Multi-robot Coverage and Redeployment Algorithms

by

Armin Sadeghi Yengejeh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2020

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Jorge Cortés Professor, Dept. of Mechanical and Aerospace Engineering, University of California, San Diego
Supervisor:	Stephen L. Smith Associate Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Christopher Nielsen Associate Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Mahesh Tripunitara Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Baris Fidan Professor, Dept. of Mechanical and Mechatronics Engineering, University of Waterloo

Abstract

In this thesis, we focus on two classes of multi-robot task allocation and deployment problems motivated by applications in ride-sourcing transportation networks and service robots: 1) coverage control with multiple robots, and 2) robots servicing tasks arriving sequentially over time.

The first problem considers the deployment of multiple robots to cover a domain. The multi-robot problem consists of multiple robots with sensors on-board observing the spatially distributed events in an environment. The objective is to maximize the sensing quality of the events via optimally distributing the robots in the environment. This problem has been studied extensively in the literature and several algorithms have been proposed for different variants of this problem. However, there has been a lack of theoretical results on the quality of the solutions provided by these algorithms. In this thesis, we provide a new distributed multi-robot coverage algorithm with theoretical guarantees on the solution quality, run-time complexity, and communication complexity. The theoretical bound on the solution quality holds for on-board sensors where the sensing quality of the sensors is a sub-additive function of the distance to the event location in convex and non-convex environments.

A natural extension of the multi-robot coverage control problem is considered in this thesis where each robot is equipped with a set of different sensors and observes different event types in the environment. Servicing a task in this problem corresponds to sensing an event occurring at a particular location and does not involve visiting the task location. Each event type has a different distribution over the domain. The robots are heterogeneous in that each robot is capable of sensing a subset of the event types. The objective is to deploy the robots into the domain to maximize the total coverage of the multiple event types. We propose a new formulation for the heterogeneous coverage problem. We provide a simple distributed algorithm to maximize the coverage. Then, we extend the result to the case where the event distribution is unknown before the deployment and provide a distributed algorithm and prove the convergence of the approach to a locally optimal solution.

The third problem considers the deployment of a set of autonomous robots to efficiently service tasks that arrive sequentially in an environment over time. Each task is serviced when a robot visits the corresponding task location. Robots can then redeploy while waiting for the next task to arrive. The objective is to redeploy the robots taking into account the next N task arrivals. We seek to minimize a linear combination of the expected cost to service tasks and the redeployment cost between task arrivals. In the single robot case, we propose a one-stage greedy algorithm and prove its optimality. For multiple robots, the problem is NP-hard, and we propose two constant-factor approximation algorithms, one for the problem with a horizon of two task arrivals and the other for the infinite horizon when the redeployment cost is weighted more heavily than the service cost.

Finally, we extend the second problem to scenarios where the robots are self-interested service units maximizing their payoff. The payoff of a robot is a linear combination of its relocation cost and its expected revenue from servicing the tasks in its vicinity. In this extension, the global objective is either to minimize the expected time or minimize the maximum time to respond to the tasks. We introduce two indirect control methods to relocate the self-interested service units: 1) an information sharing method, and 2) a method that incentivizes relocation with payments. We prove NP-hardness of finding the optimal controls and provide algorithms to find the near-optimal control. We quantify the performance of the proposed algorithms with analytical upper-bounds and real-world data from ride-sourcing applications.

Acknowledgements

I would like to thank my supervisor Professor Stephen Smith and express my appreciation for his support, patience, motivation, and guidance throughout the Ph.D. program.

I would like to thank the readers of this thesis – Professor Jorge Cortés, Professor Christopher Nielsen, Professor Mahesh Tripunitara, and Professor Baris Fidan – for their valuable time.

Finally, I would like to thank my family for their love and support.

Dedication

This dissertation is dedicated to my mother, Effat Moslemi. Thank you for your support and encouragement.

Table of Contents

List of Tables **xii**

List of Figures **xiii**

1 Introduction **1**

1.1 Literature Review . . . 3

1.2 Contribution and Organization . . . 5

2 Preliminaries **8**

2.1 Graphs . . . 8

2.2 Complexity Theory . . . 9

2.3 Reductions and Complexity Classes . . . 10

2.4 Combinatorial problems . . . 11

2.4.1 Minimum Weight Bipartite Matching . . . 11

2.4.2 Facility Location Problem . . . 11

2.4.3 k-median Problem . . . 12

2.4.4 k-center Problem . . . 13

2.4.5 CNF-SAT . . . 13

3 Homogeneous Multi-Robot Coverage **14**

3.1 Related Work . . . 15

3.2 Continuous and Discrete Coverage Problems 16
3.2.1 Continuous Environment 16
3.2.2 Discrete Environment 17
3.2.3 From a Continuous to a Discrete Problem 17
3.3 Distributed Algorithm On Graphs 20
3.3.1 High-level Idea 21
3.3.2 Detailed Description 23
3.4 Analysis of the Algorithm 26
3.4.1 Correctness and Approximation Factor 26
3.4.2 Time Complexity 30
3.5 Simulation Results 31
3.5.1 Convex Environments 32
3.5.2 Non-Convex Environments 34
3.5.3 Examples of Different Local Solutions 35

4 Heterogeneous Multi-Robot Coverage **39**
4.1 Related Work 40
4.2 Problem Formulation 41
4.2.1 Continuous Environments 41
4.2.2 Discrete Environments 42
4.3 Gradient Descent Algorithm for Continuous Environments 44
4.4 Gradient Descent Algorithm for Discrete Environments 47
4.5 Experimental Results 49
4.5.1 Continuous Environment 49
4.5.2 Discrete Environment 54

5 Re-Deployment of Multiple Service Robots 56

5.1 Related Work . . . 57

5.2 Problem Formulation . . . 58

5.3 Optimal Policy for Single Robot . . . 60

5.4 Multiple Robots . . . 61

5.4.1 Bounding the Value Function . . . 62

5.4.2 Policy I: Approximation Algorithm for TWO-STAGE-HORIZON . . . 65

5.4.3 Policy II: Move to k-median . . . 70

5.5 Simulation Results . . . 71

6 Re-Balancing Self-Interested Drivers in Transportation Networks 77

6.1 Related Work . . . 79

6.2 Problem Formulation . . . 80

6.2.1 Environment Model . . . 81

6.2.2 Self-interested Drivers' Model . . . 81

6.2.3 Service Provider's Model . . . 82

6.3 Control by Sharing Information . . . 83

6.3.1 Formal Definition and Complexity Class . . . 83

6.3.2 Minimizing The Expected Wait Time . . . 86

6.3.3 Minimizing The Maximum Wait-Time . . . 87

6.4 Pay to Control . . . 89

6.4.1 Service Provider's Game . . . 89

6.4.2 Minimizing The Expected Wait-Time . . . 90

6.4.3 Minimizing The Maximum Wait-Time . . . 92

6.5 Simulation Results . . . 95

6.5.1 Drivers' model . . . 95

6.5.2 Information Sharing . . . 98

6.5.3 Pay to Control . . . 100

7 Conclusions and Future Work **105**

7.1 Future Work 106

7.1.1 Distributed Coverage Control of Multiple Robots 106

7.1.2 Re-balancing Self-interested Drivers in Transportation Network . . 107

References **109**

List of Tables

4.1 Parameters of the experiments 50

4.2 Average percentage improvement of the sensing cost for the proposed algorithm compared to Heterogeneous Lloyd's algorithm over 500 instances for each experiment. The mean deviation from average is denoted by σ. 53

6.1 Summary of algorithms proposed. *For the pay-to-control method we minimize a linear combination of the total amount paid and the global objective. 83

List of Figures

1.1 Examples of coverage and responding to task. (a) Three aerial vehicles, equipped with cameras, are monitoring a forest. (b) A fleet of taxis responding to the ride-requests arriving in a transportation network. 2

3.1 Sampled locations in an environment with dispersion ζ. 18

3.2 Example environment with $3n$ vertices and $n + 1$ robots 21

3.3 An example of a local move in the centralized algorithm (green) and its equivalent sequences of moves (red) in the distributed algorithm. 22

3.4 Percentage difference of the solutions of different algorithms to the solution of the centralized algorithm 33

3.5 Percentage of Successful Local Move Type Attempts 33

3.6 Percentage improvement of different algorithms with random initial configurations. 34

3.7 Final Configuration and the paths of the robots for the two algorithms in a test environment 35

3.8 A sample environment with obstacles and the discrete representation of the environment. 36

3.9 Percentage difference of the solutions of different algorithms to the solution of the centralized algorithm 37

3.10 Robot movements in a non-convex environment using the proposed distributed algorithm 37

3.11 A discrete test environment and the final configurations of algorithm in [1] and the proposed algorithm 38

3.12 A test environment for Algorithm in [2] 38

4.1 The Voronoi partitions of an environment for each event type. A robot considered in a Voronoi partition if it is equipped with the required sensor for the event type. 43

4.2 Instance of coverage of three event types on a line. Triangles (resp. rectangle) show the density function for event type 1(resp. 2 and 3). Robots are capable of sensing the event types with the same color. 43

4.3 Robot located on an edge of a graph. 48

4.4 Performance of the proposed control law for Experiment 3 51

4.5 The comparison of the total sensing cost for the proposed algorithm and the Heterogeneous Lloyd's algorithm with environment settings of Experiment 3 51

4.6 Paths of eight robots covering four event types with the proposed algorithm. 52

4.7 Eight robots with different sensing capabilities in the Robotarium with the environment size of 2×3.2 meters. The density functions of each event type are shown via contours. The colored circles next to the robots show the generators of each Voronoi partition. 53

4.8 Pick-up locations in Manhattan N.Y. and a sample distribution represented with a bar chart 54

4.9 Improvement in the expected response time of the ride-requests with different sizes 55

5.1 Demonstrating a single stage of the problem. (a) Configuration of the robots after servicing task t_{i-1} at v (b) New waiting location of the robots for the new task arrival (c) Task arrival and the closest robot deployed to service the task (d) New waiting location of the robots for the next arrival. 59

5.2 Example of different solutions of single-stage optimal and infinite horizon optimal solutions. The dark vertices are occupied with the robots. (a) shows the initial configuration and the optimal solution of single-stage. (b) shows the optimal policy for the given initial configuration in infinite horizon. . . 62

5.3 Floor plan of E5 building at University of Waterloo. Floor plan contains 62 task locations (circles and squares) and 15 potential waiting locations for the robots (squares) (b) A stage of the two-stage policy performing a task and redeploying. 72

5.4 Two stages of the two-stage policy performing two tasks and two re-deployments. 73

5.5 Average deviation from optimal for different policies 74

5.6 Two-stage-Horizon with unknown probability distribution. 76

5.7 Stage cost of Two-stage-Horizon and Move-to-Median policies as a function of task arrival rate. 76

6.1 A set of drivers in the ride-sharing system and a set of locations with high probability of ride request arrival. 78

6.2 Ride-sharing problem with two vehicles and two request arrival locations. 84

6.3 An instance of Problem 6.3.3. The green vertices represent the desired waiting location of each driver if $I_i = \emptyset$, and the red vertices represent the waiting location of the drivers if $I_i = Q$. 85

6.4 Constructed MFL instance for optimizing the utility of the service provider. An instance of the edges between the subsets is shown with their respective costs. 92

6.5 Constructed bipartite graph for the problem of minimizing service provider's utility with the global objective of minimizing the maximum wait-time. . . 94

6.6 The set of pick-up and drop off locations in Manhattan. The bars at the locations of clusters represent the ride request arrival rates. 96

6.7 Environment with three drivers and a pick-up location v. (a) The red car has full information on the position of the other drivers, therefore, $S_{\text{red}}(v, q_{\text{red}}) = 2$, (b) The positions of the other drivers are not known to the driver of red car, thus, $S_{\text{red}}(v, q_{\text{red}}) = 0$. 97

6.8 Improvement in J_{exp} by sharing partial information 99

6.9 Improvement in J_{max} by sharing partial information 99

6.10 The expected wait-time of ride requests in a system of 80 drivers executing 300 ride requests arriving over time under the partial information control method. 100

6.11 The maximum wait-time of ride requests in a system of 80 drivers executing 300 ride requests arriving over time under the partial information control method. 101

6.12 Improvement in the global objectives and the amount paid to drivers using pay-to-control method. The blue represents the results for the objective J_{exp} and the green shows the results for the objective $J_{\max}$. The solid bars represent the jammed initial configuration and the hatched bars represent the random initial configuration. 102

6.13 The expected wait time and the paid amount to a system of 80 drivers executing 300 ride requests arriving over time under the pay-to-control method 103

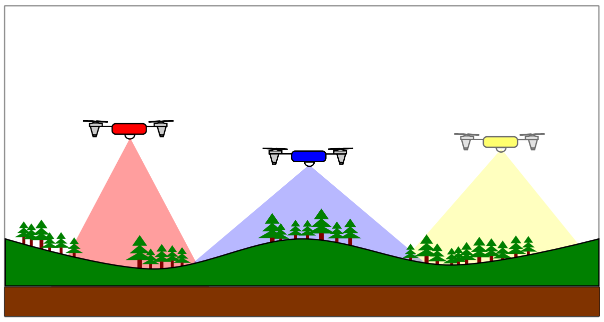
6.14 The maximum wait time and the paid amount to a system of 80 drivers executing 300 ride requests arriving over time under the pay-to-control method 104

Chapter 1

Introduction

The presence of robots is growing rapidly in commercial and industrial applications, assisting humans to perform different tasks. Some of these applications are monitoring and surveillance [3, 4, 5] by robots equipped with sensors, assisting patients in a hospital setting [6], dispatching robots in transportation applications [7, 8], data collection applications where robots are assigned to visit a set of target locations to gather data [9, 10], mobility-on-demand applications [11, 12] where a set of robots relocate the costumers between designated locations in the environment. The goal of these applications is to deploy a fleet of agents or robots to minimize the time to respond to tasks arriving in a stochastic manner or maximize the quality of observing events in an environment in a dynamically changing environment. With the dynamically changing environments, the robots are required to continuously relocate in the environment to maintain service quality. Figure 1.1 illustrates two instances of robots performing tasks in different setups. In the first example, three unmanned aerial vehicles equipped with cameras are monitoring a terrain. The second example shows a scenario where multiple drivers are responding to ride requests arriving in a transportation network.

