □

Proposition 42 (Convergence of $T_{\mathcal{H}_{\min}}$)**.** *The evolution of the One-to-Base Coverage Algorithm $(c(t), P(t))$ generated by the map $T_{\mathcal{H}_{\min}}$ converges in finite time to the intersection of the equilibria of the maps $T_{\mathcal{H}_{\min},i}$, that is, to a pair (c, P) where P is a centroidal Voronoi partition generated by c.*

Proof. The proof proceeds with an application of Lemma 39 to $(c(t), P(t))$. The algorithm is the mapping $T_{\mathcal{H}_{\min}} : Q^n \times \mathrm{Cov}_n(Q) \mapsto Q^n \times \mathrm{Cov}_n(Q)$ defined above and is well-posed. We can form a Lyapunov function using Proposition 41 as follows. Since the set Q is finite, there exists only a finite number of possible values for $\mathcal{H}_{\min}$ and $|M|$. Let ϵ_m be the magnitude of the smallest non-zero difference between any two values of $\mathcal{H}_{\min}$. Let α_M be larger than twice the maximum possible value of $|M|$. Define $V : Q^n \times \mathrm{Cov}_n(Q) \to \mathbb{R}_{\geq 0}$ by

$$V(c, P) = \mathcal{H}_{\min}(c, P) + \frac{\epsilon_m}{\alpha_M}|M(P)|.$$

Thanks to this scaling of $|M(P)|$, Proposition 41 implies that if $(c', P') \in T_{\mathcal{H}_{\min}}(c, P)$, then either $V(c', P') < V(c, P)$ or $(c', P') = (c, P)$. Thus, $V(c, P)$ fulfills assumption (i) in Lemma 39. Moreover, the communication model (C3) assures that assumption (ii) in Lemma 39 is met. Now, applying Lemma 39, we are assured that the dynamics converge to a fixed point (c^*, P^*). It remains to show that P^*

is a centroidal Voronoi partition generated by c^*. We do this by refining in three sequential steps the properties that the fixed point must have: P^* is a partition, (c^*, P^*) is a a Voronoi partition, and finally (c^*, P^*) is a centroidal Voronoi partition. First, if P^* is not a partition, then $P^*_{i,-} \neq \emptyset$; this establishes that P^* is a partition. Second, if the partition P^* is not Voronoi, then $P^*_{i,+} \neq \emptyset$; this establishes that (c^*, P^*) is a Voronoi partition. Third, if P^* is a Voronoi partition generated by c^*, but $c^*_i \notin C(P^*_i)$ for any i, then from part (iii) of Proposition 31 there exists a location c^{**}_i (at a centroid location) that improves the cost to cover P^*_i; line 4 of the algorithm guarantees that this location is checked, and lines 8-10 ensure the position is updated to that location or one with an even lower cost. It should be noted that an update in location to c^{**}_i can simultaneously lead to an update in territory P^*_i: however, given that P^* is a partition, only $P^*_{i,+}$ can contribute to the territory, which further decreases the coverage cost by the definition of $\mathcal{H}_{\min}$ and $P^*_{i,+}$. Therefore, we have established that there exists an update reducing the cost function $T_{\mathcal{H}_{\min}}$ if the fixed point is not a centroidal Voronoi partition. Then, P^* must be a centroidal Voronoi partition generated by c^*. (Note that the fixed point could also potentially have other properties in addition to being a centroidal Voronoi partition, however, establishing those properties are beyond the scope of this proof.)

□

Since updates to agent i in base-station memory also occur on the physical agent, we can conclude the convergence proof of Theorem 35.

Finally, we state two propositions which allow us to conclude Theorem 38.

Proposition 43 (Decaying $\mathcal{H}_{\inf}$ cost function)**.** *After each iteration of the one-to-base station algorithm if* $(c^+, P^+) \neq (c, P)$ *using* $\mathcal{H}_{\inf}$ *as the cost function then one of the following holds:*

(i) $\mathcal{H}_{\inf}(c^+) < \mathcal{H}_{\inf}(c)$*; or*

(ii) $\mathcal{H}_{\inf}(c^+) = \mathcal{H}_{\inf}(c)$*, and*

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

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

Proof. If $c^+ = c$ then $\mathcal{H}_{\min}(c^+, P^+) \leq \mathcal{H}_{\min}(c, P)$ and $\mathcal{H}_{\inf}(c^+) = \mathcal{H}_{\inf}(c)$. This is a direct consequence of how we define $\mathcal{H}_{\inf}$ and the sets $P_{i,+}$ and $P_{i,-}$. Points are added to P_i if and only if they are strictly closer to c_i than any other center c_j and hence the cost of $\mathcal{H}_{\min}$ must decrease by the addition of these points. Points are removed if and only if they are strictly farther away or tied points and so $\mathcal{H}_{\min}$ must decrease or stay the same. If $c^+ \neq c$ then by the lines 8-10 of the algorithm $\mathcal{H}_{\inf}(c^+) < \mathcal{H}_{\inf}(c)$. For the case $\mathcal{H}_{\min}(c^+, P^+) = \mathcal{H}_{\min}(c, P)$ then for every $x \in P_j \backslash P_i$, for all $j \neq i$, there exist no point that is strictly closer to the

center c_i than any other center $c_j,\ j \neq i$. Therefore, no points can be added to P_i^+, and so if $P^+ \neq P$ it must be the case that $|M(P^+)| < |M(P)|$. □

Proposition 44 (Convergence of $T_{\mathcal{H}_{\inf}}$)**.** *The evolution of the One-to-Base Coverage Algorithm $(c(t), P(t))$ generated by the map $T_{\mathcal{H}_{\inf}}$ converges in finite time to the intersection of the equilibria of the maps $T_{\mathcal{H}_{\inf},i}$, that is, to a pair (c, P) where P is a Pareto-optimal partition generated by c.*

Proof. The proof follows the lines of the proof of Proposition 42, with the important modification of using a different Lyapunov function, defined as follows. Let ϵ_i and ϵ_m be the magnitude of the smallest possible non-zero difference between two values of $\mathcal{H}_{\inf}$ and $\mathcal{H}_{\min}$, respectively. Let α_m and α_M be larger than twice the maximum possible value of $\mathcal{H}_{\min}$ and $|M|$, respectively. If we define the function V as

$$V(c, P) = \mathcal{H}_{\inf}(c) + \frac{\epsilon_i}{\alpha_m}\mathcal{H}_{\min}(c, P) + \frac{\epsilon_i \epsilon_m}{\alpha_m \alpha_M}|M(P)|$$

and we invoke Proposition 43 and Lemma 39, we conclude that the dynamics converge to a fixed point (c^*, P^*).

It remains to show that P^* is a Pareto-optimal partition generated by c^*. If P^* is not a partition, then $P^*_{i,-} \neq 0$ and if the partition is not Voronoi then $P^*_{i,+} \neq 0$. Continuing by contradiction, assume that c^* forms a Voronoi partition which is not Pareto-optimal. This implies that there exists a $c' \in P^*$ where for at

least one agent $c'_i \neq c^*_i$ and a partition P' generated by c' such that $\mathcal{H}(c', P') < \mathcal{H}(c^*, P^*)$. By the definition of the algorithm and Proposition 37 this is not possible. Therefore the fixed point partition is Pareto-optimal. □

4.3 Implementation and Simulations

In order to efficiently implement the One-to-Base Coverage Algorithm and under the assumption of using $\mathcal{H}_{\min}$ or $\mathcal{H}_{\inf}$ as cost functions, we provide the following revised version, which can be easily seen to be equivalent to that in Section 4.2.2.

One-to-Base Coverage Algorithm – revised

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

1: update $\overline{P} := P$, $\overline{c} := c$, $\overline{\overline{c}} := c$, $\overline{P}_i = Q$
2: **for** $k \in P_i \backslash c$ **do**
3: update $\overline{c}_i := k$
4: **if** $U\big(\overline{c}, \overline{P}\big) < U\big(\overline{\overline{c}}, \overline{P}\big)$ **then**
5: update $\overline{\overline{c}}_i := k$
6: compute sets

$$P_{i,+} := \left\{ x \in Q \;\middle|\; \tfrac{1}{w_i} d_G(x, \overline{\overline{c}}_i) < \min\left\{ \tfrac{1}{w_j} d_G(x, c_j) \;\middle|\; x \in P_j,\ j \neq i \right\} \right\}$$

$P_{i,-} := \left\{ x \in P_i \cap \left(\cup_{i \neq j} P_j \right) \;\middle|\; \frac{1}{w_i} d_G(x, \overline{\overline{c}}_i) \geq \min \left\{ \frac{1}{w_j} d_G(x, c_j) \;\middle|\; x \in P_j,\ j \neq i \right\} \right\}$

7: $P_i^+ := (P_i \backslash P_{i,-}) \cup P_{i,+}$

8: $c_i^+ := \overline{\overline{c}}_i$

9: tell agent i to set $S_i^+ := P_i^+$ and $s_i^+ = c_i^+$

This revised version of the algorithm takes advantage of how the cost functions $\mathcal{H}_{\min}$ and $\mathcal{H}_{\inf}$, and sets $P_{i,+}$ and $P_{i,-}$ are defined. Indeed, setting $\overline{P}_i = Q$ in line 1 avoids having to calculate $P_{i,+}$ and $P_{i,-}$ for every $k \in P_i \backslash c_i$, as was done in Section 4.2.2, because $\mathcal{H}_{\min}$ and $\mathcal{H}_{\inf}$ already distribute costs to nodes that are closer to one agent as opposed to another. Hence this implementation produces the same evolutions but requires less memory, as we no longer need the set $\overline{\overline{P}}$, and less computation time, as the sets $P_{i,+}$ and $P_{i,-}$ are calculated only once.

We are now ready to proceed with our simulation results, which are obtained by running the revised version of the algorithm. To demonstrate the utility of the One-to-Base Coverage Algorithm for various values of cost function U, we implemented it using the open-source Player/Stage robot control system and the Boost Graph Library (BGL). All results presented here are generated using Player 2.1.1, Stage 2.1.1, and BGL 1.34.1. A non-convex environment (borrowed from [26]) is specified with three robots. The free space is modeled using an occupancy grid with $0.6m$ resolution, producing a lattice-like graph with all edge weights equal to $0.6m$. The $0.6m$ resolution is chosen so that each robot can fit in a grid cell.

One example with $U = \mathcal{H}_{\text{min,inf}}$ is shown in Figure 4.4. In the simulation, the robots have uniform weight assignment defined by $w_i = \frac{1}{3}$ for $i \in \{1, \ldots, 3\}$. We start with each robot owning the entire environment and stationed at its unique centroid as shown in the first panel, and then proceed by choosing a random robot to communicate with the base station at each iteration. The second panel shows an intermediate covering of the environment before convergence to a centroidal Voronoi partition. The third panel shows convergence of the $U = \mathcal{H}_{\text{min}}$ portion of the $U = \mathcal{H}_{\text{min,inf}}$ algorithm. The fourth panel shows the Pareto-optimal partition which is achieved after convergence of the $U = \mathcal{H}_{\text{inf}}$ portion of the $U = \mathcal{H}_{\text{min,inf}}$ algorithm. As can be seen, the movement of the robot relative to the third panel is marginal, but the partition appears to be more balanced and is still centroidal Voronoi. The cost to cover in terms of the multi-center function, $\mathcal{H}$, decreases from $\mathcal{H} = 729$ to $\mathcal{H} = 728$. Although the final partition and the decrease in cost-to-cover change only marginally in this example, the change can be much more noticeable as is explained in the following.