In this thesis, we consider the deployment of robots in continuous and discrete environments. The discrete environment is represented with a graph where the vertices are the locations where the tasks arrive and the edges are the connections between different locations. The weight on each edge represents the time required for the robots to traverse the edge and the robots travel on the edges to service the tasks arriving on the vertices of the graph. We focus on the high-level control of the robots in the environment and finding the paths for them. The robots will follow the paths returned by the algorithms in this work using onboard controllers such as the controllers for the joints and collision avoidance.


(a) Aerial vehicles monitoring an environment

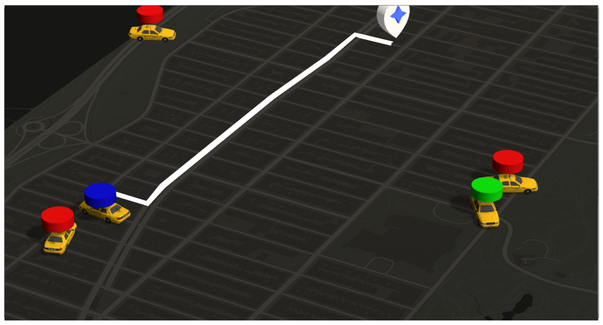

(b) Taxis responding to ride-requests

Figure 1.1: Examples of coverage and responding to task. (a) Three aerial vehicles, equipped with cameras, are monitoring a forest. (b) A fleet of taxis responding to the ride-requests arriving in a transportation network.

There are two main models for the problem of using robots to autonomously service tasks. In the first model, the set of tasks to be serviced is provided *a priori* to any action by the robots. On the other hand, the second model captures the scenarios where the tasks arrive over time. Our focus is on the problem of servicing tasks arriving sequentially over time in a stochastic environment. A common method to solve these problems is to distribute the robots in the environment based on the event distribution to minimize the expected time to service the tasks [13, 14]. Once the robots are deployed in the environment, they wait for new tasks, and upon a task arrival, the closest robot is assigned to service the task.

In applications such as forest monitoring [15], robots equipped with sensors are deployed to observe forest fires. The quality of servicing a task in these applications is analogous with the quality of observing the events using the sensor equipment on the robots. We refer to these problems in which servicing a task does not require visiting the location of it as the *coverage* problems. The coverage problems with multiple robots in convex and non-convex environments and environments with multiple events types are studied in this thesis.

In service applications, robots are tasked with responding to requests for service that arrive periodically over time [16]. For example, in a hospital setting [6] a fleet of robots may be used to assist patients by traveling to their locations. A key aspect in such applications is where the robots should wait (i.e., at their deployment locations) to optimally respond to the next service request. After a request has been serviced, the robots can redeploy to re-optimize their positions for future requests. There is an inherent trade-off between the expected response time for a service request and the cost incurred to redeploy robots between successive requests. The problem of deploying autonomous service robots to respond to sequentially arriving tasks is one of the problems considered in this thesis.

In contrast to the previous applications of the deployment problem where the robots are autonomous service units, in the ridesharing applications such as Uber and Lyft, the service units are self-interested units with the objective of maximizing their profit. The individual objective of each driver is not aligned with the objective of minimizing the time to respond to the ride requests. Therefore, to maintain the service quality by relocating the drivers, the ride-sharing companies are required to incentivize the drivers to relocate. The main challenge in these applications is that there is no direct control on the position of the service units in the network. The problem of providing indirect controls to relocate self-interested units in a transportation network is studied in this thesis.

The rest of this chapter provides a survey of the existing studies on these problems.

1.1 Literature Review

In this section, we provide a brief review of the related literature on the multi-robot deployment problem. A detailed review of the literature related to the specific problems are provided in the corresponding chapters.

The coverage control of mobile sensors has been studied extensively in the literature of robotics. In [17], the authors provide a distributed algorithm for homogeneous robots in a convex environment that utilizes Voronoi partitioning and the Lloyd descent algorithm. The significance of the algorithms is that computing the control law for each robot to converge to a local maximum sensing quality over the event in the domain requires only communication between neighboring robots in a Voronoi partitioning. Several studies address different types of heterogeneity in the sensors. In [18], the authors consider the problem of sensing an event where the sensors have different functions governing their sensing quality. The approach defines a generalized Voronoi partitioning based on the sensing functions and provides Lloyd descent type algorithm for controlling each robot. Other types of heterogeneities for mobile sensors addressed in the literature consider the sensors with different sensing ranges [19, 20] and different additive weights on the sensing quality of robots [21]. In [19], authors address the coverage problem of circular sensors with different radii. In [20], authors provide a gradient descent algorithm for the distributed control of sensors with different size circular footprints in a non-convex environment.

In the operations research literature, the problem of positioning service units in the environment is posed in the structure of the facility location problem [13, 22]. The facility location problem is the problem of installing facilities in a set of locations with a fixed cost of opening a facility. Demands arrive at different locations, and the objective is to minimize the time to respond to the demand and the total cost of opening facilities. The first constant-factor approximation algorithm for the facility location problem on a metric graph is given in [23] with an LP-rounding method providing a solution within 3.16 factor of the optimal. The next approximation algorithm followed the previous result, with a new local search approach by Korupolu *et al.* [24] proving a $(5+\epsilon)$ approximation factor and worst-case run-time of the $O(n^6 \log n/\epsilon)$. Later, Jain and Vazirani [25] improved the constant factor approximation to 3 and the run-time to $O(m \log m)$, where m is the number of edges, with a primal-dual schema. Regarding the complexity of providing approximation algorithms for the problem, unless $P = NP$, no approximation algorithm can achieve a better approximation factor than 1.463 [26]. Chudak and Shmoys in [27] proposed an LP-rounding method with a close to the theoretical bound on the approximation factor, providing a 1.736-approximation factor for the facility location problem. A special case of the facility location problem is the k-median problem [28] where the cost of opening facilities is zero.

An extension to the facility location problem is the mobile facility location problem (MFL), introduced in [29]. The objective is to move the facilities while minimizing the total movement cost and the response time. Friggstad and Salavatipour [30] proposed an LP-rounding method proving an 8-approximation factor for MFL. In [31], authors provide a simple local-search algorithm for the MFL with a $3+o(1)$ approximation ratio. In [32], an extensive set of experiments conducted on the algorithm characterize the performance of the local search algorithm in solution quality and the run-time. The two main differences between the problems considered in this thesis and the MFL are 1) robots service tasks by visiting the task location; thus the configuration changes with each arrival and 2) MFL considers just the next arrival and plans the next waiting configuration for a single-ahead stage. In contrast, we plan the next configuration of the robots for a horizon of N task arrivals.

In mobility-on-demand (MOD) applications, the problem of deploying vehicles to respond to the ride requests arriving in the network has been subject to an extensive research [11, 33, 34]. In these applications, a group of vehicles is located at a set of stations. The customers arrive at the stations, hire vehicles for a ride, and then drop the vehicles off at the station closest to their destination. The objective is to balance the vehicles at the stations to minimize the expected wait time of the customers. Several studies are dedicated to the problem of relocating self-interested drivers in transportation networks [35, 36, 37].

These studies provide pricing schemes for the ride-sharing companies to minimizing the wait-time of the customers by incentivizing the drivers to relocate. The general approach in these studies is to provide a flow formulation to approximate the average number of vehicles to relocate from a station to others. The problem of distributing self-interested units in a transportation network is one of the problems studied in this thesis. In contrast to the MOD studies, in this thesis, we focus on the movement of the individual drivers in the transportation network.

1.2 Contribution and Organization

The organization and contributions of this thesis are as follows.

Chapter 2 In this chapter, we provide mathematical preliminaries and some definitions and notation.

Chapter 3 In this chapter, we revisit the distributed coverage control problem with multiple robots on both metric graphs and in non-convex continuous environments. Traditionally, the solutions provided for this problem converge to a locally optimal solution with no guarantees on the quality of the solution. We consider sub-additive sensing functions, which capture the scenarios where sensing an event requires the robot to visit the event location. For these sensing functions, we provide the first constant factor approximation algorithms for the distributed coverage problem. The approximation results require twice the conventional communication range in the existing coverage algorithms. However, we show through extensive simulation results that the proposed approximation algorithms outperform several existing algorithms in convex, non-convex continuous, and discrete environments even with the conventional communication ranges. Moreover, the proposed algorithms match the state-of-the-art centralized algorithms in the solution quality.

This chapter is the result of a collaboration with Ahmad Bilal Asghar. I took the lead in defining the problem and proposing the algorithm in this chapter. We collaborated in proving the analytical results and providing the experimental results in this chapter.

The work presented in this chapter is based on the following publication.

- Armin Sadeghi, Ahmad Bilal Asghar, and Stephen L Smith. Approximation algorithms for distributed multi-robot coverage in non-convex environments. The *14th International Workshop on the Algorithmic Foundations of Robotics*, 2020.

Chapter 4 In this chapter, we focus on a natural extension of the well-known distributed coverage control problem where a set of autonomous robots are deployed to efficiently monitor multiple types of events in an environment. There is a density function over the environment for each event type representing the weighted likelihood of the event at each location. The robots are heterogeneous in that each robot is equipped with a set of sensors and it is capable of sensing a subset of event types. The objective is to deploy the robots in the environment to minimize a linear combination of the total sensing quality of the events. We propose a new formulation for the problem which is a natural extension of the homogeneous problem. Then, we propose distributed algorithms that drive the robots to locally optimal positions in both continuous environments that are obstacle-free, and in discrete environments that may contain obstacles. In both cases we prove convergence to locally optimal positions. We provide extension to the case where the density functions are unknown prior to the deployment in continuous environments. Finally, we present benchmarking results and physical experiments to characterize the solution quality.

The work presented in this chapter is based on the following publication.

- Armin Sadeghi and Stephen L Smith. Coverage control for multiple event types with heterogeneous robots. In *IEEE International Conference on Robotics and Automation*, pages 3377–3383, 2019.

Chapter 5 In this chapter, we focus on the problem of deploying a set of autonomous robots to efficiently service tasks that arrive sequentially in an environment over time. Each task is serviced when the robot visits the corresponding task location. Robots can then redeploy while waiting for the next task to arrive. The objective is to redeploy the robots taking into account the next N task arrivals. We seek to minimize a linear combination of the expected cost to service tasks and the redeployment cost between task arrivals. In the single robot case, we propose a one-stage greedy algorithm and prove its optimality. For multiple robots, the problem is NP-hard, and we propose two constant-factor approximation algorithm, one for the problem with a horizon of two task arrivals and the other for the infinite horizon when redeployment cost is weighted more heavily than service cost. Finally, we present extensive benchmarking results to characterize both solution quality and run-time.

The work presented in this chapter is based on the following publication.

- Armin Sadeghi and Stephen L Smith. Re-deployment algorithms for multiple service robots to optimize task response. In *IEEE International Conference on Robotics and Automation*, pages 2356–2363, 2018.

Chapter 6 This chapter focuses on the problem of controlling self-interested drivers in ride-sourcing applications. The objective of the ride-sharing company is to improve the customer experience by minimizing the wait-time before pick-up. Meanwhile, the drivers attempt to maximize their profit by choosing the best location to wait in the environment between the ride requests assigned to them. The objectives of the ride-sharing company and the drivers are not aligned, and the company has no direct control over the waiting locations of the drivers. The focus of this chapter is to provide two indirect control methods for the ride-sharing company to optimize the set of waiting locations of the drivers, thereby minimize one of two objectives: 1) the expected wait-time of the customers, or 2) the maximum wait-time of customers. The proposed indirect control methods are 1) sharing information to a subset of the drivers on the location of other waiting drivers, and 2) paying drivers to relocate. We show that the problem of finding the optimal control is NP-hard for both objectives and both control methods. For the information sharing method, we provide an LP-rounding algorithm to minimize the expected wait-time and a 3-approximation algorithm to minimize the maximum wait-time. To incentivize the drivers to relocate with payments, we provide 3-approximation algorithms for both objectives. Finally, we evaluate the proposed control methods on real-world data and show that we can achieve up to 80% improvement for both objectives.

The work presented in this chapter is based on the following publications.

- Armin Sadeghi and Stephen L Smith. On re-balancing self-interested agents in ride-sourcing transportation networks. In *IEEE International Conference on Decision and Control*, pages 5119–5125, 2019.

- Armin Sadeghi and Stephen L Smith. Re-Balancing Self-Interested Drivers in Ride-Sharing Networks to Improve Customer Wait-Time. (submitted).

Chapter 7 This chapter consists of a summary of the work presented in this thesis along with potential directions for future work.

Other Publications I have collaborated on the following publication which is not presented in this thesis.

- Armin Sadeghi, Ahmad Bilal Asghar, and Stephen L. Smith. On Minimum Time Multi-Robot Planning with Guarantees on the Total Collected Reward. In *IEEE International Symposium on Multi-Robot and Multi-Agent Systems*, pages 16–22, 2019.

Chapter 2

Preliminaries

In this chapter, we provide mathematical background and notation for this thesis. In In Section 2.1, we provide some definitions from graph theory. Section 2.2 consists of a review of the complexity theory. Finally, in Section 2.4 we provide some combinatorial problems and the state-of-the-art algorithms for these problems. The discussion on complexity theory is from [38] and [39], and the review of the graph theory definitions is from [40].

2.1 Graphs

A graph G is defined as a pair of sets $G = (\mathcal{V}, \mathcal{E})$, where the set $\mathcal{V}$ represents the vertices in the graph and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges between the vertices. An edge in $\mathcal{E}$ is an ordered tuple $\langle u, v \rangle$ connecting $u \in \mathcal{V}$ to $v \in \mathcal{V}$.