Another example with $U = \mathcal{H}_{\text{min,inf}}$ is shown in Figure 4.5. As before, the robot's have uniform weight assignment defined by $w_i = \frac{1}{3}$ for $i \in \{1, \ldots, 3\}$. The example starts with each agent owning the entire territory, the agents being stationed at their unique centroid, and the simulation continuing with the agents being selected at random to communicate with the base station. The second panel

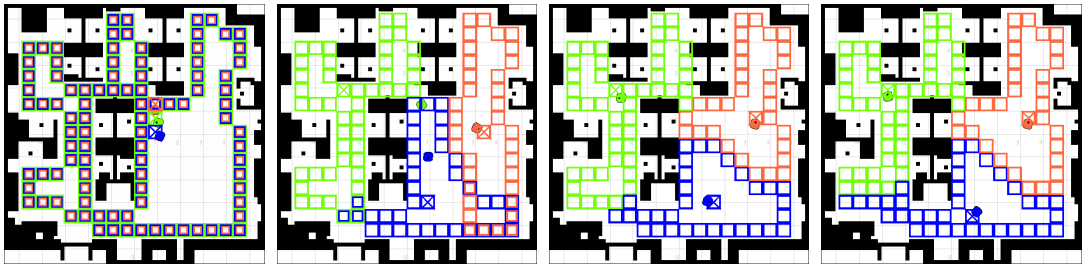

Figure 4.4: Snapshots from a simulation of three robots partitioning an environment with black obstacles using the $\mathcal{H}_{\text{min,inf}}$ One-to-base station algorithm. The free space of the environment is modeled using the indicated occupancy grid where each cell is a vertex in the resulting graph. The robots' optimal coverage position is marked by an X and the boundary of each robot's territory drawn in its color. Some cells are on the boundary of multiple territories: for these we draw superimposed robot colors.

shows the convergence of the $U = \mathcal{H}_{\text{min}}$ portion of the algorithm, which leads to a final multi-center cost of $\mathcal{H} = 804$. The third panel shows the update after the first iteration of the $U = \mathcal{H}_{\text{inf}}$ portion of the algorithm with the green agent. The update shows that the lower portion of the environment is getting less than optimal coverage and is improved by moving an agent closer to that region. The fourth panel shows the Pareto-optimal partition which is achieved after convergence of the $U = \mathcal{H}_{\text{inf}}$ portion of the $U = \mathcal{H}_{\text{min,inf}}$ algorithm. Notice that in this example, the final partition is quite different from the partition achieved at the end of the $U = \mathcal{H}_{\text{min}}$ portion of the algorithm. The final multi-center function cost of this partition is $\mathcal{H} = 753$, which is a noticeable improvement in coverage.

Thus far we have looked at two representative examples of using the algorithm with $U = \mathcal{H}_{\text{min,inf}}$. These examples illustrate that, like centroidal Voronoi

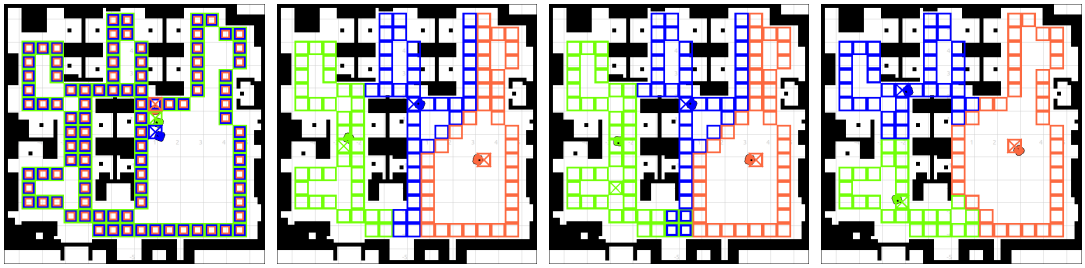

Figure 4.5: Snapshots from a simulation of three robots partitioning an environment with black obstacles using the $\mathcal{H}_{\text{min,inf}}$ One-to-base station algorithm. Note that the initial condition is the same as in Figure 4.4, but the evolution is different.

partitions, Pareto-optimal partitions are not necessarily unique, and that the evolution under the One-to-base station algorithm is only guaranteed to converge to a locally optimal solution. To see how the algorithm compares in general and for different choices of U, we simulate the algorithm with the same initial setup as shown in both Figure 4.4 and Figure 4.5. The One-to-Base Coverage Algorithm with $U = \mathcal{H}_{\text{min}}$, $U = \mathcal{H}_{\text{inf}}$, and $U = \mathcal{H}_{\text{min,inf}}$ is run 100 times for each choice of U. Table 4.1 summarizes the final cost-of-coverage for each choice of U. We observe that the One-to-Base Coverage Algorithm with $U = \mathcal{H}_{\text{min}}$ converges to partitions that have the same minimum cost as those attained with the algorithm using $U = \mathcal{H}_{\text{inf}}$ or $U = \mathcal{H}_{\text{min,inf}}$. On the other hand, the maximum cost-to-cover with $U = \mathcal{H}_{\text{min}}$ can be much larger than with the other two choices of U. Of the three algorithms, the algorithm with $U = \mathcal{H}_{\text{inf}}$ converges consistently to partitions with the lowest coverage costs, however, as discussed earlier it is computationally the

most expensive. Finally, the algorithm with $U = \mathcal{H}_{\text{min,inf}}$ behaves as expected, converging on average to partitions with values similar to that of the algorithm with $U = \mathcal{H}_{\text{inf}}$ although with a slightly larger deviation.

Algorithm	**Min**	**Mean**	**Max**	**StdDev**
$\mathcal{H}_{\text{min}}$	728	746.02	804	27.74
$\mathcal{H}_{\text{inf}}$	728	730.26	732	1.92
$\mathcal{H}_{\text{min,inf}}$	728	730.38	753	4.91

Table 4.1: Multi-center function cost-to-cover statistics for each algorithm from 100 simulation runs.

4.3.1 Handling Dynamic Changes

Evolving overlapping coverings in the One-to-Base Coverage Algorithm enables simple handling of environmental changes along with dynamic arrivals, departures, and even the disappearance of robots. Changes in the environment along with robot departures or disappearances can increase coverage cost, but those increases are only a transients and, with the appropriate algorithmic additions, the system will converge in finite steps after such an event. The One-to-Base Coverage Algorithm also has the added advantage that it can account for changes in robot performance due to changes in capability caused by potential damage to the hardware. The following algorithmic additions address how to handle the events described.

Environment Changes Each region in the environment is initially assigned an importance according to the weight function $\phi(x)$. As robots explore the environment, they may determine that certain regions are more/less important than what was originally assigned. Robots can communicate this to the base station at which point the base station can update $\phi(x)$.

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

Departure & Disappearance A robot i might announce to the base station that it is departing, perhaps to recharge its batteries or to perform some other task. In this situation, the base station can simply add P_i to the territory of the next robot it talks to before executing the normal steps of the algorithm. The disappearance or failure of a robot i can be detected if it does not communicate with the base station for longer than Δ. If this occurs, then the departure procedure above can be triggered. Should i reappear later, it can be handled as a new arrival or given its old territory.

Performance Malfunction of a robot i can be detected by the agent via self diagnosis and communicated to the base station. If this malfunction causes the robot to survey less territory, then w_i will have changed, so the base station can simply re-normalize the vector of weights w.

4.4 Summary

We have described a coverage algorithm, with corresponding cost-functions, which uses the One-to-Base station communication architecture that drive territory ownership among a team of robots in a non-convex environment to a centroidal Voronoi partition in finite time. We have also defined the notion of *Pareto-optimal partition* and have provided a provably correct method to reach such a partition using the One-to-Base Coverage Algorithm. Finally, we have demonstrated the effectiveness of the algorithm through simulation, and have outlined various ways the algorithm can be adapted to allow for dynamic changes in the system. We have focused on dividing territory in this work, but the algorithm can easily be combined with methods to provide a service over Q, as in [14].

In this chapter we looked at coverage control in a discrete time and space environment with unreliable asynchronous communication. In the next chapter

we look at what happens in a continuous time and space environment, and when there are constraints on the area owned by each agent.

Chapter 5

Partitioning with Area-Constraints

This chapter is organized as follows. In section 5.1 we setup preliminary notation, introduce the concept of generalized Voronoi partitions and present our problem in technical detail. In section 5.2 we compute some useful properties of the objective functions of interest. In section 5.3 we state existence properties of area-constrained generalized Voronoi partitions and present algorithms to reach the set of area-constrained Voronoi partitions. In section 5.4 we state the main result of our paper on centroidal equitable generalized Voronoi partitions. In section 5.6 we discuss the application of our algorithm to real systems and provide numerical simulations. In the final section we summarize our findings.

5.1 Preliminaries and problem statement

Let us have a convex compact set $Q \subset \mathbb{R}^2$, endowed with a density function $\phi : Q \to \mathbb{R}_{\geq 0}$, so that the measure (or area) of a region $A \subset Q$ is defined as

$$\phi(A) = \int_A \phi(q)dq,$$

provided the set A is measurable in the sense of Lebesgue. Without loss of generality, we assume that Q has unit measure, that is, $\phi(Q) = \int_Q \phi(q)dq = 1$. Let $p_1, \ldots, p_n$ denote the positions of n robotic agents in Q. We assume that each agent is associated with a (measurable) sub-region $W_i \subset Q$, where $\{W_i\}_{i=1}^n$ partitions Q into sets whose interiors are pairwise disjoint. A vector can be defined to collect the measures of the regions of a partition, as $\phi(W) = [\phi(W_1), \ldots, \phi(W_n)]^T$. By our assumptions on Q, we have $\sum_{i=1}^n \phi(W_i) = 1$. Let $f : \mathbb{R} \to \mathbb{R}$ be a strictly convex, increasing, and differentiable function. Then, given n locations $p = (p_1, \ldots, p_n)$ and a partition $W = (W_1, \ldots, W_n)$, the *multicenter function* is defined by

$$\mathcal{H}(p, W) = \sum_{i=1}^{n} \int_{W_i} f(\|q - p_i\|)\phi(q)dq.$$

Our goal in this work is minimizing the function $\mathcal{H}$ under certain constraints, namely, that the areas of each region are fixed. Specifically, we consider constants $c_i > 0$ for each agent $i \in \{1, \ldots, n\}$ such that $\sum_{i=1}^n c_i = 1$ and we require $\phi(W_i) = c_i$ for every i. For brevity, we denote $S = \{c \in \mathbb{R}^n_{>0} \mid \sum_{i=1}^n c_i = 1\}$.

Problem 7 (Multicenter optimization with area constraint)**.** *Given $c \in S$, determine the locations of the agents $p = (p_1, \ldots, p_n)$ and the partition $W = (W_1, \ldots, W_n)$ solving:*

$$\begin{aligned} \min_{p,W} \quad & \mathcal{H}(p, W) \\ \text{subject to} \quad & \phi(W_i) = c_i, \;\; i \in \{1, \ldots, n-1\}. \end{aligned} \tag{5.1}$$

Note that the nth constraint $\phi(W_n) = c_n$ is omitted because it is redundant.

In order to solve this problem, we introduce a useful partitioning scheme. To begin, we define $\mathcal{D} := \{p \in Q^n \mid p_i \neq p_j, \;\; i \neq j\}$ as the set of disjoint positions in Q. Then, given the function f as above, n distinct locations $p \in \mathcal{D}$, and n scalar weights $w = (w_1, \ldots, w_n)$, the *generalized Voronoi partition* of Q is the collection of subsets $V^f(p, w) = (V_1^f(p, w), \ldots, V_n^f(p, w))$ of Q, defined by

$$V_i^f(p, w) = \{q \in Q \mid f(\|q - p_i\|) - w_i \leq f(\|q - p_j\|) - w_j, \;\; \forall j \neq i\}. \tag{5.2}$$

Generalized Voronoi partitions enjoy several important properties. First, any generalized Voronoi partition of Q is in fact a partition of Q. Second, the generalized partition generated by (p, w) is equal to the generalized partition generated by $(p, w + \alpha \mathbb{1}_n)$, for any $\alpha \in \mathbb{R}$ (Here, $\mathbb{1}_n$ is the vector in $\mathbb{R}^n$ whose entries are all equal to 1). Finally, as opposed to Voronoi partitions, for generalized Voronoi partitions we do not require that $p_i \in V_i^f(p, w)$. Thus, agents need not be located in their partition. From here onward, we will refer to generalized Voronoi par-

titions simply as *Voronoi partitions*. Two important special cases are described below.

Example 2 (Standard Voronoi Diagram)**.** *Given $p \in \mathcal{D}$ and n scalar weights $w = (w_1, \ldots, w_n)$, the* Standard Voronoi Diagram *of Q is given by* (5.2) *with $w = 0$. The partition is given by*

$$V_i^{SD}(p, w) = \{q \in Q \mid f(\|q - p_i\|) \leq f(\|q - p_j\|)\},$$

regardless of the choice of $f(x)$. We call each V_i^{SD} a standard *Voronoi region. These regions are convex and have boundaries that are given by straight line segments; moreover, every generator p_i is contained in its respective region V_i^{SD}.*

Example 3 (Power Diagrams)**.** *Given $p \in \mathcal{D}$ and n scalar weights $w = (w_1, \ldots, w_n)$, the* power diagram *of Q is given by* (5.2) *with $f(x) = x^2$. The partition is given by*

$$V_i^{PD}(p, w) = \{q \in Q \mid \|q - p_i\|^2 - w_i \leq \|q - p_j\|^2 - w_j\},$$

and we call each Voronoi region V_i^{PD} a power cell. Note that Standard Voronoi Diagrams are a special case of Power Diagrams, since $V_i^{PD}(p, 0) = V^{SD}(p)$. These regions are convex and their boundaries that are line segments; however, it is possible that the generators p_i are not contained by their respective power cells V_i^{PD}.

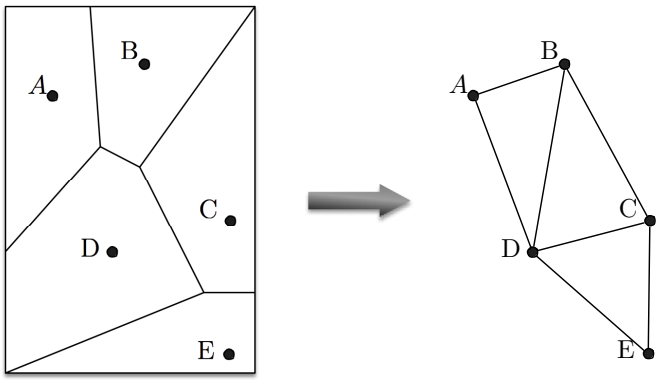

Figure 5.1: The image on the left is the Standard Voronoi Partition generated by nodes A through E. The image on the right shows the dual graph for this partition.

We now define the dual graph of a partition $\{W_i\}_{i=1}^n$, which will be useful later: the node set is $\{1, \ldots, n\}$, and there exists an edge $\{i, j\}$ if the boundary between agents i and j, denoted $\Delta_{i,j}$, has positive measure. In that case, we say that j is a neighbor of i and we write $j \in N_i$. The dual graph of the standard Voronoi Partition is known to be the classic Dirichlet triangulation, as illustrated in Figure 5.1.

Before we are ready to define a second problem of interest, we first introduce a useful definition. Given n distinct locations $p = (p_1, \ldots, p_n)$, the set of weights $w = (w_1, \ldots, w_n)$ such that every partition has non-zero measure is defined by $U = \{w \in \mathbb{R}^n \mid \phi(V_i^f(p, w)) > 0 \ \forall i\}$. Since $V^f(p, w) = V^f(p, w + \alpha \mathbb{1}_n)$ for every scalar α, this equivalence relation naturally defines equivalence classes in U: with

a slight abuse of notation, in what follows we sometimes refer to such classes as the elements of U. The following problem is a simplified version of Problem 7.

Problem 8 (Multicenter Voronoi partition optimization with area constraints)**.** *Given $c \in S$, determine the locations of the agents $p \in \mathcal{D}$ and weights $w \in U$ solving:*

$$\begin{aligned} \min_{p,w} \quad & \mathcal{H}(p, V^f(p,w)) \\ \text{subject to} \quad & \phi(V_i^f(p,w)) = c_i, \;\; i \in \{1, \dots, n-1\}. \end{aligned}$$

As a preliminary step, we should make sure that this problem has feasible solutions: this fact is shown in Section 5.3, which also provides a method to find a set of weights in U for every set of locations. Problems 7 and 8 are known to be equivalent in the following sense.

Proposition 45 (Proposition V.1 in [23])**.** *Let $p \in Q^n$ be the agent locations and $w \in U$ a weight assignment which satisfies the area constraint. Then, the Voronoi partition $V^f(p,w)$ optimizes $\mathcal{H}(p,W)$ among all partitions satisfying the area constraint.*

In order to derive a useful consequence of this fact, we consider the simpler case when there is only one agent in Q: then, the multicenter function becomes

$$p \mapsto \mathcal{H}_1\big(p, Q\big) := \int_Q f(\|q-p\|)\phi(q)dq. \tag{5.3}$$

Since f is strictly convex, $\mathcal{H}_1$ is too, and the following holds: If Q is convex, then there is a unique minimizer of (5.3), which we denote by $\mathrm{Ce}(Q)$. Moreover, we have that $\frac{\partial \mathcal{H}_1(p,Q)}{\partial p} = 0$ at $p = \mathrm{Ce}(Q)$, where $\frac{\partial \mathcal{H}_1(p,Q)}{\partial p} = \int_Q \frac{\partial}{\partial p} f(\|q-p\|)\phi(q)dq$ from [13]. The Voronoi partition $V^f(p,w)$ generated by (p,w) is said to be *centroidal* if

$$\mathrm{Ce}[V_i^f(p,w)] = p_i,$$

for all $i \in \{1,\dots,n\}$. This notation allows us to state the following fact [23]: for every solution (p^*, W^*) of Problem 7, there exists a weight assignment $w^* \in U$ such that $W^* = V^f(p^*, w^*)$ and $p_i^* = \mathrm{Ce}(W_i^*)$ for all $i \in \{1,\dots,n\}$. Equivalently, the solutions to Problem 7 are centroidal Voronoi partitions whose regions have the prescribed areas.

In the rest of this paper, we will go beyond this abstract characterization of the optimal solutions and give an optimization algorithm which is amenable to practical implementation.

5.2 Relevant partial derivatives

This section is devoted to compute relevant partial derivatives of the multicenter function, which shall be used to solve Problem 8 in the subsequent sections.

Given $p \in \mathcal{D}$, and $w \in U$, we define the partition of Q by $V^f(p,w)$. It is convenient to define the *Voronoi multi-center function* as

$$(p,w) \mapsto \overline{\mathcal{H}}(p,w) = \sum_{i=1}^{n} \int_{V_i^f(p,w)} f(\|q-p_i\|)\phi(q)dq,$$

or equivalently $\overline{\mathcal{H}}(p,w) = \mathcal{H}(p,V^f(p,w))$. It is also convenient to define the *generators-to-areas function* as

$$(p,w) \mapsto \mathcal{M}(p,w) = \left[\int_{V_1^f(p,w)} \phi(q)dq, \ldots, \int_{V_n^f(p,w)} \phi(q)dq\right]^T,$$

or equivalently $\mathcal{M}(p,w) = \phi(V^f(p,w))$. In the rest of this section, we shall compute the gradients of the functions $\overline{\mathcal{H}}$ and $\mathcal{M}$. In order to state our results, we need some notation. Let $\Delta_{i,j}(p,w)$ denote the boundary between the ith and jth Voronoi region and $\vec{n}_{i,j}$ the normal to this boundary, pointing towards region W_j. Given locations $p \in \mathcal{D}$ and weights $w \in U$, let $L_a(p,w)$ and $L_{b_k}(p,w)$, $k \in \{1,2\}$, be the $n \times n$ matrices, whose entries $a_{i,j}$ and $b_{i,j}^{(k)}$ are defined by

$$a_{i,j}(p,w) = \begin{cases} -\int_{\Delta_{i,j}} \phi(q)\left(\frac{\partial q}{\partial w_i} \cdot \vec{n}_{i,j}\right)dq, & \text{if } i \neq j, \\ -\sum_{i=1}^{n} a_{i,j}, & \text{otherwise,} \end{cases} \tag{5.4}$$

and

$$b_{i,j}^{(k)}(p,w) = \begin{cases} -\int_{\Delta_{i,j}} \phi(q)\left(\frac{\partial q}{\partial p_i^{(k)}} \cdot \vec{n}_{i,j}\right)dq, & \text{if } i \neq j, \\ -\sum_{i=1}^{n} b_{i,j}^{(k)}, & \text{otherwise,} \end{cases} \tag{5.5}$$

where $q \in \mathbb{R}^2$ has components $q^{(1)}$ and $q^{(2)}$. Clearly, entries $a_{i,j}$ and $b_{i,j}$ are zero if $\Delta_{i,j}$ has zero measure.

We are now ready to state the two main results of this section, which imply that computing the gradients of $\overline{\mathcal{H}}(p,w)$ and the $\mathcal{M}(p,w)$ is spatially-distributed over the dual graph of the Voronoi partition.