Definition 2.1.1 (Complete Graph). *A graph $G = (\mathcal{V}, \mathcal{E})$ is complete if for every pair of distinct vertices u and v in $\mathcal{V}$, the edge $\langle u, v \rangle$ is in $\mathcal{E}$.*

Definition 2.1.2 (Undirected Graph). *A graph $G = (\mathcal{V}, \mathcal{E})$ is undirected if for every edge $\langle u, v \rangle \in \mathcal{E}$, the edge $\langle v, u \rangle$ is in $\mathcal{E}$.*

For simplicity in the representation, the edges in undirected graphs are denoted by unordered tuple $(u, v) \in \mathcal{V}$.

Definition 2.1.3 (Neighbours of a Vertex). *In an undirected graph $G = (\mathcal{V}, \mathcal{E})$, neighbours of a vertex $u \in \mathcal{V}$ is the set of vertices that are connected to u by an edge, i.e.,*

$$\mathcal{N}(u) = \{v \in \mathcal{V} | (u, v) \in \mathcal{E}\}.$$

Definition 2.1.4 (Weighted Graph). *A weighted graph is defined as triple $G = (\mathcal{V}, \mathcal{E}, c)$ where $c : \mathcal{E} \to \mathbb{R}$ is a function assigning weights to each edge in the graph.*

In this thesis, the cost of an edge $e = (u, v)$ is represented by $c(e)$ or $c(u, v)$ interchangeably.

Definition 2.1.5 (Metric Graph). *A weighted graph $G = (\mathcal{V}, \mathcal{E}, c)$ is called a metric graph, if it is an undirected-complete graph and the weights on the edges satisfy the triangle inequality, i.e., for edges $(u, v), (v, w)$ and (u, w) we have*

$$c(u, w) \leq c(u, v) + c(v, w).$$

Definition 2.1.6 (Subgraph). *A Subgraph of a graph $G = (\mathcal{V}, \mathcal{E})$ is a graph $H = (\mathcal{V}_H, \mathcal{E}_H)$ with $\mathcal{V}_H \subseteq \mathcal{V}$ and $\mathcal{E}_H \subseteq \mathcal{E}$. Subgraph H is called a spanning subgraph if $\mathcal{V}_H = \mathcal{V}$.*

Definition 2.1.7 (Simple Path). *A path $\mathcal{P}$ in graph G is a subgraph $(\mathcal{V}_\mathcal{P}, \mathcal{E}_\mathcal{P})$ of G where*

- $\mathcal{V}_\mathcal{P} = \{v_1, v_2, \ldots, v_{k+1}\}$ *such that* $v_i \neq v_j$ *for* $1 \leq i < j \leq k + 1$, *and*
- $\mathcal{E}_\mathcal{P} = \{(v_i, v_{i+1}) | 1 \leq i \leq k\}$.

An equivalent representation of a path is by a sequence of vertices from v_1 to v_{k+1} connected by edges. The cost of a path $\mathcal{P}$ in a weighted graph is the total cost of the edges in the graph, i.e., $\text{cost}(\mathcal{P}) = \sum_{i=1}^{k} c(v_i, v_{i+1})$. The distance $\text{dist}(u, v)$ of two vertices $u, v \in \mathcal{V}$ in a weighted graph is the cost of the shortest path from u to v. In this thesis, the paths considered are simple paths and for the sake of brevity we will refer to them as paths.

Definition 2.1.8 (Connected Undirected Graph). *An undirected graph $G = (\mathcal{V}, \mathcal{E})$ is called connected if there is a path between every pairs of vertices in $u, v \in \mathcal{V}$.*

Definition 2.1.9 (Independent Set). *A subset of vertices S in a graph $G = (\mathcal{V}, \mathcal{E})$ is called an independent set, if there is no edge in $\mathcal{E}$ between any pairs of vertices in S.*

An independent set S is called a *maximal independent set*, if the set $S \cup \{v\}$ is not an independent set for every vertex $v \in \mathcal{V} \setminus S$.

2.2 Complexity Theory

Classifying the complexity of a problem is a key step to solve the problem. Understanding the computational complexity of a problem will help in choosing the approach to solve it. For instance, if a problem is in class P, then the problem can be solved in polynomial time, therefore, the solver approach to the problem should be an algorithm that finds the solution in polynomial amount time. On the other hand, there exist problems that finding a polynomial-time algorithms for them is an open problem, i.e., NP-hard problem (detailed description is given below). Therefore, the solver approaches for these problems are 1) *exact* solvers which are non-polynomial time solvers, 2) *approximate* solvers which are polynomial-time algorithms and find a near-optimal solution with theoretical bound on the error from the exact solution, and 3) *heuristic* solvers which are usually polynomial-time algorithms without theoretical guarantees on their solution.

2.3 Reductions and Complexity Classes

The two classes of problems discussed in this thesis are: 1) the decision problems where the answer to the problem is a *yes* or a *no*, and 2) the optimization problems where the goal is to minimize or maximize an objective function obj. The decision version of an optimization problem has the same inputs with an additional value B as a budget on the objective function. Therefore, the decision version of a maximization (resp. minimization) problem becomes the problem of asking if there exists a solution Q where the objective function at Q, i.e. $\text{obj}(Q)$, is at most (resp. at least) B. For instance, the optimization version of the Maximum Independent Set problem is to find the maximum size subset of vertices Q such that there is no edge in the graph between the vertices in Q, whereas the decision version of the problem is whether a subset Q with size $|Q| \geq B$ exists such that there is no edge between the vertices in Q.

Let A and B be two decision problems, then a *reduction* is a polynomial-time transformation R from an instance of A to an instance of B. That is, for an instance I of problem A, $R(I)$ is an instance of problem B such that I is a *yes* instance of A if and only if $R(I)$ is a *yes* instance of B. Hence, given a polynomial-time algorithm for B, there is a polynomial-time algorithm for A, i.e., the reduction R to construct an instance of B followed by the algorithm for B on the constructed instance. Therefore, given a reduction from A to B and an algorithm for problem B, we can solve A, which means problem B is at least as hard as problem A.

The complexity theory is the study of classifying problems according to the problems' complexity. The class NP is the set of problems that have solutions that are verifiable in polynomial time. For instance, given a *yes* answer to the decision version of the Maximum Independent Set problem, the answer can be verified in polynomial-time by checking the size of the solution and if the solution is an independent set. A problem A is an NP-hard problem when for every problem B in NP, there is a polynomial-time reduction from B to A. Hence, problem A is at least as hard as any problem in NP. A problem A in NP is said to be an NP-complete problem if it is an NP-hard problem.

In this thesis, we consider a set of optimization problems such that their decision version is in the class NP-hard. An algorithm is said to be an *approximation algorithm* for an optimization problem if it finds a solution within a multiplicative factor of the optimal solution to the problem.

2.4 Combinatorial problems

In this section, we discuss a number of combinatorial problems that we will be referring to in this thesis.

2.4.1 Minimum Weight Bipartite Matching

Consider a graph $G = (A \cup B, \mathcal{E}, c)$ where A and B are non-intersecting sets of vertices. The edge set $\mathcal{E}$ consists of the edges between the pairs of vertices $u \in A$ and $v \in B$. A subset of edges $\mathcal{E}' \subseteq \mathcal{E}$ is called a *perfect matching* if each vertex in A is connected to one and only vertex in B with an edge in $\mathcal{E}'$. The minimum weight bipartite matching problem is the problem of finding a perfect matching $\mathcal{E}'$ such that the total cost of the edges in $\mathcal{E}'$ is minimized. The decision version of the minimum weight bipartite matching problem is in the class P, and there is a polynomial-time algorithm for the optimization version [41]. In the remaining of the thesis, the minimum weight bipartite assignment between the sets A and B is denoted by $\text{Assgn}(A, B)$.

2.4.2 Facility Location Problem

Consider a weighted-metric graph $G = (\mathcal{V}, \mathcal{E}, c)$ where $\mathcal{V}$ represents the vertices, $\mathcal{E}$ represents the edges and $c : \mathcal{E} \to \mathbb{R}_+$ is a function assigning travel time to each edge. There is a function $w : \mathcal{V} \to \mathbb{R}_+$ representing the demands on each vertex. There is a cost $f : \mathcal{V} \to \mathbb{R}_+$ associated with constructing a facility at each vertex to service the demands. The cost of servicing a demand is the distance of the demand vertex to the closest facility. The objective is to find a subset $Q \subseteq \mathcal{V}$ to construct the facilities such that the total cost of constructing facilities and the cost of servicing demands is minimized, i.e.,

$$\underset{Q\subseteq\mathcal{V}}{\text{minimize}} \sum_{q\in Q} f(q) + \sum_{u\in\mathcal{V}} w(u) \min_{q\in Q} c(u,q).$$

Several approximation algorithms are proposed for the facility location problem [27, 23, 25]. The state-of-the-art approximation algorithm provides a solution within a 1.488-factor of the optimal solution [42].

2.4.3 k-median Problem

The k-median problem [28] is a variant of the facility location problem with a constraint on the number of facilities. The input to the problem is a weighted-metric graph $G = (\mathcal{V}, \mathcal{E}, c)$, an integer k, and a demand $w : \mathcal{V} \to \mathbb{R}_+$ for each vertex. The cost of construing facilities on the vertices are zero, i.e., $f = 0$ for all $u \in \mathcal{V}$. The objective is to find a subset $Q \subseteq \mathcal{V}$ with size $|Q| \leq k$ to construct the facilities such that the cost of servicing the demands is minimized, i.e.,

$$\begin{array}{ll} \underset{Q\subseteq\mathcal{V}}{\text{minimize}} & \sum_{u\in\mathcal{V}} w(u) \min_{q\in Q} c(u,q) \\ \text{subject to} & |Q| \leq k. \end{array}$$

An extension to the well-known k-median problem is the mobile facility location problem (MFL) [31]. The input to the MFL problem is a weighted-metric graph $G = (\mathcal{V}, \mathcal{E}, c)$, an integer k, a demand $w : \mathcal{V} \to \mathbb{R}_+$ for each vertex and an initial configuration of the facilities Q_0. The objective is to find a subset $Q \subseteq V$ with size $|Q| \leq k$ to minimize a linear combination of the relocation cost of facilities from Q_0 to Q and the service cost of facilities at Q, i.e.,

$$\begin{array}{ll} \underset{Q\subseteq\mathcal{V}}{\text{minimize}} & \text{Assgn}(Q_0, Q) + \beta \sum_{u\in\mathcal{V}} w(u) \min_{q\in Q} c(u,q) \\ \text{subject to} & |Q| \leq k, \end{array}$$

where $\text{Assgn}(Q_0, Q)$ is the minimum weight bipartite matching between vertices in Q_0 and Q and β is a user-defined variable. The high β values represent the scenarios where the service costs if the priority and low β values represent the scenarios where the relocation cost is more important. Authors in [31] provide a local-search algorithm for the MFL problem and prove that the solution obtained from the proposed algorithm is within factor 3 of the optimal solution.

2.4.4 k-center Problem

Consider a weighted-metric graph $G = (\mathcal{V}, \mathcal{E}, c)$ where $\mathcal{V}$ represents the vertices, $\mathcal{E}$ represents the edges and $c : \mathcal{E} \to \mathbb{R}_+$ is a function assigning travel time to each edge. The cost of servicing a demand is the distance of the demand vertex to the closest facility. The objective is to find a subset $Q \subseteq \mathcal{V}$ with size $|Q| \leq k$ to construct the facilities such that the maximum cost of servicing the demands is minimized, i.e.,

$$\begin{aligned} \underset{Q \subseteq \mathcal{V}}{\text{minimize}} \quad & \max\nolimits_{u \in \mathcal{V}} \min\nolimits_{q \in Q} c(u, q) \\ \text{subject to} \quad & |Q| \leq k. \end{aligned}$$

The authors in [39] provide a 2-approximation algorithm for the k-center problem.

2.4.5 CNF-SAT

Consider a set of n Boolean variables $\{x_1, \ldots, x_n\}$. A variable x_i or its negation $\neg x_i$ is called a literal. A clause is a disjunction ($\vee$, or) of literals, and a formula, in conjunctive normal form, is a conjunction ($\wedge$, and) of clauses. Given a formula in the conjunctive normal form F, the CNF-SAT problem is the problem of determining if there is an assignment for the Boolean variables $\{x_1, \ldots, x_n\}$ such that F is satisfied.

Chapter 3

Homogeneous Multi-Robot Coverage

Multi-robot Coverage is a well studied problem [17, 43, 44, 45, 46] with extensive multi-robot applications, such as environmental monitoring [5], and surveillance [4]. The objective is to deploy a set of robots to cover an environment such that each robot services or senses the events closer to that robot than any other robot. The events arrive according to some spatial distribution, and the cost of sensing an event is a function of the distance from the robot to that event. The distributed coverage control problem is to minimize the total coverage cost of the environment. The existing distributed algorithms to solve this problem converge to a locally optimal solution with no theoretical bound on the difference of the objective with the locally optimal solution and the globally optimal solution. In this chapter, we provide distributed approximation algorithms to solve the problem in non-convex continuous and discrete environments.

The contributions of this chapter are threefold. First, given a continuous non-convex coverage problem, we generate a corresponding instance on a metric graph and characterize the performance of the discrete solution on the continuous problem (Section 3.2). Second, we provide a constant-factor approximation algorithm for the distributed coverage problem on metric graphs (Section 3.3). To the best of our knowledge, this is the first deterministic approximation algorithm for the distributed coverage problem. We prove the approximation results in Section 3.4. Third, we show through extensive simulations that the proposed algorithm outperforms several existing approaches in convex and non-convex environments and matches the centralized algorithms in solution quality (Section 3.5).

This chapter is the result of a collaboration with Ahmad Bilal Asghar for [47]. I took the lead in defining the problem and proposing the algorithm in this chapter. We collaborated in proving the analytical results and providing the experimental results in this chapter.