Proposition 46 (Partial derivatives of the Voronoi multi-center function)**.** *Given $p \in \mathcal{D}$ and $w \in U$, let $p_i^{(k)}$, $k \in \{1,2\}$, denote the two components of $p_i \in \mathbb{R}^2$, for $i \in \{1,\dots,n\}$, and define L_a and L_{b_k} as in equations* (5.4) *and* (5.5)*. Then, the partial derivatives of $\overline{\mathcal{H}}(p,w)$ are*

$$\frac{\partial \overline{\mathcal{H}}(p,w)}{\partial p^{(k)}} = \Bigg[\int_{V_i^f} \frac{\partial}{\partial p_1^{(k)}} f(\|q-p_1\|)\phi(q)dq, \dots, \int_{V_n^f} \frac{\partial}{\partial p_n^{(k)}} f(\|q-p_n\|)\phi(q)dq \Bigg] + w^T L_{b_k}(p,w), \tag{5.6}$$

$$\frac{\partial \overline{\mathcal{H}}(p,w)}{\partial w} = w^T L_a(p,w). \tag{5.7}$$

Proof. Note that we write V for V^f throughout the proof and everything is done with respect to one component of p_i where $p_i \in \mathbb{R}^2$ (we drop the k from $p_i^{(k)}$ for clarity). Differentiating with respect to p_i we see that

$$\begin{aligned}\frac{\partial \overline{\mathcal{H}}}{\partial p_i} = &\int_{V_i} \frac{\partial}{\partial p_i} f(\|q-p_i\|)\phi(q)dq \\ &+ \int_{\partial V_i} f(\|q-p_i\|)\phi(q) \left(\frac{\partial q}{\partial p_i} \cdot \vec{n}_{i,j}\right) dq \\ &+ \sum_{j \in N_i} \int_{\partial V_i \cap \partial V_j} f(\|q-p_j\|)\phi(q) \left(\frac{\partial q}{\partial p_i} \cdot \vec{n}_{j,i}\right) dq,\end{aligned} \tag{5.8}$$

where (5.8) easily falls out from the conservation law [13, Proposition 2.23]. The second term on the RHS is defined as

$$\begin{aligned}
&\int_{\partial V_i} f(\|q-p_i\|)\phi(q)\left(\frac{\partial q}{\partial p_i}\cdot n\right)dq \\
&\qquad = \sum_{j\in N_i}\int_{\Delta_{i,j}} f(\|q-p_i\|)\phi(q)\left(\frac{\partial q}{\partial p_i}\cdot \vec{n}_{i,j}\right)dq \\
&\qquad = \sum_{j\in N_i}\int_{\Delta_{i,j}} (f(\|q-p_j\|)+w_i-w_j)\phi(q)\left(\frac{\partial q}{\partial p_i}\cdot \vec{n}_{i,j}\right)dq,
\end{aligned}$$

where $f(\|q-p_i\|) = f(\|q-p_j\|) + w_i - w_j$ holds true along the boundary $\Delta_{i,j}$ and is given from the definition of the Voronoi partition. Therefore, combining the second and third terms on the RHS of (5.8) and noting that $\vec{n}_{i,j} = -\vec{n}_{j,i}$, we have

$$\frac{\partial \overline{\mathcal{H}}}{\partial p_i} = \int_{V_i}\frac{\partial}{\partial p_i} f(\|q-p_i\|)\phi(q)dq + \sum_{j\in N_i}(w_i-w_j)\int_{\Delta_{i,j}}\phi(q)\left(\frac{\partial q}{\partial p_i}\cdot\vec{n}_{i,j}\right)dq. \quad (5.9)$$

Writing (5.9) in vector form gives $\frac{\partial \overline{\mathcal{H}}(p,w)}{\partial p}$ as defined in (5.6), with matrices whose entries are defined by (5.5). Similarly the derivative with respect to w_i is given as

$$\frac{\partial \overline{\mathcal{H}}}{\partial w_i} = \sum_{j\in N_i}(w_i-w_j)\int_{\Delta_{i,j}}\phi(q)\left(\frac{\partial q}{\partial w_i}\cdot\vec{n}_{i,j}\right)dq. \quad (5.10)$$

The vector form of this equation can easily be seen to be (5.7) with the Laplacian matrix L_a defined by (5.4). □

Proposition 47 (Partial derivatives of the generators-to-areas function)**.** *Given $p \in \mathcal{D}$ and $w \in U$, let $p_i^{(k)}$, $k \in \{1,2\}$, denote the two components of $p_i \in \mathbb{R}^2$, for*

$i \in \{1, \ldots, n\}$, and define L_a and L_{b_k} as in equations (5.4) *and* (5.5)*. Then, the partial derivatives of $\mathcal{M}(p, w)$ are*

$$\frac{\partial \mathcal{M}(p,w)}{\partial p^{(k)}} = L_{b_k}(p, w), \tag{5.11}$$

$$\frac{\partial \mathcal{M}(p,w)}{\partial w} = L_a(p, w). \tag{5.12}$$

Proof. For clarity, we write V for V^f throughout the proof. Looking at the i^{th} component of $\mathcal{M}(p, w)$ and differentiating with respect to w_i, by the conservation law [13, Proposition 2.23] we get

$$\begin{aligned}\frac{\partial \mathcal{M}_i}{\partial w_i} &= \int_{V_i} \frac{\partial}{\partial w_i}\phi(q)dq + \int_{\partial V_i} \phi(q)\left(\frac{\partial q}{\partial w_i}\cdot \vec{n}_{i,j}\right)dq, \\ &= \sum_{j\in N_i}\int_{\Delta_{i,j}} \phi(q)\left(\frac{\partial q}{\partial w_i}\cdot \vec{n}_{i,j}\right)dq, \end{aligned} \tag{5.13}$$

where $\Delta_{i,j}$ is the boundary between agents i and j. The first term on the right hand side of (5.13) is zero since the density function is not dependent on the weights. Similarly, the derivative of $\mathcal{M}_j$ with respect to w_i, is given as

$$\begin{aligned}\frac{\partial \mathcal{M}_j}{\partial w_i} &= \int_{V_j} \frac{\partial}{\partial w_i}\phi(q)dq + \int_{\partial V_j} \phi(q)\left(\frac{\partial q}{\partial w_i}\cdot \vec{n}_{j,i}\right)dq, \\ &= -\int_{\Delta_{i,j}} \phi(q)\left(\frac{\partial q}{\partial w_i}\cdot \vec{n}_{i,j}\right)dq, \end{aligned} \tag{5.14}$$

where we note that $\vec{n}_{i,j} = -\vec{n}_{j,i}$. Therefore it is easily seen that the total gradient of $\mathcal{G}$ with respect to the vector of weights w is given by (5.12), with $L_a(p, w)$ defined by (5.4). Similarly for $p_i^{(k)}$ we get the gradient defined by (5.11) whose matrix $L_{b_k}(p, w)$ is given by (5.5). □

The following useful Proposition follows from Proposition 47 and Proposition IV.1 in [23] and is due to the monotonicity properties of the Voronoi partition. Indeed, any increase in the weights of agent i (keeping the weights of other agents fixed) guarantees that the area of agent i increases and the areas of the neighboring agents decrease.

Proposition 48 (Sign-definiteness of the partial derivatives of $\mathcal{M}$)**.** *Given $p \in \mathcal{D}$ and $w \in U$, then*

$$\frac{\partial \mathcal{M}_i(p,w)}{\partial w_i} > 0 \;\; \textit{and} \;\; \frac{\partial \mathcal{M}_i(p,w)}{\partial w_j} \leq 0, \;\; j \neq i,$$

where the second inequality is strict if and only if $\Delta_{i,j}$ has non-zero measure.

Next, consider the dual graph of the Voronoi partition defined by (p, w) and assign to each edge $\{i, j\}$ of this graph the (i, j) entry of $L_a(p, w)$, which is strictly-negative by Proposition 48. With this definition, the matrix $L_a(p, w)$ is the Laplacian matrix naturally associated to this weighted dual graph of the Voronoi partition defined by (p, w).

5.3 Area-constrained Voronoi partitions

In this section, we solve the problem of finding, given area constraints and locations of the agents, suitable weights such that the Voronoi partition generated

by these locations and weights satisfies the area constraint. We begin by stating two useful results.

Proposition 49 (Existence and uniqueness of weights for area-constrained Voronoi partitions)**.** *Define constants $c \in S$. Given $p \in \mathcal{D}$, there exists a unique vector $w^* \in U$, up to translation, such that $\{V_1^f(p, w^*), \ldots, V_n^f(p, w^*)\}$ satisfies $\phi(V_i^f) = c_i$ for all i.*

Proof. Existence follows from Proposition IV.4 in [23]. Before beginning the uniqueness argument, we introduce the set $V_{i \to j}(w^*, w)$ as the set of points that move from agent i to agent j due to a change in weights from w^* to w. For example, if agents i and j are neighbors, $w_j > w_j^*$, and $w_i = w_i^*$ for all $i \neq j$, then $\phi(V_{i \to j}(w^*, w)) > 0$. That is to say that some region is transferred to agent j due to its weight increasing. With this notation in mind, we begin the proof with a set of weights, $w^* \in U$, which define some arbitrary partition. Associated with this partition are a set of non-zero areas, $c_i^* > 0$ for $i \in \{1, \ldots, n\}$, that correspond to each agents region. Assume there exists another set of weights, $w \neq w^*$, such that $\mathcal{M}(p, w) = \mathcal{M}(p, w^*) = c^*$. Since weights are translation invariant, we can translate w such that $w_1^* - w_1 = 0$. Without loss of generality (wlog), assume that generator 2 is the neighbor of generator 1 in $V^f(p, w^*)$, and that $w_2 - w_1 < w_2^* - w_1^*$ ($w_2 < w_2^*$, since $w \neq w^*$) or equivalently $w_2 - w_2^* < w_1 - w_1^* = 0$. Due to the monotonicity of the weights, $w_2 < w_2^*$ implies that part of the region that was

once owned by agent 2 in $V^f(p, w^*)$ is either now owned by agent 1 or by some other agent in the partition $V^f(p, w)$. Thus $\phi(V_{2\to i}(w^*, w)) > 0$ for some $i \neq 2$. This means that agent 2 in $V^f(p, w)$ must own the space of some other agent (or combination of agents) in order to maintain its area-constraint. This can only happen if there exists at least one neighboring agent (wlog; agent 3) whose weight satisfies the condition $w_3 - w_2 < w_3^* - w_2^*$. Thus it must be the case that $w_3 - w_3^* < w_2 - w_2^* < w_1 - w_1^*$ and $w_3 < w_3^*$. Since every agent has at least one neighbor, we can continue in this fashion of ordering agents, until we reach agent k such that $w_j - w_j^* < w_k - w_k^*$, where $j > k$, cannot be satisfied, and so $\phi(V_k^f(p, w)) < c_k^*$. Thus, the set of weights that satisfy the area-constraint for a generalized Voronoi partition are unique, up to translation. □

Given this result, for any given set of area constraints $c \in S$, we may formally define the map $w_{\text{ac}} : \mathcal{D} \to U$ as $p \mapsto w_{\text{ac}}(p)$, such that $\sum_{i=1}^n (w_{\text{ac}}(p))_i = 0$ and $\phi(V_i^f(p, w_{\text{ac}}(p))) = c_i$ for all i.

Proposition 50 (Smoothness of mapping from positions to weights)**.** *The map $p \mapsto w_{\text{ac}}(p)$ is continuously differentiable.*

Proof. The proof makes use of the implicit function theorem in conjunction with a modified mapping of $\mathcal{M}(p, w)$, to be described later, to show continuous differentiability of $w_{\text{ac}}(p)$. Let $c \in S$ denote the vector of areas for the constraint surface.