3.1 Related Work

The first distributed algorithm for coverage control in convex environments was proposed by Cortes *et al* [17]. The algorithm utilizes Lloyd's descent to converge to a locally optimal solution, and the partition of each robot is defined using Voronoi partitioning. The robots communicate with the robots in their neighboring partitions to implement the algorithm.

Building on Lloyd's descent-based algorithm in [17], there have been extensive studies on the coverage control problem in non-convex environments. In [48, 49], the authors map non-convex environments through a diffeomorphism to a convex region and then solve the problem using Lloyd's algorithm [17] before mapping the locally optimal solution back to the original environment. For non-convex polygonal environments, a distributed algorithm was presented in [2], where Lloyd's algorithm for convex environments was combined with a local path planning algorithm to avoid obstacles. A key idea introduced in [50] is to consider geodesic distance when computing partitions. In [51] and [52], the authors construct the Voronoi partitions based on the visibility of the robot in the presence of obstacles. Unlike the approaches mentioned above, we discretize the non-convex environment, and solve the coverage problem on the discrete environment and provide guarantees on the solution quality.

The approach of converting a continuous non-convex environment to a discrete environment is used in [43, 53, 54]. We utilize the same approach but we characterize the cost of the solution obtained from the discretized environment in the corresponding continuous environment as a function of the sampling density. The authors in [1] study the coverage problem defined on an undirected graph and present a distributed algorithm that converges to a local optimum. Their algorithm requires the robots to know the information of the neighbors of their neighbors. In this chapter, we make the same assumption on the communication range of the robots but establish approximation guarantees.

A closely related problem to the discretized coverage control is the facility location problem [25, 55] where the objective is to minimize the cost of the robots and the total service time of the demands arriving on the vertices. A special case of this problem is the k-median problem [28, 56, 31] where the objective is to place k robots on vertices of the graph to minimize the total service time. A centralized approximation algorithm was presented for the k-median problem in [28], and the analysis of our distributed approximation algorithm leverages this centralized approximation algorithm. In [57], we consider the mobile facility location problem with sequentially arriving demands where the goal is to minimize a linear combination of the relocation costs and the expected service times of the demands in a time horizon, and propose a centralized algorithm which provides solutions within a constant factor of the optimal solution. The authors in [58] study a k-median clustering problem with distributed Euclidean data, and provide a randomized constant factor approximation based on distributed coreset computation. In contrast, we consider more general non-convex environments and provide a deterministic approximation algorithm.

3.2 Continuous and Discrete Coverage Problems

We begin by reviewing the coverage problems in both continuous [17] and discrete environments [1].

3.2.1 Continuous Environment

Consider m mobile robots in a compact environment with obstacles and let $\mathcal{X}$ be the obstacle free subset of the environment. There is an event distribution $\phi : \mathcal{X} \to \mathbb{R}_+$ defined over the environment. Let $d(p, q)$ be the length of the shortest path between two locations p and q in $\mathcal{X}$. The sensing cost of an event at location p by a robot at q is a monotonically increasing function $f : \mathbb{R}_+ \to \mathbb{R}_+$ of $d(p, q)$. Following the non-convex problem formulation in [2], which extends the original formulation in [17], the continuous problem is defined as the problem of finding the set of locations in the environment for the robots that minimizes the sensing cost of the events, i.e.,

$$\min_{Q \in \mathcal{X}^m} \mathcal{H}(\mathcal{Q}) = \min_{Q \in \mathcal{X}^m} \int_{\mathcal{X}} \min_{q_i \in Q} f(d(p, q_i))\phi(p)dp. \tag{3.1}$$

Without loss of generality, in the rest of the chapter we assume that $\int_{\mathcal{X}} \phi(p)dp = 1$. Observe that the best sensing cost for an event is provided by the closest robot to that event location. Then for a given configuration Q, we partition the environment into Voronoi subsets as follows:

$$\mathtt{Vor}_i(Q) = \{p \in \mathcal{X} | d(p, q_i) \leq d(p, q_j) \ \forall q_j \in Q \setminus \{q_i\}\}.$$

The robots move according to some dynamics $\dot{q}_i = g(q_i, u_i)$ where the computation of the shortest path between two configurations of the robot is tractable. Typically first order dynamics $g(r_i, u_i) = u_i$ is considered for the robots in coverage control literature [17]. We are interested in the distributed version of the coverage problem, where the robots have local information on the other robots and each robot computes its control input locally.

3.2.2 Discrete Environment

Consider a metric graph $G = (\mathcal{S}, \mathcal{E}, c)$ where $\mathcal{S}$ is the vertex set, $\mathcal{E}$ is the set of edges between the vertices, c is the metric edge cost and let w be the weight on the vertices. Given a team of m robots, the coverage problem on graph G, is the problem of finding a set of m locations to optimally cover the vertices of G, i.e., minimize

$$\mathcal{D}(Q) = \sum_{v \in \mathcal{S}} \min_{q \in Q} w(v) c(v, q)$$

. For a given configuration $Q \subseteq \mathcal{S}$, we can partition the vertices into m subsets

$$W_i(Q) = \{u \in \mathcal{S} | c(u, q_i) < c(u, q_j) \ \forall q_j \in Q \setminus \{q_i\}\}.$$

If there exists a vertex that has equal distance to two or more robots in Q, then the vertex is assigned to the robot with smaller unique identifier (UID). Robots travel on the edges of the graph and the control input to a robot is a sequence of edges leading to its destination vertex.

Centralized Approximation Algorithm: The centralized version of this problem is a well-known NP-hard problem called the k-median problem [28]. The best known approximation algorithm for this problem on metric graphs is a centralized local search algorithm which provides solutions within a constant factor of the optimal [31]. Starting from a configuration Q, the centralized local search algorithm swaps p vertices in Q at a time with a subset of p vertices in $\mathcal{S} \setminus Q$. If the new configuration improves the coverage by at least some $\epsilon_0 > 0$, then we call this move a valid local move. The procedure terminates if there are no more valid swaps improving the total sensing cost. We will refer to this local search algorithm as CentralizedAlg in the rest of the chapter. The solution obtained from CentralizedAlg is within a $3 + 2/p + o(\epsilon_0)$ factor of the globally optimal solution.

We focus on the distributed version of this problem introduced in [1] where the robots use only the local information to compute their control input. In the following section, we establish the connection between the continuous and discrete coverage problems.

3.2.3 From a Continuous to a Discrete Problem

To establish a connection between the coverage problem in continuous and discrete environments, we first convert the continuous coverage problem to a coverage problem in a discrete environment through sampling of the environment.

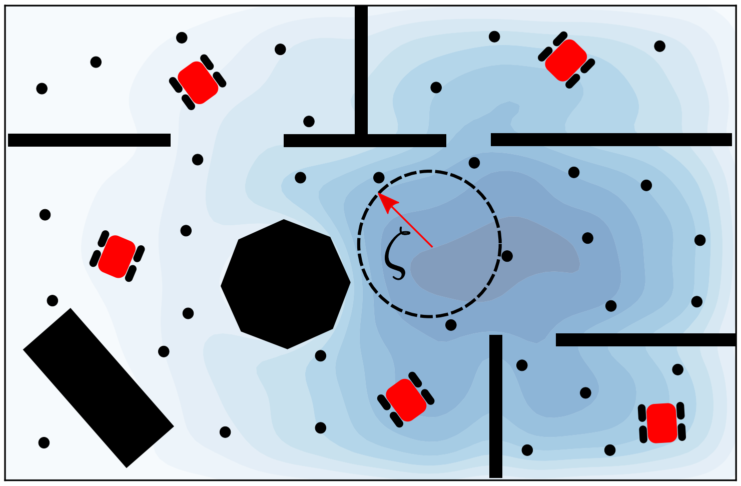


Figure 3.1: Sampled locations in an environment with dispersion ζ.

Let $\mathcal{S}$ be the set of samples of $\mathcal{X}$ with dispersion ζ [59], this means ζ is the maximum distance of any point in the environment $\mathcal{X}$ from the closest point in $\mathcal{S}$, i.e., $\zeta = \max_{p\in\mathcal{X}} \min_{u\in\mathcal{S}} d(p,u)$, (See Figure 3.1). We construct a metric graph $G = (\mathcal{S}, \mathcal{E}, c)$ on sampled locations $\mathcal{S}$, where $\mathcal{E}$ is the edge set and c is a function assigning costs to the edges of the graph. The cost of an edge between two sampled locations $u, v \in \mathcal{S}$ is $c(u,v) = f(d(u,v))$. Let $\sigma(v)$ for $v \in \mathcal{S}$ be the points in $\mathcal{X}$ closer to v than other samples in $\mathcal{S}$, i.e.,

$$\sigma(v) = \{p \in \mathcal{X} | d(p,v) \leq d(p,u)\ \forall u \in \mathcal{S} \setminus \{v\}\}.$$

With a slight abuse of notation, let $\sigma^{-1}(p)$ be the closest sample in $\mathcal{S}$ to $p \in \mathcal{X}$. The function $w : \mathcal{S} \to \mathbb{R}_+$ assigning weights to the vertices of the graph is $w(v) = \int_{p\in\sigma(v)} \phi(p)dp$. We assume the following property on the sensing function.

Assumption 3.2.1 (Subadditivity of sensing function)**.** *We assume that the sensing cost function f is a sub-additive function, i.e.,*

$$f(d(p,u) + d(u,v)) \leq f(d(p,u)) + f(d(u,v)).$$

For instance $f(x) = \sqrt{x}$ and $f(x) = x$ are sub-additive functions.

Remark 3.2.2. In applications such as dynamic vehicle routing problems [60] and facility location problems [55] where the sensing cost is determined by the distance traveled by the clients, the sensing function falls under Assumption 3.2.1. •

Due to Assumption 3.2.1, the cost function c on the edges of the graph G satisfies the triangle inequality, i.e., for all u, v, z in $\mathcal{S}$

$$c(u,v) = f(d(u,v)) \leq f(d(u,z) + d(z,v)) \leq f(d(u,z)) + f(d(z,v)) = c(u,z) + c(z,v).$$

The following result establishes a connection between the sensing costs of an approximate solution to the discrete coverage problem and the optimal coverage in continuous environment.

Theorem 3.2.3. *Consider a continuous coverage problem on environment $\mathcal{X}$ with optimal solution $S^* \in \mathcal{X}^m$, and its corresponding discrete instance obtained through the set of samples $\mathcal{S}$ with dispersion ζ. Then if Q is the solution obtained from an α-approximation algorithm for the discrete coverage problem instance, the sensing cost of Q on the corresponding continuous problem is $\mathcal{H}(Q) \leq \alpha\mathcal{H}(S^*) + o(f(\zeta))$.*

Proof. We have,

$$\begin{aligned}
\mathcal{H}(Q) &= \int_{\mathcal{X}} \min_{q_i \in Q} f(d(q_i, p))\phi(p)dp \\
&\leq \int_{\mathcal{X}} \min_{q_i \in Q} f(d(q_i, \sigma^{-1}(p)) + d(\sigma^{-1}(p), p))\phi(p)dp \quad \text{(triangle inequality)} \\
&\leq \int_{\mathcal{X}} \min_{q_i \in Q} [f(d(q_i, \sigma^{-1}(p))) + f(d(\sigma^{-1}(p), p))]\phi(p)dp \quad \text{(Assumption 3.2.1)} \\
&= \int_{\mathcal{X}} \min_{q_i \in Q} f(d(q_i, \sigma^{-1}(p)))\phi(p)dp + \int_{\mathcal{X}} f(d(\sigma^{-1}(p), p))\phi(p)dp \\
&\leq \mathcal{D}(Q) + f(\zeta)\int_{\mathcal{X}} \phi(p)dp = \mathcal{D}(Q) + f(\zeta). \qquad (3.2)
\end{aligned}$$

Let $S^* = \{q_1^*, \ldots, q_m^*\}$ be the optimal configuration of the continuous problem, and S_G^* be the configuration constructed by moving each robot location in S^* to the closest sampled location in $\mathcal{S}$. Also note that,

$$\begin{aligned}
\mathcal{D}(S_G^*) &= \sum_{q_i \in S_G^*} \sum_{u \in W_i} c(u, q_i)w(u) = \sum_{q_i \in S_G^*} \sum_{u \in W_i} c(u, q_i) \int_{p \in \sigma(u)} \phi(p)dp \\
&= \sum_{q_i \in S_G^*} \sum_{u \in W_i} \int_{p \in \sigma(u)} f(d(u, q_i))\phi(p)dp \\
&\leq \sum_{q_i \in S_G^*} \sum_{u \in W_i} \int_{p \in \sigma(u)} [\min_{q_j \in S_G^*} f(d(q_j, p)) + 2f(d(u,p)) + f(d(p,u))]\phi(p)dp \\
&\leq \mathcal{H}(S_G^*) + 3f(\zeta), \qquad (3.3)
\end{aligned}$$

where the first inequality is due to triangle inequality and Assumption 3.2.1. Furthermore, we have,

$$\begin{aligned}\mathcal{H}(S_G^*) &= \sum_{q_i \in S_G^*} \int_{\texttt{Vor}_i(S_G^*)} f(q_i, p)\phi(p)dp \le \sum_{q_i \in S_G^*} \int_{\texttt{Vor}_i(S^*)} f(q_i, p)\phi(p)dp \\ &\le \sum_{q_i \in S_G^*} \int_{\texttt{Vor}_i(S^*)} f(d(q_i^*, p))\phi(p)dp + \int_{\mathcal{X}} f(d(q_i, q_i^*))\phi(p)dp \\ &\le \mathcal{H}(S^*) + f(\zeta), \end{aligned} \tag{3.4}$$

where the second inequality is due to triangle inequality and Assumption 3.2.1. Let Q_G^* be the optimal solution to the discrete coverage problem on graph G, then

$$D(Q) \le \alpha \mathcal{D}(Q_G^*) \le \alpha \mathcal{D}(S_G^*).$$

Therefore, by Equations (3.2), (3.3) and (3.4), we have,

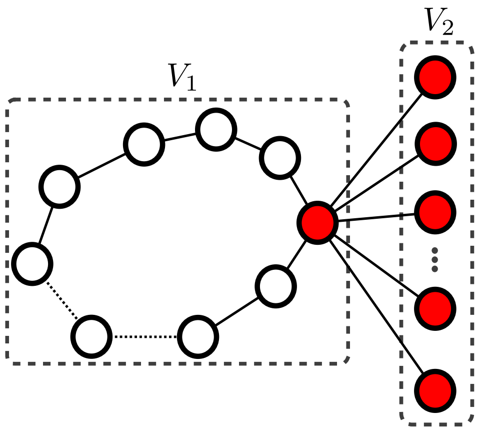
$$\mathcal{H}(Q) \le \alpha \mathcal{H}(S^*) + (4\alpha + 1) f(\zeta). \qquad \square$$

A 5-approximation algorithm for the centralized coverage on metric graphs is provided in [28]. In the following section, we provide the first distributed approximation algorithm for the coverage in metric graphs.