Then given the mapping $w_{\text{ac}}(p)$, we have that $\mathcal{M}(p, w_{\text{ac}}(p)) = c$. Since Voronoi partitions are translation invariant with respect to weights, w, we can define any Voronoi partition as a function of $n-1$ weights, keeping the n^{th} weight constant at zero. Define $\tilde{w} \equiv [w_1, \ldots, w_{n-1}]^T$, and define the modified mapping $(p, \tilde{w}) \mapsto \tilde{\mathcal{M}}(p, \tilde{w}) \in \mathbb{R}^{n-1}$ of $\mathcal{M}(p, w)$ by $\tilde{\mathcal{M}}(p, \tilde{w}) = \left[\int_{V_1^f(p,\tilde{w})} \phi(q)dq, \ldots, \int_{V_{n-1}^f(p,\tilde{w})} \phi(q)dq\right]^T$. Define the mapping $p \mapsto \tilde{w}_{\text{ac}}(p)$ such that $\tilde{\mathcal{M}}(p, \tilde{w}_{\text{ac}}(p)) = \tilde{c}$, where we set $\tilde{c}_i = c_i$ for $i \in \{1, \ldots, n-1\}$. Note that this is sufficient to define the constraint surface since the n^{th} constraint in $\mathcal{M}(p, w)$ and the n^{th} weight w_n are both redundant. For clarity, calculations in the rest of the proof are done with respect to one component of p_i, where $p_i \in \mathbb{R}^2$. Differentiating $\tilde{\mathcal{M}}(p, \tilde{w}) = \tilde{c}$ with respect to $\tilde{w}$ and p we obtain

$$\frac{\partial \tilde{\mathcal{M}}(p, \tilde{w})}{\partial \tilde{w}} = \tilde{L}_a(p, \tilde{w}),$$
$$\frac{\partial \tilde{\mathcal{M}}(p, \tilde{w})}{\partial p} = \tilde{L}_b(p, \tilde{w}),$$

where $\tilde{L}_b(p, \tilde{w})$ is defined as $L_b(p, w)$ with the n^{th} row removed and $\tilde{L}_a(p, \tilde{w})$ is defined as the Laplacian matrix $L_a(p, w)$ with the n^{th} column and row removed (Note $L_b(p, w)$ is dependent on which component of p_i we choose). Proposition 48 guarantees that the Laplacian of the Voronoi dual graph is always well-defined, therefore $\tilde{L}_a(p, \tilde{w}_{\text{ac}}(p)) \in \mathbb{R}^{n-1 \times n-1}$ and is full rank [40, Corollary 6.2.27]. Therefore, since $\tilde{\mathcal{M}}(p, \tilde{w})$ is continuously differentiable and $\frac{\partial \tilde{\mathcal{M}}(p, \tilde{w})}{\partial \tilde{w}}$ is invertible, then by

the implicit function theorem we have that $\tilde{w}_{\mathrm{ac}}(p)$ is continuously differentiable. Finally, note that the mapping $w_{\mathrm{ac}}(p)$ can be written as $[\tilde{w}_{\mathrm{ac}}(p)^T, w_{\mathrm{ac},n}(p)]^T$, where $w_{\mathrm{ac},n}(p)$ is a constant value of zero. Since $\tilde{w}_{\mathrm{ac}}(p)$ and $w_{\mathrm{ac},n}(p)$ are continuously differentiable, so is $w_{\mathrm{ac}}(p)$. □

We now present an algorithm to compute $w_{\mathrm{ac}}(p)$ for a specific area-constraint. Given $p \in \mathcal{D}$ and $w \in U$, the *area-constraint cost function* for the Voronoi partition generated by (p, w) is defined as

$$\begin{aligned}\mathcal{J}(p,w) &= n \log\left(\int_Q \phi(q)dq\right) - \sum_{i=1}^{n} c_i \log\left(\int_{V_i^f(p,w)} \phi(q)dq\right) \\ &= n\log\left(\phi(Q)\right) - \sum_{i=1}^{n} c_i \log\left(\phi\big(V_i^f(p,w)\big)\right) \end{aligned} \tag{5.15}$$

where c_i for $i \in \{1, \ldots, n\}$ are strictly positive constants. Note that $n \log(\phi(Q)) = 0$ when $\phi(Q) = 1$. The following result extends [71, Theorem 3.7] to (generalized) Voronoi partitions, and has the added property of being better conditioned numerically.

Theorem 51 (Gradient of the area-constraint cost function)**.** *Let $\Delta_{i,j}$ denote the boundary between the i^{th} and j^{th} Voronoi region and $\vec{n}_{i,j}$ the normal vector along that boundary. Define constants $c \in S$. Given $p \in \mathcal{D}$ and $w \in U$, we have*

$$\frac{\partial}{\partial w_i}\mathcal{J}\big(p,w\big) = \sum_{j \in N_i} \Xi_{i,j}\big(c_j\phi(V_i^f) - c_i\phi(V_j^f)\big), \tag{5.16}$$

where

$$\Xi_{i,j} = \frac{1}{\phi(V_i^f)\phi(V_j^f)} \int_{\Delta_{i,j}} \left(\frac{\partial q}{\partial w_i} \cdot \vec{n}_{i,j} \right) \phi(q) dq, \tag{5.17}$$

so that

(i) every w generating an area-constrained Voronoi partition is a critical point of the function $w \mapsto \mathcal{J}(p, w)$, and

(ii) every solution to the negative gradient flow

$$\dot{w}_i = -\frac{\partial}{\partial w_i} \mathcal{J}(p, w), \tag{5.18}$$

converges asymptotically to $w_{\mathrm{ac}}(p)$, yielding an area-constrained Voronoi partition such that $\phi(V_i^f) = c_i$.

Proof. Let $w \mapsto \mathcal{J}(p, w)$ be a candidate Lyapunov function. First, we check that $\mathcal{J}$ is continuously differentiable. Using (5.13) and (5.14) from the proof of Proposition 47 and the chain rule, we quickly have (5.16) with coefficients defined by (5.17). Therefore given $p \in \mathcal{D}$, we have that $\mathcal{J}$ is continuously differentiable with respect to the weights $w \in U$. Second, we see with $\dot{w}_i$ defined according to (5.18) that $\dot{\mathcal{J}} = \sum_{i=1}^n \frac{\partial \mathcal{J}}{\partial w_i} \dot{w}_i \leq 0$. To determine when $\dot{\mathcal{J}}$ is identically zero, let $y_i = \frac{1}{\phi\left(V_i^f(p,w)\right)}$, and rewrite (5.16) and (5.17) in vector notation to get the following:

$$\frac{\partial}{\partial w} \mathcal{J}(p, w) = y^T \mathrm{diag}\left([c_1, \ldots, c_n]\right) L_a(p, w),$$

where $L_a(p, w)$ is the Laplacian matrix defined by (5.4). Let $x = y^T \text{diag}\left([c_1, \ldots, c_n]\right)$, then $\frac{\partial}{\partial w}\mathcal{J}\big(p, w\big)$ is identically zero when $x = \alpha \mathbb{1}_n^T$ for any constant α. Given the constraints of the system this can only happen when $y_i = \frac{1}{c_i}$, or equivalently when $\phi\left(V_i^f(p, w)\right) = c_i$ for all $i \in \{1, \ldots, n\}$. Thus the invariant set for (5.18) is such that for $\phi\left(V_i^f(p, w)\right) = c_i$ all i and $\sum_{i=1}^n c_i = 1$. By these observations, we have proved claim (i). Third, we have to show that trajectories are bounded. From the gradient descent law (5.18) we deduce that the measures of each agent are bounded away from zero. Indeed, if the measure of an agent's region were to approach zero, then the function $\mathcal{J}$ would grow unbounded; this is impossible because we know $\mathcal{J}$ is monotonically non-increasing. Notice that the measures of the agents depend on the weights, and it is not hard to verify that the sum of weights stays constant. Hence, if a weight were to become very large, another weight would become arbitrarily small, which would cause a region to vanish. This contradicts the fact that the measures are bounded: therefore, the weights must also be bounded. After these three observations, we can invoke LaSalle Invariance Principle and deduce that the weights converge to the set of weights w such that $\phi(V_i^f(p, w)) = c_i$ for all i. Additionally, since the sum of the weights is constant and the vector of weights that satisfy the area constraint is unique by Proposition 49, we conclude that the weights converge to the vector of weights w^* such that $\sum_{i=1}^n w_i^* = \sum_{i=1}^n w_i(0)$ and $\phi(V_i^f(p, w^*)) = c_i$ for all i, proving claim (ii). $\square$

We now specialize the gradient (5.18) to the case of Example 3.

Example 4 (Gradient flow for equitable power diagrams)**.** *Define the partition of the region Q according to* (5.2) *with $f(x) = x^2$ and let $c_i = \frac{1}{n}$ for all i in* (5.15)*. From [71] we have that $\frac{\partial q}{\partial w_i} \cdot \vec{n}_{i,j} = \frac{1}{\|p_i - p_j\|}$ so then the gradient flow* (5.18) *is given by*

$$\dot{w}_i = -\sum_{j \in N_i} \left(\frac{1}{\phi(V_j^{PD})} - \frac{1}{\phi(V_i^{PD})} \right) \int_{\Delta_{i,j}} \frac{\phi(q)}{\|p_i - p_j\|} dq.$$

The vector of weights w converges to the value such that $\phi\left(V_i^{PD}(p,w)\right) = \phi\left(V_j^{PD}(p,w)\right)$ for all $i \neq j$.

5.4 Centroidal area-constrained Voronoi partitions

Given constants $c \in S$, and given $p \in \mathcal{D}$, it is convenient to define the *area-constrained Voronoi partition* generated by p as

$$V_{\mathrm{ac}}^f(p) = V^f(p, w_{\mathrm{ac}}(p)),$$

with Voronoi regions $V_{\mathrm{ac},i}^f(p)$, $i \in \{1, \ldots, n\}$ such that $\phi\left(V_{\mathrm{ac},i}^f(p)\right) = c_i$ for all i. The associated *area-constrained multicenter function* is given by

$$p \mapsto \mathcal{H}\big(p, V_{\mathrm{ac}}^f(p)\big) = \sum_{i=1}^{n} \int_{V_{\mathrm{ac},i}^f(p)} f(\|q - p_i\|)\phi(q) dq,$$

or equivalently by $p \mapsto \overline{\mathcal{H}}\big(p, w_{\mathrm{ac}}(p)\big)$. We are now ready for the main result of this section.

Theorem 52 (Gradient of the area-constrained multicenter function)**.** *Given* $p \in \mathcal{D}$*,*

$$\frac{\partial}{\partial p_i}\mathcal{H}\big(p, V^f_{\mathrm{ac}}(p)\big) = \int_{V^f_{\mathrm{ac},i}(p)} \frac{\partial}{\partial p_i} f(\|q - p_i\|)\phi(q)dq, \tag{5.19}$$

so that

(i) every p *generating a centroidal area-constrained Voronoi partition is a critical point of the function* $p \mapsto \mathcal{H}\big(p, V^f_{\mathrm{ac}}(p)\big)$*, and*

(ii) every solution to the negative gradient flow

$$\dot{p}_i = -\int_{V^f_{\mathrm{ac},i}(p)} \frac{\partial}{\partial p_i} f(\|q - p_i\|)\phi(q)dq \tag{5.20}$$

converges asymptotically to the set of centroidal area-constrained Voronoi partitions.