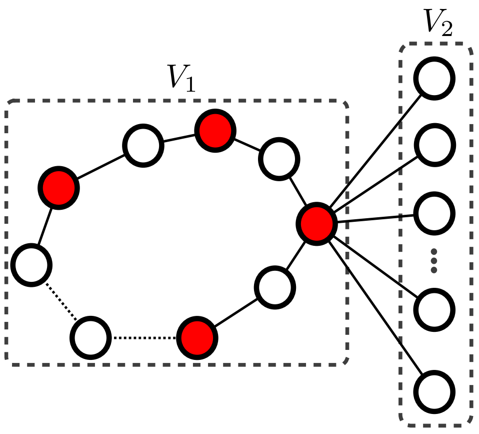
3.3 Distributed Algorithm On Graphs

In distributed coverage control algorithms for continuous environments, and their adaptations to discrete environments, the algorithm drives each robot to the position inside its partition such that the sensing cost of its partition is minimized, i.e., the centroid of its Voronoi cell in the continuous problem. Although these algorithms converge to locally optimal solutions, there are no global guarantees on the quality of the solution. The following example provides a graph construction where such "move to centroid" algorithms perform poorly.

Example 3.3.1. Consider the environment shown in Figure 3.2 with $3n+1$ vertices, $n+1$ robots and unit costs for the shown edges. We consider the metric completion of the shown graph. The vertices are partitioned into two subsets: 1) V_1 with $2n+1$ vertices and unit weights on the vertices and 2) V_2 with n vertices of weights ϵ for some $0 < \epsilon \ll 1$. The highlighted vertices show the configuration of the robots. The configuration in Figure 3.2a



(a) Locally optimal configuration under move to centroid control law



(b) A better configuration

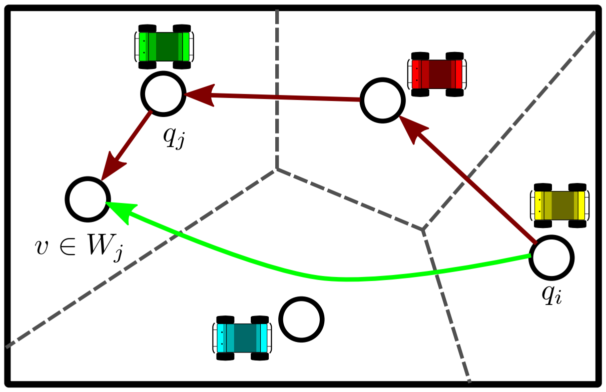
Figure 3.2: Example environment with $3n$ vertices and $n+1$ robots

is a locally optimal solution under the move to centroid control law with global cost of $n(n+1)$. However, the configuration shown in Figure 3.2b provides a global cost of $n+n\epsilon$. Therefore, the locally optimal solution provided by the move to centroid algorithm provides a solution with cost at least $\frac{n+1}{1+\epsilon}$ of the optimal cost on the shown instance. •

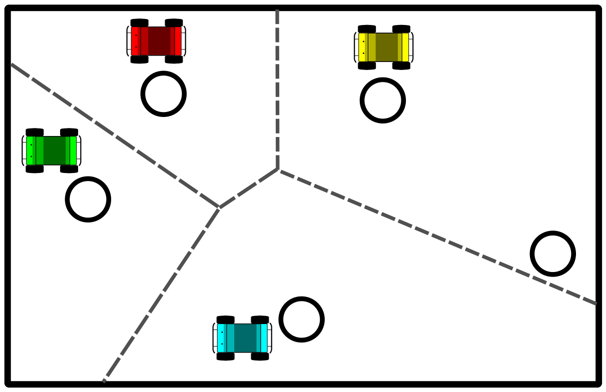
3.3.1 High-level Idea

The basic idea of our distributed coverage algorithm is to imitate the local-search algorithm for the k-median problem (see Section 3.2.2), namely CENTRALIZEDALG, in a distributed manner. The challenge in performing a local move in the distributed manner is that the robots are only aware of the partitions of their neighboring robots, therefore, the effect of a local move on the global objective is not known to the robots. However, we break down a local move in CENTRALIZEDALG into a sequence of moves between neighbors. Let robot j with position q_j and neighbors $\mathcal{N}(j)$ be the closest robot to vertex v. Then a local move of CENTRALIZEDALG swapping the position q_i of robot i with vertex v is equivalent to a sequence of swaps inside Q between the neighboring robots and a move from q_j to v. Figure 3.3 shows an example of a local move in the centralized algorithm performed by a sequence of local moves.

For this distributed coverage algorithm, we define the minimum communication range and neighbouring robots as follows:



(a) Equivalent local moves



(b) Final configuration

Figure 3.3: An example of a local move in the centralized algorithm (green) and its equivalent sequences of moves (red) in the distributed algorithm.

Definition 3.3.2 (Neighbour robots)**.** *Given a configuration Q, the set of neighbours of robot i is defined as*

$$\mathcal{N}(i) = \{j \in [m] | d(q_i, q_j) \leq 4 \max\{\max_{p \in Vor_i} d(p, q_i), \max_{p \in Vor_j} d(p, q_j)\}\}$$

where V_i is the Voronoi partition of robot i in the continuous environment.

Remark 3.3.3. The conventional definition of neighbours in the literature [17] is that two robots are neighbours if the intersection of their Voronoi cell boundaries is not empty. Therefore, the distance of two neighbouring robots i and j can be

$$2 \max\{\max_{p \in \mathtt{Vor}_i} d(q_i, p), \max_{p \in \mathtt{Vor}_j} d(q_j, p)\}.$$

In [1], authors show that in environments represented as graphs, with the conventional communication range, a move inside a robots partition might change the partition of the neighbours of their neighbours. Therefore, they assume that the robots communicate with the neighbours of their neighbours, which is analogous to twice the communication range needed to implement the Lloyd's descent-based algorithms in continuous environments. In Section 3.5, we evaluate the performance of the algorithm with both the conventional and our definition of neighbours. •

We extend the definition of neighbouring robots in a continuous environment to capture the cases where the graph instance for the discrete coverage problem is given and the underlying continuous environment is unknown. Consider a graph instance $G = (\mathcal{V}, \mathcal{E}, c)$, then for each edge $(u, v) \in \mathcal{E}$ we add a dummy vertex z with zero weight and replace edge (u, v) by two edges (u, z) and (z, v) such that $c(u, v) = 2c(u, z) = 2c(z, v)$. We let $c(u, v)$ for $(u, v) \notin E$ be the length of the shortest path in G between u, v. Let W_i be the partition of robot i, in the resulting graph. Then the equivalent definition of the neighbouring robots is given as follows:

Definition 3.3.4 (Neighbour robots in graphs)**.** *Given a configuration Q, the set of neighbours of robot i is defined as*

$$\mathcal{N}(i) = \{j \in [m] | c(q_i, q_j) \leq 4 \max\{\max_{p \in W_i} c(p, q_i), \max_{p \in W_j} c(p, q_j)\}\}.$$

3.3.2 Detailed Description

We are now ready to provide a detailed description of the proposed algorithm.

Algorithm Framework (Algorithm 3.1): For the ease of presentation, we provide a description of the algorithm in which the robots perform local moves sequentially. A remark about how the algorithm can be implemented with parallel execution of the local moves is given in the supplementary material. Each robot is assigned a unique identifier UID. Starting from an active robot, say robot i, the robot will make the possible local moves using Algorithm 3.2. If it can not make a local move, the robot will become inactive and will send a completion message to the neighbouring robots via SendCompletionMessage (line 6 of Algorithm 3.1). After execution of the local move by a robot, the robot becomes inactive. The next active robot to execute the local move can be selected in a distributed manner using a token passing algorithm [61], or any other method that ensures each robot gets a turn at making a local move. The process terminates when all the robots become inactive.

Local Move of Robot i (Algorithm 3.2): At an iteration of Algorithm 3.2, let the current configuration of robots be given by $Q = \{q_1, \ldots, q_m\}$ where the vertices in the partition of robot i are given by W_i. The robot i considers moving to a vertex $v \in W_i$ from its current vertex q_i. This move can only change the sensing cost of the vertices in the neighbouring robots' partitions (See Lemma 3.4.1 in Section 3.4). Hence, the robot i can calculate the new neighboring partitions after a potential move to v. In line 2 of Algorithm 3.2, robot i calculates the change in local objective δ_v due to this move for all $v \in W_i$. Since only robot i is executing Algorithm 3.2 at the current time, δ_v also represents the change in the global objective function. If $\min_{v \in W_i} \delta_v \leq -\epsilon_0$, robot i moves to $q'_i = \arg\min \delta_v$ and the iteration terminates (local move type 1). If there is no valid local move of type 1, then

Algorithm 3.1 DistributedCoverageAlgorithm

```
1: Each robot sets itself to active
2: while there exists an active robot do
3:     for any active robot $i \in \{1, \ldots, m\}$ do
4:         if $\sum_{u \in W_i} c(u, q_i) > 0$ then
5:             LocalMove($i$)
6:             SendCompletionMessage()
7:         end if
8:         Robot $i$ deactivates
9:     end for
10: end while
```

robot i calculates the change in local objective if it moves to v and a new robot appears at q_i, i.e.,

$$\rho_v = \sum_{j \in \mathcal{N}(r)} \sum_{u \in R_j(v)} w(u)[c(u, v) - c(u, q_j)], \tag{3.5}$$

where $R_j(v) = \{u \in W_j | c(u, q_j) > c(u, v)\}$ represents the vertices in the partition of robot j that are closer to v than the robot j at q_j. Robot i then passes the message with the set $\Gamma_i = \{\rho_v | v \in W_i\}$ and a counter set to 1 to all its neighbors (line 7 of Algorithm 3.2) and waits for a response (line 8 of Algorithm 3.2). If the response is a rejection from all the neighbors, Algorithm 3.2 terminates. Otherwise it selects the acceptance message with the largest change in the objective and moves to the corresponding vertex. It also sends an acknowledgement message to the neighbor k whose message was selected so that robot k can move to q_i.

Response of Other Robots (Algorithm 3.3): When a robot k receives messages from its neighbors, it follows Algorithm 3.3. Since messages from only one sender robot are propagating through the system at any time, it can select the message with the smallest counter value if it receives messages from multiple neighbors. The neighbor who sent the message with the smallest counter value is called the parent of robot k. It sends back a rejection message to all other neighbors. If the counter value of the message was one, it means that the message originated from its neighboring robot, say i. Then robot k calculates the change in the sensing costs of the vertices in $W_k \setminus R_k(v)$ if it moves to vertex q_i and robot i moves to vertex v resulting in configuration $Q' = \{q_j | j \in \mathcal{N}(k)\} \cup \{v\}$, i.e.,

$$\ell_v = \sum_{u \in W_k \setminus R_k(v)} \min_{q \in Q'} [c(u, q) - c(u, q_k)] w(u), \tag{3.6}$$

Algorithm 3.2 LocalMove(i)

```
1:  while $\exists$ a local move do
2:      Calculate $\delta_v$ for all vertices in $W_i$
3:      if $\min_v \delta_v \leq -\epsilon_0$ then
4:          Move to $v$
5:      else
6:          Calculate $\Gamma_i = \{\rho_v | v \in W_i\}$          ▷ Using Equation (3.5)
7:          SendMessage($\Gamma_i$, 1)
8:          ReceiveMessage()
9:          SendAcknowledgement()
10:     end if
11: end while
```

If $\min_v(\rho_v + \ell_v) \leq -\epsilon_0$, robot k decides to move to q_i and sends an acceptance message to robot i with the vertex $\arg\min_v \rho_v + \ell_v$ and the change associated with this move. Otherwise it increments the counter and sends the message with Γ_i to its neighbors. If the counter value in a message is greater than one, robot k calculates ℓ as follows:

$$\ell = \sum_{u \in W_k} \min_{j \in \mathcal{N}(k)} [c(u, q_j) - c(u, q_k)] w(u) \tag{3.7}$$

If $\min_v(\rho_v + \ell) \leq -\epsilon_0$, robot k sends an acceptance message back to its parent. Otherwise it increments the counter and sends the message to its neighbors.

In function ReceiveMessage in line 8 of Algorithm 3.2 and line 16 of Algorithm 3.3, if any robot receives at least one acceptance message from its neighbors, it passes the message with lowest increase in sensing cost value to its parent. If it receives rejection messages from all its neighbors, it sends back a rejection message to its parent. Robot i selects the move with maximum improvement in the sensing cost and sends back an acknowledgment using SendAcknoledgment to the accepted messages. Robots that receive the acknowledgment move to their parent's location. If there is no more local move available, robot i sends a completion message using SendCompletionMessage to the neighbouring robots which will be propagated to all the robots. Then the next active robot executes LocalMove.

Algorithm 3.3 RECEIVER

```
    Input: message = ($\Gamma_i$, MessageCounter)
 1: if MessageCounter = 1 then
 2:     Calculate $\ell_v$ for all $v$ in the message                ▷ Equation (3.6)
 3:     if $\exists v$ in the message with $\rho_v + \ell_v \leq -\epsilon_0$ then
 4:         send acceptance message to parent
 5:     else
 6:         SendMessage($\Gamma_i$, MessageCounter + 1)
 7:     end if
 8: else
 9:     Calculate $\ell$                                              ▷ Equation (3.7)
10:     if $\exists v$ in the message with $\rho_v + \ell \leq -\epsilon_0$ then
11:         send acceptance message to parent
12:     else
13:         SendMessage($\Gamma_i$, MessageCounter + 1)
14:     end if
15: end if
16: ReceiveMessage()
```

3.4 Analysis of the Algorithm

In this section, we provide analysis on the quality of the solutions provided by the proposed algorithm. The results showing that the proposed algorithm terminates in finite time is provided in the supplementary materials.

3.4.1 Correctness and Approximation Factor

Prior to providing the main results on the correctness and approximation factor of the algorithm, we provide two results on the change in the sensing cost of vertices in the partitions of the neighbouring robots with a move of a robot.

The following result shows that a move by a robot inside its partition can only change the sensing cost of the vertices in its neighbouring partitions.

Lemma 3.4.1. *Consider a vertex $z \in W_j$ where robot j at position q_j is the closest robot to z. Then robot j is closer to z than any vertex in the partition of a non-neighbour robot,* *i.e.*,

$$c(z, q_j) \leq \min_{i \notin \mathcal{N}(j)} \min_{u \in W_i} c(z, u).$$

Proof. Proof by contradiction. Suppose there exists a move to vertex $v \in W_i$ by robot i that changes the sensing cost of a vertex $z \in W_j$ of a non-neighbour robot j, i.e., $c(v, z) < c(q_j, z)$ which is equivalent to $d(v, z) < d(q_j, z)$ by the monotonicity of function f. By the triangle inequality, we have,

$$d(q_j, v) \leq d(v, z) + d(q_j, z) < 2d(q_j, z). \tag{3.8}$$

Observe that $v \in W_i$, then $c(q_j, v) \leq c(q_i, v)$ which implies $d(q_j, v) \leq d(q_i, v)$. By Equation (3.8), we have

$$d(q_i, q_j) \leq d(q_j, v) + d(q_i, v) \leq 2d(q_j, v) < 4d(q_j, z).$$

Then by the definition of the neighbouring robots in Section 3.2, robots i and j are neighbours. This is a contradiction. □

Also, the following result shows that if a robot i moves anywhere in the graph, then the vertices previously in W_i will be assigned to robot in $\mathcal{N}(i)$.

Lemma 3.4.2. *For any vertex $z \in W_i$, there exists a robot $j \in \mathcal{N}(i)$ at q_j where $c(z, q_j) \leq c(z, q_k)$ for all $k \notin \mathcal{N}(i)$.*

Proof. Suppose there exists $k \notin \mathcal{N}(i)$ and vertex $z \in W_i$ such that $c(z, q_k) < \min_{j \in \mathcal{N}(i)} c(z, q_j)$. Let P be the shortest path from z to q_k. Let p be the point on the path where P intersects the boundary of Voronoi cell of robot i. The point p is not on the boundary of robot k, otherwise the distance between $d(q_i, q_k) \leq 4 \max\{\max_{p \in \texttt{Vor}_i} d(q_i, p), \max_{p \in \texttt{Vor}_k} d(q_k, p)\}$ which is a contradiction. Hence, the point p is on the boundary of a neighbouring robot j. Therefore, there is a path from q_j to z shorter than P, then

$$c(q_j, z) = f(d(q_j, z)) \leq f(d(q_k, z)) = c(q_k, z). \quad \square$$

Then we provide the following result on the change in the global objective with a successful move in the distributed algorithm.

Lemma 3.4.3. *If a local move is accepted by the robot, then the global objective improves by at least ϵ_0.*

Proof. A local move falls under the following cases:

(i) Since, by Lemma 3.4.1, a move of type 1 can only change the sensing cost of the neighbouring robots. Then the result is trivial for the local moves of type 1.

(ii) If the local move consists of a move by the robot i that is executing LocalMove to a vertex v in its partition and a neighbouring robot j moving to vertex q_i. Let Q' be the configuration after the local move, then the change in the global objective $\Delta\mathcal{D} = \mathcal{D}(Q') - \mathcal{D}(Q)$ is given by the following:

$$\Delta\mathcal{D} = \sum_{k\in[m]} \sum_{u\in W_k} \min_{q\in Q'} w(u)c(u,q) - \sum_{k\in[m]} \sum_{u\in W_k} w(u)c(u,q_k). \tag{3.9}$$

They by Lemma 3.4.1 and 3.4.2, the sensing cost changes only for vertices in $u \in \cup_{k\in\mathcal{N}(i)} W_k$, therefore,

$$\begin{aligned}\Delta\mathcal{D} &= \sum_{k\in\mathcal{N}(i)} \sum_{u\in W_k} w(u)[\min_{q\in Q'} c(u,q) - c(u,q_k)] \\ &= \sum_{k\in\mathcal{N}(i)} \sum_{u\in R_k(v)} w(u)[c(u,v) - c(u,q_k)] \\ &+ \sum_{k\in\mathcal{N}(i)} \sum_{u\in W_k\setminus R_k(v)} w(u)[\min_{q\in Q'} c(u,q) - c(u,q_k)].\end{aligned}$$

Observe that the sensing cost for vertex $u \in W_k \setminus R_k(v)$ for robot k at $q_k \in Q' = \mathcal{N}(i) \cup \{v\} \setminus \{j\}$ does not change, therefore, we have

$$\Delta\mathcal{D} = \sum_{k\in\mathcal{N}(i)} \sum_{u\in R_k(v)} w(u)[c(u,v) - c(u,q_k)] + \sum_{u\in W_j\setminus R_j(v)} w(u)[\min_{q\in Q'} c(u,q) - c(u,q_j)].$$

Hence, the result follows immediately as $\Delta\mathcal{D} = \rho_v + l_v$.

(iii) Suppose the local move consists of a move by the robot i that is executing LocalMove to a vertex v in its partition and a sequence of moves between the neighbouring robots. Without loss of generality, let $\langle v, q_i, q_{i+1}, \ldots, q_{j-1}, q_j\rangle$ be the sequence of moves between the neighbouring robots where each robot moves to the previous vertex of the preceding robot in the sequence. Let Q' be the configuration after the local move, then the change in the global objective is given by Equation (3.9). Observe that each robot accepts only the message from the parent robot. Therefore, among the neighbours of the robots in the sequence only the parent of each robot moves.

First we show that the change in the sensing cost under this sequence of moves only occurs for vertices in $\cup_{k\in\mathcal{N}(i)}W_k$ and vertices in W_j. Let u be a vertex assigned to robot $k' \in [m] \setminus \{j \cup \mathcal{N}(i)\}$ in configuration Q, i.e., $u \in W_{k'}$. Since $k' \notin \mathcal{N}(i)$, then by Lemma 3.4.1 a move to vertex v will not improve the sensing cost of u. Also if k' is among the robots moving in the sequence, then there is a robot moving to its previous location, therefore, each vertex in $W_{k'}$ will be sensed by another robot with the same sensing cost. Therefore, the total change $\Delta\mathcal{D}$ in the sensing cost of the vertices becomes

$$\sum_{k\in\mathcal{N}(i)}\sum_{u\in R_k(v)} w(u)[c(u,v)-c(u,q_k)] + \sum_{u\in W_j} w(u)[\min_{q\in Q'} c(u,q) - c(u,q_j)].$$

Hence, the result follows immediately as $\Delta\mathcal{D} = \rho_v + l$.

□

Now we show the following result on the valid local moves in CentralizedAlg given the final configuration of the proposed distributed algorithm.

Lemma 3.4.4. *If the proposed distributed algorithm terminates, then there is no single swap move in the centralized local search algorithm* CentralizedAlg *(see Section 3.2.2) that improves the objective function.*

Proof. Suppose that there exists a centralized local move of robot j at vertex q_j to a vertex $v \in W_i$ that improves the objective function by ϵ_0. Therefore, adding a robot to v improves the sensing cost of the vertices in $\cup_{k\in\mathcal{N}(i)}W_k$ by $\rho_v \le -\epsilon_0$. Therefore, by construction of the algorithm, the robot i would have suggested the move to its neighbouring robots. Suppose after l communications, robot j at q_j receives the message for the first time. In Line 2 (resp. Line 9) of Algorithm 3.3 if $j \in \mathcal{N}(i)$ (resp. $j \notin \mathcal{N}(i)$), robot j calculates the increase in the sensing cost ℓ_v (resp. ℓ) for the vertices in $\cup_{k\in\mathcal{N}(j)}W_k$ by the move from q_j to the parent of robot j. Since robot j has rejected this offer, by Lemma 3.4.3 the change in the global sensing cost is less than ϵ_0. This is a contradiction. □

Theorem 3.4.5. *The proposed distributed coverage control algorithm provides a solution within $5 + o(\epsilon_0)$ factor of the optimal configuration.*

Proof. The result follows immediately from Lemma 3.4.4. The final configuration in the distributed algorithm is a locally optimal solution for the CentralizedAlg with single swap at each iteration, i.e. $p = 1$, therefore, the configuration provides a coverage within a factor $5 + o(\epsilon_0)$ of the global optimal. □

Corollary 3.4.6. *Given an environment $\mathcal{X}$ with m mobile robots and a sampling of $\mathcal{X}$ with dispersion ζ, the solution Q obtained from the proposed distributed coverage control algorithm provides coverage cost $\mathcal{H}(Q) \leq 5\mathcal{H}(S^*) + o(f(\zeta) + \epsilon_0)$, where S^* is the optimal solution of the continuous coverage problem.*

Proof. Proof follows immediately from Theorems 3.2.3 and 3.4.5. □

3.4.2 Time Complexity

In this section, we characterize the runtime and the communication complexity of the proposed algorithm.

Let Q_0 be the starting configuration of the robots and Q^* be the optimal configuration for problem of coverage on graph G, then we have the following result on the runtime of the proposed algorithm. For $\epsilon_0 = 0$, we follow the the analysis similar to [1]. Since there are a finite number of possible local moves that improve the global sensing cost, each iteration improves the global sensing cost by at least $\epsilon' > 0$. Therefore the algorithm terminates in $\frac{\mathcal{D}(Q_0)-\mathcal{D}(Q^*)}{\epsilon'}$ iterations. Observe that with $\epsilon_0 = 0$, the distributed algorithm provides a solution within 5 factor of the optimal solution with possibly non-polynomial number of iterations. However, we can prove convergence in polynomial time if the sampling of the environment $\mathcal{S}$ satisfies the following properties.

Assumption 3.4.7. *The weight of a vertex v in $\mathcal{S}$ is at least w_0 for some $w_0 > 0$, i.e., $\int_{p\in\sigma(v)} \phi(p)dp \geq w_0 > 0$.*

This follows the common assumption in the coverage control literature where there is a basis function defined for ϕ [17]. For a given w_0, we remove a vertex v with with weight $w(v) < w_0$ from the samples and recalculate the weights on the vertices.

Assumption 3.4.8. *The ratio $\max_{u,v\in\mathcal{S}} f(d(u,v))/\min_{u,v\in\mathcal{S}} f(d(u,v))$ is polynomial in the number of the samples $|\mathcal{S}|$.*

For instance, if a graph is constructed via a grid sampling of the continuous environment, where each cell is a $d \times d$ square, then we have

$$\frac{\max_{e\in E} c(e)}{\min_{e\in E} c(e)} = \frac{f(\sqrt{2|\mathcal{S}|}d)}{f(d)} = O(\sqrt{|\mathcal{S}|}),$$

where the second equality is by the sub-additivity of sensing function f.

For a given $\epsilon > 0$ and a polynomial $p(|\mathcal{S}|, m)$, we have,

Lemma 3.4.9. *The proposed algorithm with* $\epsilon_0 = \frac{\epsilon w_0}{p(|\mathcal{S}|,m)} \min_{e\in E} c(e)$ *terminates in the polynomial number of iterations, i.e.,*

$$\log(\mathcal{D}(Q_0)/\mathcal{D}(Q^*))/\log(1 - \frac{\epsilon w_0}{p(|\mathcal{S}|, m)} \frac{\min_{e\in E} c(e)}{\max_{e\in E} c(e)}).$$

Proof. Let Q_i be the configuration of the robots at step i of the algorithm. As the global sensing cost improves by at least ϵ_0 after each iteration, we have,

$$\mathcal{D}(Q_{i+1}) - \mathcal{D}(Q_i) \leq -\epsilon_0 = -\frac{\epsilon w_0}{p(|\mathcal{S}|, m)} \min_{e\in E} c(e).$$

Observe that $\mathcal{D}(Q_i) \leq \max_{e\in E} c(e)$. Therefore,

$$\mathcal{D}(Q_{i+1}) \leq \mathcal{D}(Q_i) - \frac{\epsilon w_0}{p(|\mathcal{S}|, m)} \min_{e\in E} c(e) \leq \mathcal{D}(Q_i)(1 - \frac{\epsilon w_0}{p(|\mathcal{S}|, m)} \frac{\min_{e\in E} c(e)}{\max_{e\in E} c(e)}).$$

Therefore, at each iteration of the distributed coverage algorithm the sensing cost improves by the factor of $1 - \frac{\epsilon w_0}{p(|\mathcal{S}|,m)} \frac{\min_{e\in E} c(e)}{\max_{e\in E} c(e)}$. Hence, the algorithm terminates in

$$\log(\mathcal{D}(Q_0)/\mathcal{D}(Q^*))/\log(1 - \frac{\epsilon w_0}{p(|\mathcal{S}|, m)} \frac{\min_{e\in E} c(e)}{\max_{e\in E} c(e)})$$

iterations which is polynomial in the input size, $1/\epsilon$ and $1/w_0$. □

Remark 3.4.10. At each iteration of the algorithm, at most m^2 messages are sent with size at most $n \log(\max_{e\in E} c(e)) + \log(m)$ bits by the robot executing LocalMove. Then at most m^2 messages are sent back between the robots in the acceptance/rejection step. Finally, the SendAcknowledgment step requires at most m messages. Hence, the communication complexity of the proposed distributed algorithm is

$$O(\log(\mathcal{D}(Q_0)/\mathcal{D}(Q^*))/\log(1 - \frac{\epsilon w_0}{p(|\mathcal{S}|, m)} \frac{\min_{e\in E} c(e)}{\max_{e\in E} c(e)})m^2)$$

. •

3.5 Simulation Results

In this section, we evaluate the performance of the proposed distributed algorithm and compare it to convex and non-convex distributed coverage algorithms and the centralized algorithm in [31]. To construct the discrete problem described in Section 3, we use a grid sampling of the environment. We denote the maximum distance inside a Voronoi cell of a robot i by $R_{\text{comm}} = \max_{p \in \texttt{Vor}_i} d(q_i, p)$ and we evaluate the performance of the proposed algorithm with communication ranges of $4R_{\text{comm}}$ (See Definition 1) and $2R_{\text{comm}}$ which is analogous to the conventional communication model in the continuous coverage literature. In the rest of this section, we use $f(x) = x$ as the sensing cost function.