Proof. Let $\overline{\mathcal{H}}(p, w_{\mathrm{ac}}(p)) = \mathcal{H}(p, V^f(p, w_{\mathrm{ac}}(p)))$, the Voronoi multicenter function restricted to the area-constraint surface, be our candidate Lyapunov function. By differentiating with respect to p we obtain

$$\begin{aligned}\frac{\partial}{\partial p}\overline{\mathcal{H}}(p, w_{\mathrm{ac}}) &= \frac{\partial \overline{\mathcal{H}}(p, w_{\mathrm{ac}})}{\partial p} + \frac{\partial \overline{\mathcal{H}}(p, w_{\mathrm{ac}})}{\partial w}\frac{dw_{\mathrm{ac}}}{dp}, \\ &= \left(\left[\frac{\partial \mathcal{H}_1(p_1, V^f_{\mathrm{ac},1}(p))}{\partial p_1}, \ldots, \frac{\partial \mathcal{H}_1(p_n, V^f_{\mathrm{ac},n}(p))}{\partial p_n}\right]\right. \\ &\qquad \left. + w^T L_b(p, w)\right) + w^T L_a(p, w)\frac{dw_{\mathrm{ac}}}{dp},\end{aligned} \tag{5.21}$$

where $L_a(p,w)$ and $L_b(p,w)$ are defined by (5.4) and (5.5), respectively. Differentiating $\mathcal{M}(p, w_{\text{ac}}(p)) = c$ with respect to p we obtain

$$\frac{\partial \mathcal{M}(p, w_{\text{ac}}(p))}{\partial p} + \frac{\partial \mathcal{M}(p, w_{\text{ac}}(p))}{\partial w} \frac{dw_{\text{ac}}(p)}{dp} = 0,$$

that is, $L_b(p,w) + L_a(p,w)\frac{dw_{\text{ac}}(p)}{dp} = 0$, therefore $L_a(p,w)\frac{dw_{\text{ac}}(p)}{dp} = -L_b(p,w)$. Substituting this equality into (5.21) it follows that $\frac{\partial}{\partial p}\mathcal{H}(p, V^f(p, w_{\text{ac}})) = \frac{\partial}{\partial p}\overline{\mathcal{H}}(p, w_{\text{ac}}) = \left[\frac{\partial \mathcal{H}_1(p_1, V^f_{\text{ac},1}(p))}{\partial p_1}, \ldots, \frac{\partial \mathcal{H}_1(p_n, V^f_{\text{ac},n}(p))}{\partial p_n}\right]$. Therefore $\mathcal{H}(p, V^f(p, w_{\text{ac}}))$ is continuously differentiable with respect to p, and its critical points are characterized, proving claim (i). For each agent the trajectories under (5.20) point towards the centers of their region, and since Q is convex and compact this gives that the trajectories stay in Q and are bounded. We must also show that the agents maintain distinct locations. If two agents i and j have the same weight ($w_j = w_i$), then relative to each other they generate a Standard Voronoi partition (Example 2) and the center for each agent stays in the same region as the agent, therefore the agents can not collide. Now we look at the case when the weights are different. Without loss of generality let $w_j > w_i$ and assume that agent j approaches agent i. From (5.2) we have that for $q \in Q$, the region of agent j satisfies $f(\|q - p_j\|) - w_j + w_i \leq f(\|q - p_i\|)$. If p_j and p_i are close enough, all points in a neighborhood of p_i belong to region j. This implies that the measure of region i is zero; this is impossible since the flow stays along the area-constraint surface. Therefore, agents can not collide and agent locations remain distinct along the flow. Under control law (5.20) we have

that

$$\begin{aligned}\dot{\mathcal{H}}(p, V^f(p, w_{\text{ac}})) &= \sum_{i=1}^{n} \frac{\partial \mathcal{H}_1(p_i, V^f_{\text{ac},i}(p))}{\partial p_i} \dot{p}_i \\ &= -\sum_{i=1}^{n} \left(\frac{\partial \mathcal{H}_1(p_i, V^f_{\text{ac},i}(p))}{\partial p_i} \right)^2 .\end{aligned}$$

By LaSalle's Invariance Principle the positions p converge to the invariant set of positions such that $p_i = Ce[V^f_{\text{ac},i}(p)]$ for all i. Therefore, the positions converge to the set of centroidal area-constrained Voronoi partitions and claim (ii) is proved.

□

There are some interesting points worth noting. First, assuming that the set of centroidal area-constrained Voronoi partitions is finite, the positions and weights converge to one of the partitions in that set. Second, the gradient descent (5.20) is not guaranteed to find the global minimum. Finally, the gradient restricted to the constraint surface is formally the same as the *reduced gradient* as defined in nonlinear programming [57].

We now specialize the gradient (5.19) to the case of Example 3.

Example 5 (Constrained gradient flow for power diagrams)**.** *Let the partition of the region Q be defined according to* (5.2) *with $f(x) = x^2$. Given any* powercell *A of Q the center is given by $Ce[A] = \frac{1}{\phi(A)} \int_A q\phi(q)dq$. Thus for* power diagrams*, Ce is equivalent to the well known expression for* center of mass *of a region.*

From [13] we have that $\frac{\partial \mathcal{H}_1\left(p, V_{\mathrm{eq},i}^{PD}(p)\right)}{\partial p_i} = \frac{1}{n}\left(Ce[V_{\mathrm{ac},i}^{PD}] - p_i\right)$*, therefore the (scaled, negative) gradient flow* (5.20) *is given by*

$$\dot{p}_i = Ce[V_{\mathrm{ac},i}^{PD}(p)] - p_i.$$

We easily see that $\mathcal{H}(p, V_{\mathrm{ac}}^{PD})$ *is minimized when* $p_i = Ce[V_{\mathrm{ac},i}^{PD}(p)]$ *for all* $i \in \{1, \ldots, n\}$.

5.5 Simultaneous change of agent positions and weights

The previous gradient descent laws (5.18) and (5.20), are designed to find the area-constraint surface and reach the center of a area-constrained region, respectively. In this section, we introduce a distributed algorithm which achieves both tasks simultaneously. As before, for the desired constraint surface we choose constants $c \in S$. Then, given $p \in \mathcal{D}$ and $w \in U$, the *simultaneous gradient* algorithm is given by

$$\begin{aligned} \dot{p}_i &= -\int_{V_i^f(p,w)} \frac{\partial}{\partial p_i} f(\|q - p_i\|)\phi(q)dq, \\ \dot{w}_i &= -\sum_{j \in N_i} \Xi_{i,j}\left(c_j\phi(V_i^f) - c_i\phi(V_j^f)\right), \end{aligned} \tag{5.22}$$

where

$$\Xi_{i,j} = \frac{1}{\phi(V_i^f)\phi(V_j^f)} \int_{\Delta_{i,j}} \left(\frac{\partial q}{\partial w_i} \cdot \vec{n}_{i,j} \right) \phi(q) dq.$$

The proposed control law is a natural combination of laws (5.18) and (5.20). If all the weights are initialized to the same value $\overline{w}$, and if $\dot{w}_i$ is set to zero, then (5.22) reduces to the continuous-time Lloyd algorithm presented in [13]. Simulations show that the proposed law does in fact converge to the set of centroidal area-constrained Voronoi partitions; however, a proof is not currently available.

We now specialize the gradient (5.22) to the case of Example 3.

Example 6 (Gradient flow for equitable power diagrams)**.** *Define the partition of the region Q according to* (5.2) *with $f(x) = x^2$ and let $c_i = \frac{1}{n}$ for all i in* (5.15)*. From [13] we have that $\frac{\partial \mathcal{H}_1\left(p, V_i^{PD}(p)\right)}{\partial p_i} = \frac{1}{n}\left(Ce[V_i^{PD}] - p_i\right)$, therefore the (scaled, negative) gradient flow* (5.22) *is given by*

$$\begin{aligned} \dot{p}_i &= Ce[V_i^{PD}(p)] - p_i, \\ \dot{w}_i &= -\sum_{j \in N_i} \left(\frac{1}{\phi(V_j^{PD})} - \frac{1}{\phi(V_i^{PD})} \right) \int_{\Delta_{i,j}} \frac{\phi(q)}{\|p_i - p_j\|} dq. \end{aligned}$$

We easily see that the stationary set for this system is achieved when $p_i = Ce[V_{\mathrm{ac},i}^{PD}(p)]$ for all $i \in \{1, \ldots, n\}$ and the vector of weights w converges to the value such that $\phi\left(V_i^{PD}(p,w)\right) = \phi\left(V_j^{PD}(p,w)\right)$ for all $i \neq j$.

5.6 Implementation and simulations

In this section, we discuss the practical implementation of the control law (5.20) and compare it against the control law (5.22) using representative simulation examples. Writing the area-constrained gradient flow (5.20), we assume that we always stay on the constraint surface and thus move along this surface continuously. In order to put this law into practice, we would need an explicit formula to instantaneously compute the weights of the current Voronoi partition, as functions of the generator locations. Since such a formula is not available, we instead have to rely on system (5.18) in order to determine the weights. We then design an implementation which alternates dynamics (5.20) and (5.18). Assuming that the agents start at a feasible configuration, they move their locations according to (5.20) for a small time duration δ, while keeping the weights fixed. After this amount of time, the area constraint is not satisfied: we then let the weights evolve according to (5.18), while the locations are fixed, until we are within the proximity of the area-constraint surface.

Some points in this implementation require attention. First, if the positions are allowed to move too much, the regions can become undefined (i.e., have measure zero), therefore care must be taken to make sure this does not happen by selecting a sufficiently small δ. Second, care must also be taken to insure that the agent

location do not collide (i.e., agent locations must remain distinct). If they do, the step size δ should be reduced in order to avoid collision. Third, in order to drive the system exactly back to the constraint surface, we would need to bring the dynamics of the weights to convergence, which would take an infinitely long time: in simulations, convergence is approximated up to truncation error. In spite of these difficulties, in our simulations we have found that the algorithm is not sensitive to how far the agents deviate from the area-constraint surface (provided measures stay non-zero) during each movement step: in all our experiments, the algorithm converges to a centroidal area-constrained Voronoi diagram.

An illustrative example of the performance of the algorithm is presented in Figure 5.2. In the simulation, 10 agents have been randomly placed in a square region Q, where the density function $\phi(x)$ is constant. The region Q is to be partitioned according to (5.2) with $f(x) = x^2$, that is, as a power diagram. We define area-constraint surface such that $\phi(V_i^f) = \frac{i}{\sum_{i=1}^{n} i}\phi(Q)$ for all $i \in \{1, \ldots, 10\}$; which means that if $i < j$, then $\phi(V_i^f) < \phi(V_j^f)$. The gradient (5.20) is followed during steps of duration $\delta = 0.1s$. The first panel shows the initial condition of the system at $T = 0$, where each agent has been randomly placed and the corresponding weights which generate the area-constrained partition determined. In the second panel, at $T = \delta$, the agents have moved in the gradient direction which causes each region to have a different area (the agents moved off the constraint

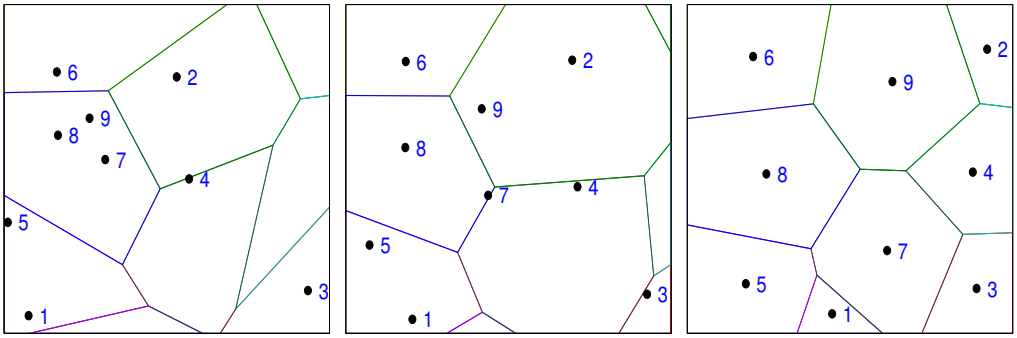

Figure 5.2: Simulation of 9 agents partitioning a square environment with uniform density using the iterative gradient algorithm.

surface of equitable area). The last panel shows the final state of the system at $T = 54.2s$. Each agent is at its region's centroid and the regions have the desired measure.