3.5.1 Convex Environments

In this experiment, we compare our algorithm to distributed Lloyd's algorithm [17] in convex environments. We use the Euclidean distance as the metric between two points. The comparison is conducted in a 1500×850 environment with 100 different event distributions. The event distributions are truncated multivariate normal distributions with mean $[1400, 800]$ and covariance matrices $\Sigma = [\sigma, 0; 0, \sigma]$ where σ is uniformly randomly selected from interval $[5, 10] \times 10^4$. In this experiment, the robots are initialized in the bottom left corner of the environment. Figure 3.4 illustrates the percentage difference of the solutions provided by the two algorithms with respect to the solution of the centralized algorithm. Observe that the proposed aolgorithm achieves solution quality very close to the centralized algorithm, even with a large number of robots, while Lloyd's algorithm provides solutions with approximately 15% deviation.

Figure 3.5 shows the percentage share of the different local move types. The dashed lines show the results for the proposed algorithm with $2R_{\text{comm}}$ communication range. The single hop Move type II, is a local move which involves a robot and its neighbours in comparison to the multi-hop local move which involves non-neighbour robots. Observe that the majority of the local moves are Move Type I which only require communication with the neighbouring robots. However, the local moves of type II help the proposed algorithm to leave locally optimal solutions.

Figure 3.6 illustrates the improvement in the sensing cost for the Lloyd's and the proposed algorithm in a convex environment with uniformly random initial configurations. We used the solution obtained from the centralized algorithm as the reference point and compared the deviation from the reference for the distributed algorithms. The results are obtained for 50 initial configurations in the environment. Observe that even with the large number of robots where the random configuration provides a relatively good sensing cost, the proposed algorithm improves the solution by 50% on average. In a system of 40 robots, our proposed algorithm provided $\approx$ 15% additional improvement on the sensing cost as compared to the Llyod's algorithm. Figure 3.7 shows the final configuration and the paths

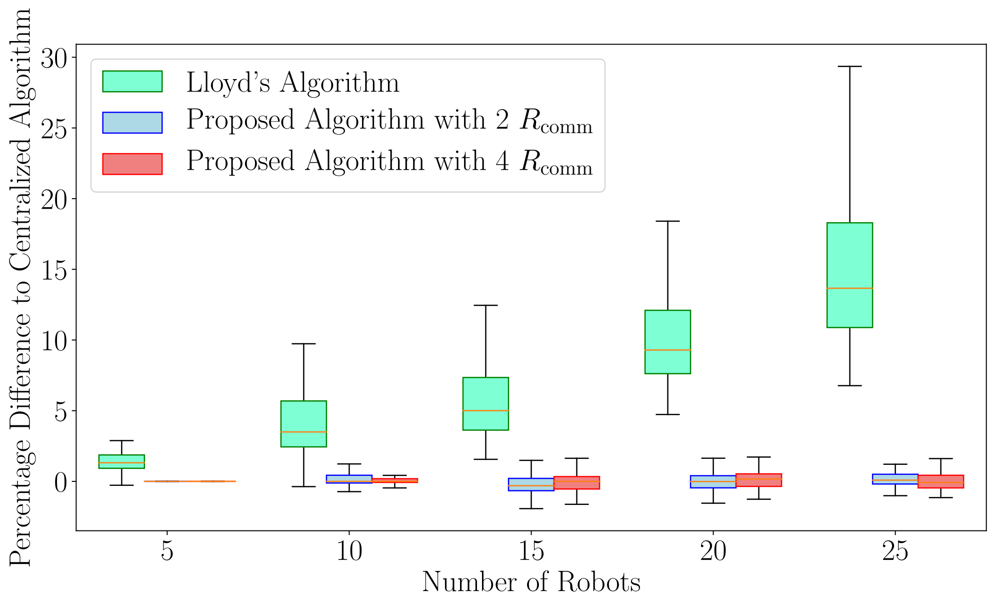


Figure 3.4: Percentage difference of the solutions of different algorithms to the solution of the centralized algorithm

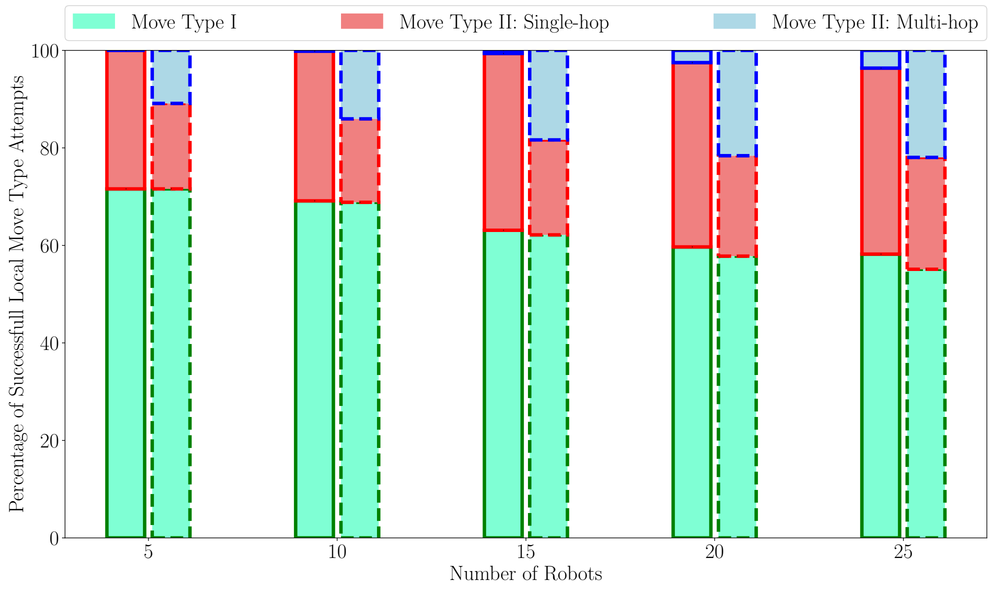


Figure 3.5: Percentage of Successful Local Move Type Attempts

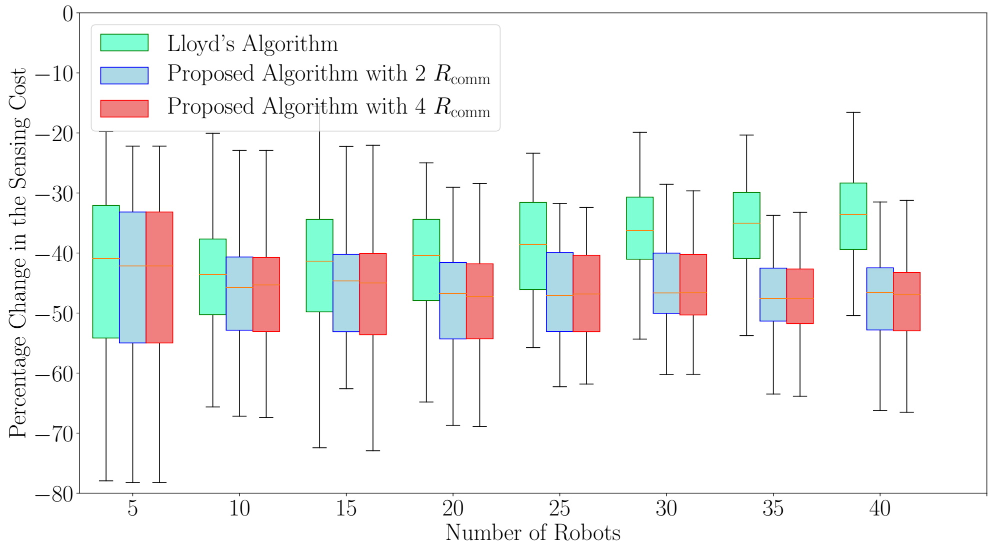


Figure 3.6: Percentage improvement of different algorithms with random initial configurations.

of 10 robots for the two algorithms in a test environment. The partitions are the Voronoi partition for the final configuration of the robots.

3.5.2 Non-Convex Environments

In this section, we compare the solution quality of the proposed algorithm with two different communication ranges to the algorithms in [2], [1] and the centralized algorithm [31]. The experiment is conducted in a 1500×850 environment that contains obstacles (See Figure 3.8a), and using 100 different event distributions. The distributions are generated in the same manner as in the convex environment experiments with uniformly random mean and covariance matrices. The communication model in the non-convex studies are different, for instance, two robots are neighbours in [2], if the intersection of the Voronoi cells of the robots in the environment without obstacles is non-empty, and two robots are neighbours in [1] if the two partitions of the robots share an edge in the discrete representation of the environment. Therefore in the implementations of algorithms in [2] and [1], we assume that robots are connected to every other robot. Figure 3.9 shows the percentage difference between the solutions of each algorithm compared to the centralized algorithm. Observe that the proposed algorithm even with the conventional communication range out-performs both other algorithms by $\approx 20\%$ on average in a system with 30 robots and matches the solution quality of the centralized algorithm. Figure 3.10 illustrates the

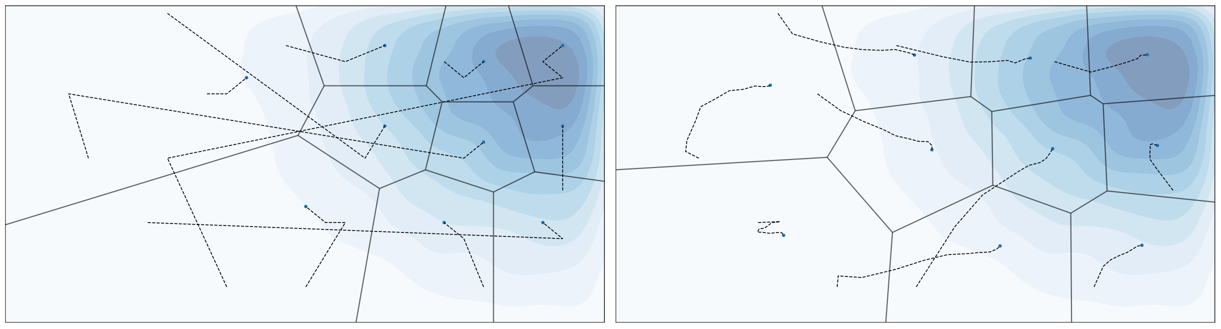

(a) Proposed Algorithm with $4R_{\text{comm}}$ communication range. (b) Lloyd's Algorithm

Figure 3.7: Final Configuration and the paths of the robots for the two algorithms in a test environment

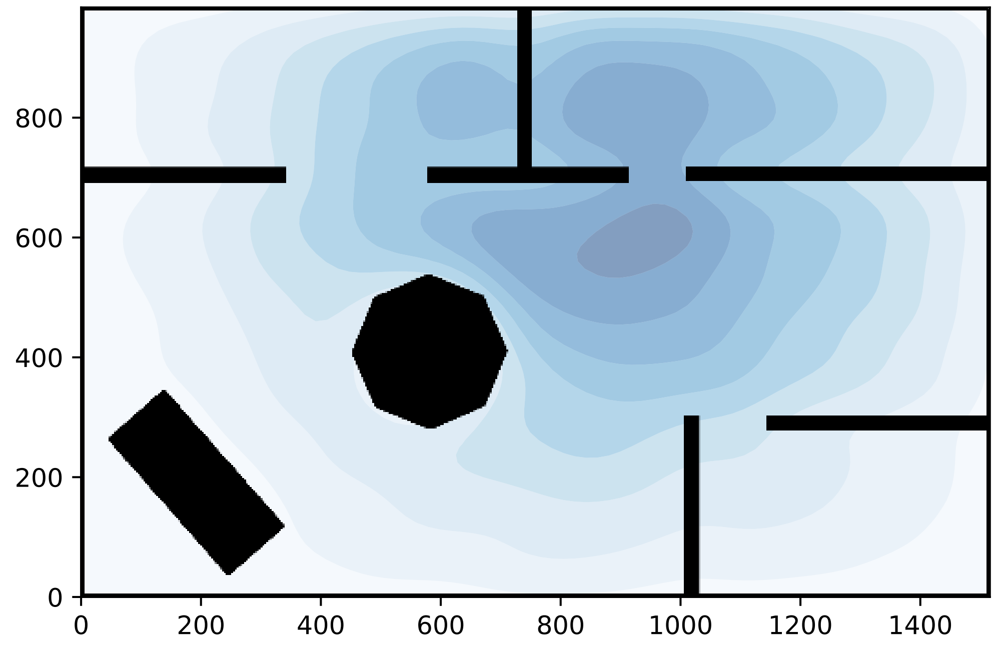
final configuration and the movement of the robots using the proposed algorithm in a non-convex environment.