Next we observe the performance of control law (5.22), which simultaneously optimizes the weights and the positions. We set the initial weights of the system to zero, but we keep the same initial positions and area-constraint requirements described above. The first panel of Figure 5.3 shows that we do in fact start with the same initial positions as the previous control algorithm, and since the weights are zero, the partition is the Standard Voronoi Diagram; the second panel shows how the position trajectories evolve over time, and the final panel shows the system's final configuration. Figure 5.4 shows better how the areas of each agent's region evolves over time; the second panel shows how each agent's position converges to its regions centroid. It is interesting to observe that given the same

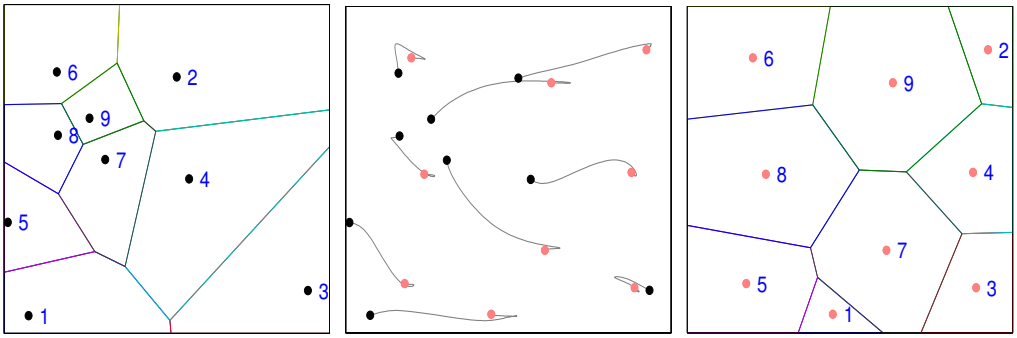

Figure 5.3: Simulation of 9 agents partitioning a square environment with uniform density using the simultaneous gradient algorithm.

initial conditions, the iterative algorithm and control law (5.22) converge to the same configuration. In fact, we did not come across a case where this did not happen.

5.7 Summary

In this chapter we studied the problem of how to optimally deploy a set of agents over a convex workspace while each agent maintains a pre-specified area. We have designed a provably correct, spatially-distributed continuous-time policy that solves this optimization problem. We proposed a method for implementation of the control policy and demonstrated its effectiveness in simulation.

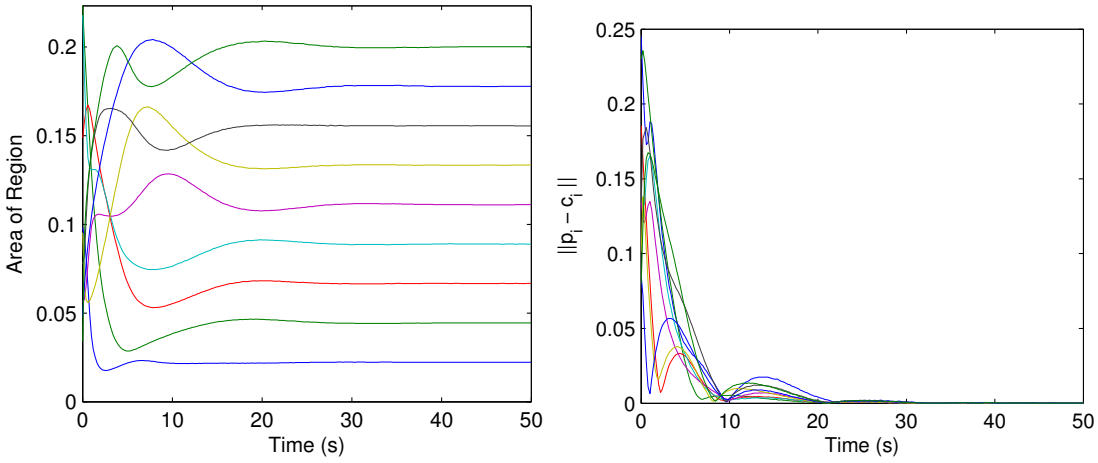

Figure 5.4: Area (left) and position (right) trajectories for 9 agents partitioning a square environment with uniform density using the simultaneous gradient algorithm.

Chapter 6

Conclusion and Future Work

Coordination in multi-agent networks is becoming more relevant as tasks become increasingly complex and need to be distributed. There are a myriad of applications for such networks, each with their own complex and interesting problems. In this thesis we have tackled two important problems in particular.

First, we looked at the problem of surveillance. In particular, we looked at stochastic surveillance, which has applications when the strategy has *unpredictability* or predefined visit frequency requirements; in many surveillance problems there are *important* regions that needed to be visited more frequently than others. We showed how to formulate such problems and how to optimally find stochastic surveillance strategies.

Second, we studied the problem of deployment and territory partitioning with a mobile robotic network. This problem has many variations which depend on the type of communication law used and constraints applied. We looked at two

cases. We began by looking at the case where there is sporadic, asynchronous communication between agents in the network and a central base-station. This sees applications when continuous line of sight communication is not readily possible; underwater vehicles and spacecraft are two examples of this. Then, we looked at the case when there is continuous communication with neighboring agents and there is a constraint on the amount of coverage each agent must provide. This arises in applications where workload needs to be distributed in a predefined way, such as distributing workload amongst mobile cell towers.

6.1 Summary

We started in Chapter 2 by formally formulating the stochastic surveillance problem over a graph. We modeled our surveillance strategy as a Markov chain described by a transition matrix and showed that minimizing the hitting time of that Markov chain gives the optimal strategy for a certain class of problems. Using the hitting time of the Markov chain, we showed for reversible transition matrices that the hitting time can be formulated as an SDP, and thus a globally optimal solution, if it exists, can always be found. Then, we extended the notion of hitting time to graphs with travel distances and showed, for reversible transition matrices, that the weighted hitting time can also be formulated as a SDP. Since

the weighted hitting time accounts for travel distances, it proved to be far superior than other reversible Markov chain based surveillance strategies.

In Chapter 3 we further developed the notion of hitting time, and were able to extend hitting time to multiple random walkers, coining it the *group hitting time*. As a consequence, our results also allowed us to determine the hitting time between any set of nodes, a quantity that could previously only be calculated for (1) a single random walker with (2) a very specialized structure. These results also allowed us to extend the weighted hitting time by finding analytic expressions for the pairwise hitting times for a graph with travel distances, however, this was only possible for the single random walker case. Finally, we provided extensive simulations showing the random walks that generate optimal group hitting times can have a predictable structure for some graph topologies, and highly unpredictable structure for others.

In Chapter 4 we started looking into problems in coverage control. We considered a discrete environment and modeled the environment by a graph as was done in the previous two chapters. In this chapter we defined the one-to-base station communication protocol and provided an algorithm that, using the one-to-base protocol, converged to centroidal Voronoi partitions. We also introduced the notion of Pareto-optimal partitions and provided a provable method to converge to those partitions. In addition, we discussed various real-world application cases, and how to handle each using the algorithms presented.

Finally, in Chapter 5 we studied the area-constrained coverage problem over a convex workspace. Using generalized Voronoi partitions, we showed that when each agent can talk to its territory neighbor, that we can design a provably correct, spatially-distributed continuous-time policy that converges to the set of centroidal area-constrained partitions.

6.2 Future Work

This work leaves open various directions for further research.

Hitting time For the single agent weighted hitting time in Chapter 2, it was only optimized over the transition matrix. It would be interesting to see the implications of optimizing both the weight matrix and transition matrix simultaneously. This can have the interpretation of optimizing the "capacity" or "resistance" of the graph, a topic in optimization which is of independent interest [34].

For the multiple Markov chains scenario, a clear extension is to consider the case of heterogeneous travel times similar to what was done in Chapter 2. In addition, although we provide a method for determining the hitting time of multiple random walkers, in general, it can be difficult to compute when the size of the graph and number of agents increases. It would be of practical interest to find a formulation which is computationally less expensive or a method in which the

group hitting time can be calculated in a distributed way. On a related note, we found in Chapter 3 that the minimal group hitting time is linked to the underlying graph topology of each random walker. It would be of interest to find a way to incorporate spatial partitioning algorithms, such as those in Chapter 4 or [26], to find graph partitions/coverings that generate lower hitting times. Finally, given the maximum *pairwise* hitting time of a Markov chain, there exists bounds on the cover time for multiple copies of that Markov chain running in parallel [27, 3]. It would be interesting to see if our results can be leveraged to extend those bounds to multiple heterogenous Markov chains running in parallel.

Coverage Control For the one-to-base algorithm it would be worthwhile to adapt the algorithm to allow for area-constrained partitions similar to the work done in Chapter 5. Also, we would like to extend the One-to-base Coverage Algorithm to other communication settings (e.g., directional or pair-wise gossip) to take advantage of the notion of Pareto-optimal partitions.

For area-constrained partitions (Chapter 5) our main approach is based on alternating phases, during which we either improve the objective function, or enforce the area constraint. In contrast to our main solution, we would like to prove convergence of the proposed policy in which agents converge to the constraint surface while simultaneously optimizing the coverage problem. So far, the

effectiveness of this policy has been observed in simulations. Second, our policy requires synchronous and reliable communication along the edges of the dual graph associated to the Voronoi partition. It would be of practical interest to relax this requirement, using asynchronous, event-based, or unreliable communication as in Chapter 4 or [12, 62]. In addition, our approach is based on the assumption that the environment we partition is convex, and finding policies that work over non-convex environments would be of great practical use: works in this direction include [16] and [26]. Finally, we assume that the density function ϕ is known to the agents, which may be hard to satisfy in practice; several papers have recently appeared to overcome this assumption, including [63] and [74].

Bibliography

[1] A. Albert. Conditions for positive and nonnegative definiteness in terms of pseudoinverses. *SIAM Journal on Applied Mathematics*, 17(2):434–440, 1969.

[2] D. Aldous and J. A. Fill. Reversible Markov Chains and Random Walks on Graphs, 2002. Unfinished monograph, recompiled 2014, available at `http://www.stat.berkeley.edu/~aldous/RWG/book.html`.

[3] N. Alon, C. Avin, M. Koucky, G. Kozma, Z. Lotker, and M. R. Tuttle. Many random walks are faster than one. *Combinatorics, Probability and Computing*, 20, 7 2011.

[4] D. A. Anisi, P. Ögren, and X. Hu. Cooperative minimum time surveillance with multiple ground vehicles. *IEEE Transactions on Automatic Control*, 55(12):2679–2691, 2010.

[5] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. The role of social networks in information diffusion. In *Int. Conference on World Wide Web*, pages 519–528, Lyon, France, 2012.

[6] S. Bhattacharya, R. Ghrist, and V. Kumar. Multi-robot coverage and exploration in non-Euclidean metric spaces. In *Algorithmic Foundations of Robotics X*, volume 86, pages 245–262. Springer, 2013.

[7] T. H. Blackwell and J. S. Kaufman. Response time effectiveness: Comparison of response time and survival in an urban emergency medical services system. *Academic Emergency Medicine*, 9(4):288–295, 2002.

[8] B. Bošanský, V. Lisý, M. Jakob, and M. Pěchouček. Computing time-dependent policies for patrolling games with mobile targets. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 989–996, 2011.

[9] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. *SIAM Review*, 46(4):667–689, 2004.

[10] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[11] A. Z. Broder and A. R. Karlin. Bounds on the cover time. *Journal of Theoretical Probability*, 2(1):101–120, 1989.

[12] F. Bullo, R. Carli, and P. Frasca. Gossip coverage control for robotic networks: Dynamical systems on the space of partitions. *SIAM Journal on Control and Optimization*, 50(1):419–447, 2012.

[13] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Princeton University Press, 2009.

[14] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99(9):1482–1504, 2011.

[15] P. Burgisser, M. Clausen, and A. Shokrollahi. *Algebraic Complexity Theory*. Springer, 1997.

[16] J. G. Carlsson, E. Carlsson, and R. Devulapalli. Balancing workloads for service vehicles over a geographic territory. *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, Nov. 2013. To appear.

[17] J. G. Carlsson and R. Devulapalli. Shadow prices in territory division. University of Minnesota. Available online at `http://menet.umn.edu/~jgc/`, 2013.

[18] M. Catral, S. J. Kirkland, M. Neumann, and N.-S. Sze. The Kemeny constant for finite homogeneous ergodic Markov chains. *Journal of Scientific Computing*, 45(1-3):151–166, 2010.

[19] L. Chen and J. Leneutre. A game theoretical framework on intrusion detection in heterogeneous networks. *IEEE Transactions on Information Forensics and Security*, 4(2):165–178, June 2009.

[20] J. Clark and R. Fierro. Mobile robotic sensors for perimeter detection and tracking. *ISA Transactions*, 46(1):3–13, 2007.

[21] C. Cooper, A. Frieze, and T. Radzik. Multiple random walks and interacting particle systems. In S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. Nikoletseas, and W. Thomas, editors, *Automata, Languages and Programming*,

volume 5556 of *Lecture Notes in Computer Science*, pages 399–410. Springer, 2009.

[22] C. Cooper, A. M. Frieze, and T. Radzik. Multiple random walks in random regular graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1738–1761, 2009.

[23] J. Cortés. Coverage optimization and spatial load balancing by robotic sensor networks. *IEEE Transactions on Automatic Control*, 55(3):749–754, 2010.

[24] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[25] P. G. Doyle and J. L. Snell. *Random Walks and Electric Networks*. Mathematical Association of America, 1984.

[26] J. W. Durham, R. Carli, P. Frasca, and F. Bullo. Discrete partitioning and coverage control for gossiping robots. *IEEE Transactions on Robotics*, 28(2):364–378, 2012.

[27] K. Efremenko and O. Reingold. How well do random walks parallelize? In I. Dinur, K. Jansen, J. Naor, and J. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 5687 of *Lecture Notes in Computer Science*, pages 476–489. Springer, 2009.

[28] W. Ellens, F. M. Spieksma, P. V. Mieghem, A. Jamakovic, and R. E. Kooij. Effective graph resistance. *Linear Algebra and its Applications*, 435(10):2491–2506, 2011.

[29] Y. Elmaliach, N. Agmon, and G. A. Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57(3-4):293–320, 2009.

[30] Y. Elmaliach, A. Shiloni, and G. A. Kaminka. A realistic model of frequency-based multi-robot polyline patrolling. In *International Conference on Autonomous Agents*, pages 63–70, Estoril, Portugal, May 2008.

[31] R. Elsässer and T. Sauerwald. Tight bounds for the cover time of multiple random walks. *Theoretical Computer Science*, 412(24):2623–2641, 2011.

[32] U. Feige. A tight upper bound on the cover time for random walks on graphs. *Random Structures & Algorithms*, 6(1):51–54, 1995.

[33] E. Fiorelli, N. E. Leonard, P. Bhatta, D. A. Paley, R. Bachmayer, and D. M. Fratantoni. Multi-AUV control and adaptive sampling in Monterey Bay. *IEEE Journal of Oceanic Engineering*, 31(4):935–948, 2006.

[34] A. Ghosh, S. Boyd, and A. Saberi. Minimizing effective resistance of a graph. *SIAM Review*, 50(1):37–66, 2008.

[35] B. Golden, S. Raghavan, and E. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*. Springer, 2008.

[36] J. Grace and J. Baillieul. Stochastic strategies for autonomous robotic surveillance. In *IEEE Conf. on Decision and Control and European Control Conference*, pages 2200–2205, Seville, Spain, Dec. 2005.

[37] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, Oct. 2014.

[38] Z. Han, A. L. Swindlehurst, and K. J. R. Liu. Smart deployment/movement of unmanned air vehicle to improve connectivity in MANET. In *IEEE Wireless Communications and Networking Conference*, pages 252 –257, Apr. 2006.

[39] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[40] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[41] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1994.

[42] J. J. Hunter. Generalized inverses and their application to applied probability problems. *Linear Algebra and its Applications*, 45:157–198, 1982.

[43] J. J. Hunter. *Mathematical Techniques of Applied Probability*, volume 1 of *Discrete Time Models: Basic Theory*. Academic Press, 1983.

[44] J. J. Hunter. *Mathematical Techniques of Applied Probability*, volume 2 of *Discrete Time Models: Techniques and Applications*. Academic Press, 1983.

[45] J. J. Hunter. The role of Kemeny's constant in properties of Markov chains. *Communications in Statistics - Theory and Methods*, 43(7):1309–1321, 2014.

[46] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[47] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer, 1976.

[48] D. B. Kingston, R. W. Beard, and R. S. Holt. Decentralized perimeter surveillance using a team of UAVs. *IEEE Transactions on Robotics*, 24(6):1394–1404, 2008.

[49] S. Kirkland. Fastest expected time to mixing for a Markov chain on a directed graph. *Linear Algebra and its Applications*, 433(11-12):1988–1996, 2010.

[50] D. J. Klein and M. Randić. Resistance distance. *Journal of Mathematical Chemistry*, 12(1):81–95, 1993.

[51] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using Kronecker multiplication. In *Knowledge Discovery in Databases (PKDD)*, pages 133–145. Springer, 2005.

[52] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using Kronecker multiplication. In *Int. Conference on Machine Learning*, pages 497–504, Corvalis, OR, USA, 2007.

[53] M. Levene and G. Loizou. Kemeny's constant and the random surfer. *The American Mathematical Monthly*, 109(8):741–745, 2002.

[54] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

[55] Y. Y. Li and L. E. Parker. Intruder detection using a wireless sensor network with an intelligent mobile robot response. In *IEEE SoutheastCon*, pages 37–42, Huntsville, AL, USA, Apr. 2008.

[56] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. Presented at the 1957 Institute for Mathematical Statistics Meeting.

[57] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 2 edition, 1984.

[58] A. Macwan, G. Nejat, and B. Benhabib. Optimal deployment of robotic teams for autonomous wilderness search and rescue. In *IEEE/RSJ Int. Conf.*

on Intelligent Robots & Systems, pages 4544–4549, San Francisco, CA, USA, Sept. 2011.

[59] M. Mahdian and Y. Xu. Stochastic Kronecker graphs. *Random Structures & Algorithms*, 38(4):453–466, 2011.

[60] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, 2001.

[61] Y. Nonaka, H. Ono, K. Sadakane, and M. Yamashita. The hitting and cover times of Metropolis walks. *Theoretical Computer Science*, 411(16):1889–1894, 2010.

[62] C. Nowzari and J. Cortés. Self-triggered coordination of robotic networks for optimal deployment. *Automatica*, 48(6):1077–1087, 2012.

[63] J. L. Ny and G. J. Pappas. Adaptive deployment of mobile robotic networks. *IEEE Transactions on Automatic Control*, 58(3):654–666, 2013.

[64] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2 edition, 2000.

[65] J. L. Palacios. Closed-form formulas for Kirchhoff index. *International Journal of Quantum Chemistry*, 81(2):135–140, 2001.

[66] J. L. Palacios. On hitting times of random walks on trees. *Statistics & Probability Letters*, 79(2):234–236, 2009.

[67] J. L. Palacios and P. Tetali. A note on expected hitting times for birth and death chains. *Statistics & Probability Letters*, 30(2):119–125, 1996.

[68] F. Pasqualetti, A. Franchi, and F. Bullo. On cooperative patrolling: Optimal trajectories, complexity analysis and approximation algorithms. *IEEE Transactions on Robotics*, 28(3):592–606, 2012.

[69] A. Patcha and J. Park. A game theoretic approach to modeling intrusion detection in mobile ad hoc networks. In *Information Assurance Workshop. Proceedings from the Fifth Annual IEEE SMC*, pages 280–284, June 2004.

[70] R. Patel, P. Agharkar, and F. Bullo. Robotic surveillance and Markov chains with minimal first passage time. *IEEE Transactions on Automatic Control*, May 2014. To appear.

[71] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo. Distributed algorithms for environment partitioning in mobile robotic networks. *IEEE Transactions on Automatic Control*, 56(8):1834–1848, 2011.

[72] A. Pereira, H. Heidarsson, C. Oberg, D. Caron, B. Jones, and G. Sukhatme. A communication framework for cost-effective operation of AUVs in coastal regions. In A. Howard, K. Iagnemma, and A. Kelly, editors, *Field and Service Robotics*, volume 62 of *Tracts in Advanced Robotics*, pages 433–442. Springer, 2010.

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

[74] M. Schwager, D. Rus, and J. J. Slotine. Decentralized, adaptive coverage control for networked robots. *International Journal of Robotics Research*, 28(3):357–375, 2009.

[75] S. D. Servetto and G. Barrenechea. Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 12–21. ACM, 2002.

[76] C. Seshadhri, A. Pinar, and T. G. Kolda. An in-depth analysis of stochastic Kronecker graphs. *Journal of the Association for Computing Machinery*, 60(2):13:1–13:32, 2013.

[77] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: modeling and analysis of a three-tier architecture for sparse sensor networks. *Ad Hoc Networks*, 1(2-3):215–233, 2003.

[78] R. N. Smith, Y. Chao, P. P. Li, D. A. Caron, B. H. Jones, and G. S. Sukhatme. Planning and implementing trajectories for autonomous underwater vehicles to track evolving ocean processes based on predictions from a regional ocean model. *International Journal of Robotics Research*, 29(12):1475–1497, 2010.

[79] K. Srivastava, D. M. Stipanovič, and M. W. Spong. On a stochastic robotic surveillance problem. In *IEEE Conf. on Decision and Control*, pages 8567–8574, Shanghai, China, Dec. 2009.

[80] V. Srivastava, F. Pasqualetti, and F. Bullo. Stochastic surveillance strategies for spatial quickest detection. *International Journal of Robotics Research*, 32(12):1438–1458, 2013.

[81] S. Susca, S. Martínez, and F. Bullo. Monitoring environmental boundaries with a robotic sensor network. *IEEE Transactions on Control Systems Technology*, 16(2):288–296, 2008.

[82] P. Tetali. Random walks and the effective resistance of networks. *Journal of Theoretical Probability*, 4(1):101–109, 1991.

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

[84] P. M. Weichsel. The Kronecker product of graphs. *Proceedings of the American Mathematical Society*, 13(1):47–52, 1962.

[85] X. Wu and Z. Liu. How community structure influences epidemic spread in social networks. *Physica A: Statistical Mechanics and its Applications*, 387(2):623–630, 2008.

[86] P. R. Wurman, R. D'Andrea, and M. Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9–20, 2008.