3.5.3 Examples of Different Local Solutions

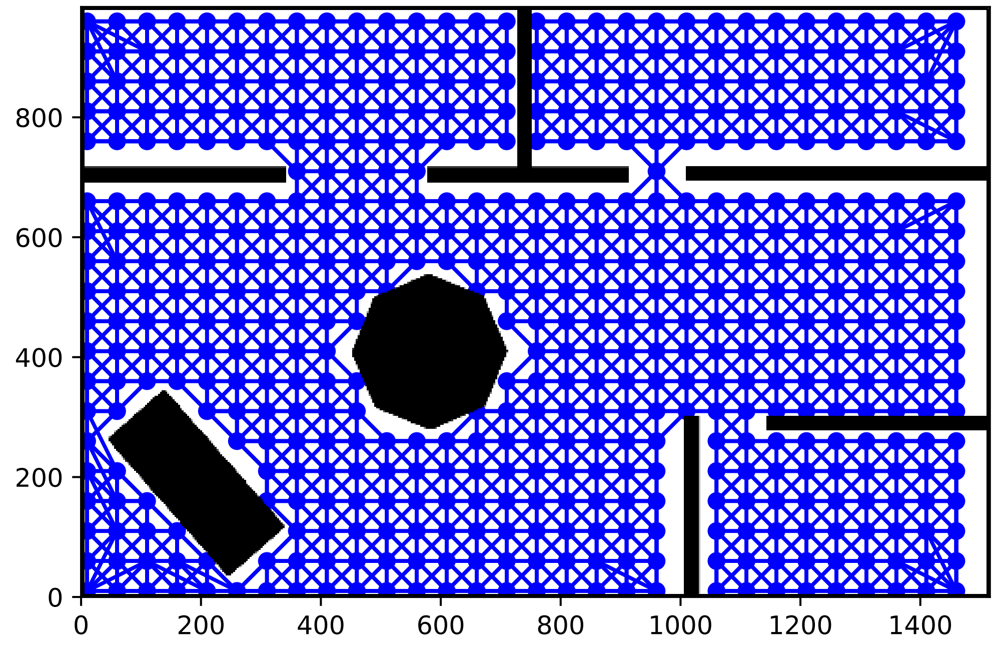
In this section, we provide two examples for the algorithms in [1] and [2]. We illustrate the locally optimal solution reached using these algorithms, and the local moves considered in the proposed algorithm which helps escaping these sub-optimal solutions.

Consider a discrete coverage problem with 4 vertices and 3 robots initialized at the configuration shown in Figure 3.11a. The bars on the vertices of the graph represents the weight of the vertices. By the communication model in [1], all the robots are neighbours of each other. The local move in algorithm in [1] moves the robots inside their partitions if the move improves the sensing cost of its partition and the neighbouring partitions. Note that the initialized configuration of the robots is a locally optimal solution for the algorithm in [1]. However, in the proposed algorithm the robots will improve on current configuration with performing single-hop move type II. Figure 3.11b shows the final configuration with the proposed algorithm.

Figure 3.12 shows a continuous environment with a single robot. The sensing cost of an event is a function of the geodesic distance from the robot. The high-level idea of the algorithm in [2] is to find the centroid in the environment without obstacles (see t_{virt} in Figure 3.12) and if the centroid is inside an obstacle, then the algorithm projects the centroid to a face of the obstacle (see t_{real} in Figure 3.12) and moves the robot towards t_{real}. Observe that in the scenarios where the sensing cost is a function of the length of the



(a) Sample Environment with Obstacles



(b) Graph representation of the Environment

Figure 3.8: A sample environment with obstacles and the discrete representation of the environment.

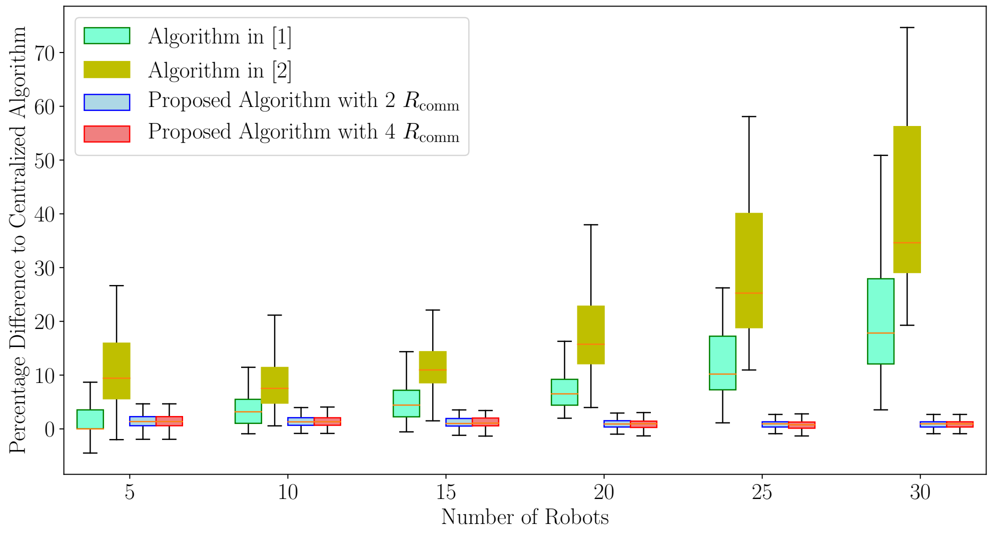


Figure 3.9: Percentage difference of the solutions of different algorithms to the solution of the centralized algorithm

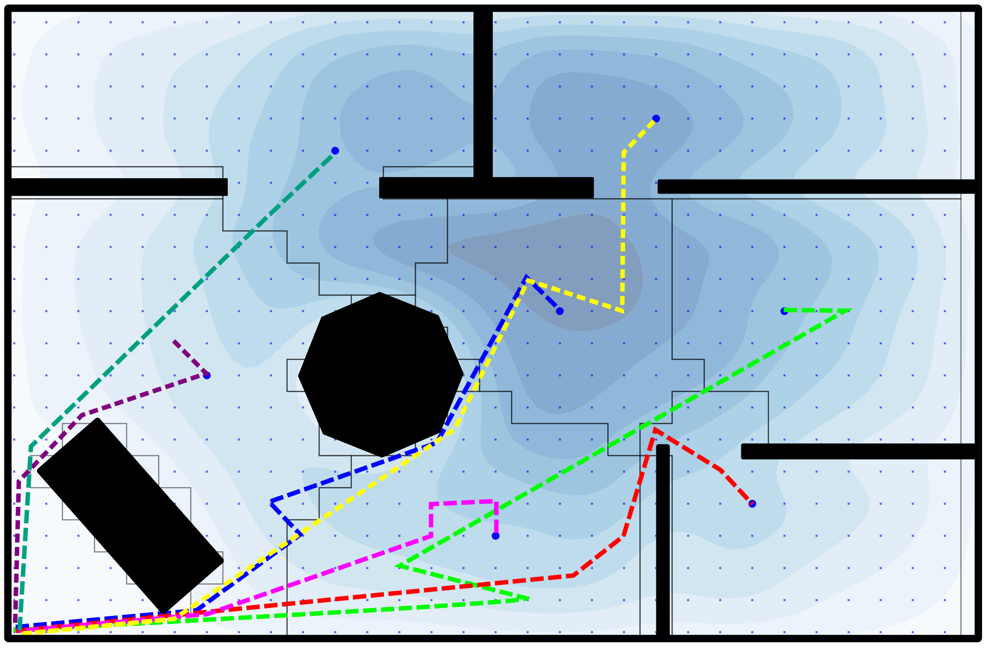
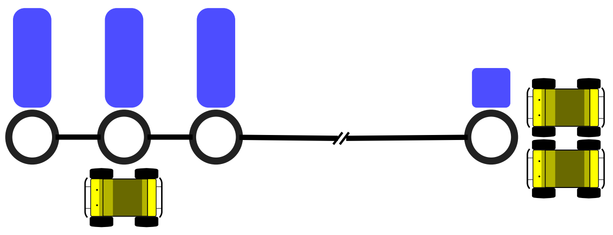
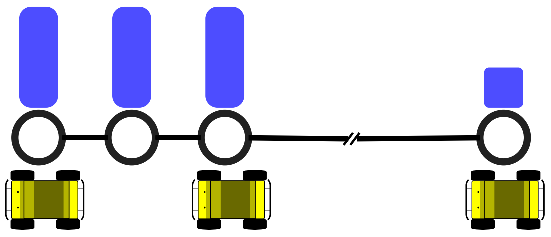


Figure 3.10: Robot movements in a non-convex environment using the proposed distributed algorithm



(a) A test environment for algorithm in [1]



(b) Final configuration with proposed algorithm

Figure 3.11: A discrete test environment and the final configurations of algorithm in [1] and the proposed algorithm

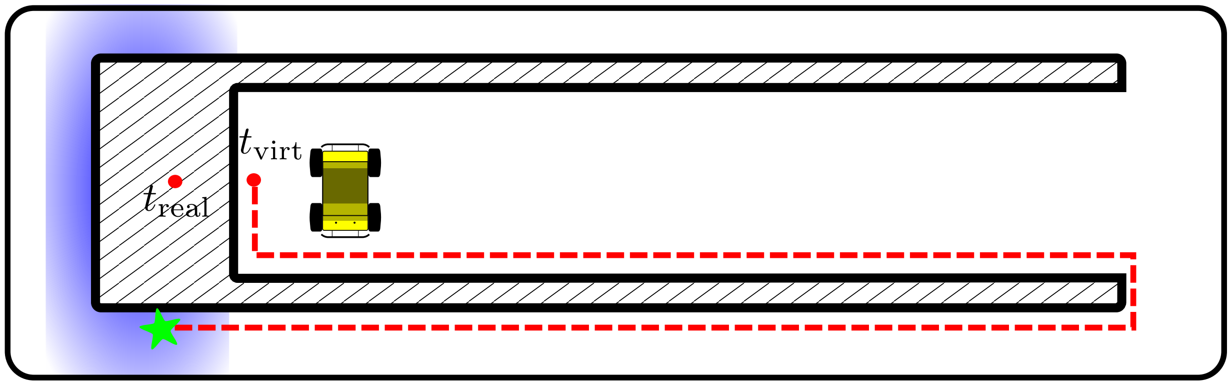


Figure 3.12: A test environment for Algorithm in [2]

shortest path between the robot and the event location, the projection of the centroid may result in sub-optimal solutions. However, the proposed algorithm avoids these scenarios by solving the coverage problem on the discrete representation of the environment.

Chapter 4

Heterogeneous Multi-Robot Coverage

In this chapter, we focus on the problem of deploying multiple heterogeneous robots to cover an environment with different event types. Each event type has a different spatial distribution in the environment and can be sensed only with a specific type of sensor. The robots are heterogeneous in that each of the robots is equipped with a subset of sensors. Thus, each robot is capable of measuring a subset of event types. The quality of sensing for an event at a location is a non-decreasing function of the distance of the sensor from the location. The objective is to deploy the robots to maximize the total coverage quality of the events of different types. This appears in several applications including reconnaissance, surveillance [4], and monitoring [5], as well as in the deployment of robots with different capacities [62, 7, 63].

The contributions of this chapter as follows. In Section 4.1, we review the literature on the heterogeneous multi-robot coverage problem. In Section 4.2, we propose a new formulation for the problem of coverage control for heterogeneous mobile robots, which is a natural extension of the homogeneous formulation. We provide a distributed algorithm for minimizing the sensing cost with the new formulation in Section 4.3. In Section 4.4, we extend the formulation and the distributed algorithm to capture discrete environments with different event types. Section 4.5 consists of an extensive set of experiments to evaluate the performance of the proposed algorithms in continuous and discrete environments.

4.1 Related Work

In Section 3.1, we reviewed the recent studies on the distributed multi-robot coverage with homogeneous robots. Several studies address different types of heterogeneity in robots. In [18], the authors consider the problem of sensing an event where the sensors have different functions governing their sensing cost. The approach defines a generalized Voronoi partition based on the sensing functions and provides a Lloyd descent type algorithm for controlling each robot. Other types of heterogeneities for mobile sensors addressed in the literature include the sensors with different sensing ranges [19, 64, 20] and different additive weights on the sensing cost of robots [21]. In [19], the authors address the coverage problem for circular sensors with different radii. In [20], authors provide a gradient descent algorithm for the distributed control of circular sensors with different radii in a non-convex environment.

All the aforementioned studies consider the coverage control for a single event type. In [44], the authors introduced the coverage control problem for multiple event types, each with a different density function. An event of a certain type can be sensed by a robot if the robot is equipped with the required sensor. The proposed approach considers a Voronoi partition generated by all the robots, and the objective function is a convex combination of the sensing cost of each robot for the events inside its Voronoi cell and the sensing cost over the whole environment. In this chapter, we consider the same heterogeneous problem, but we pose a different objective function, and thus distributed control law. As such, we compare our approach to [44] in detail, and demonstrate its advantages in simulation. In our approach, we define a Voronoi partition of the environment for each event type, each generated by the robots with suitable sensors. With these partitions, we are able to ensure that each event is sensed by the closest robot with the required sensor. Moreover, our formulation captures the case where sensors for different events have different sensing functions.

Several studies considered the coverage control in an environment where the event density is unknown prior to deployment [65, 66]. In [65], the authors assume a basis function approximation for the event density with adaptive weights for each robot and propose a decentralized algorithm that updates the weights estimating the true density. The authors in [66] provide a simple stochastic gradient descent algorithm without estimating the real event density and prove convergence to a locally optimal solution. Built on the results in [66], we extend our analysis to environments with multiple events types each with an unknown density.

In [43], a distributed algorithm is proposed for partitioning and coverage of a non-convex environment. Yun and Rus [1] presented a distributed vertex swap algorithm for the robots that converges to locally optimal solutions with two-hop communication. To the best of our knowledge, the existing literature on the distributed coverage control on graphs is limited to homogeneous robots sensing a single event type. On the contrary, we consider multiple event types with heterogeneous robots both in the sensing capability and quality.

4.2 Problem Formulation

In this section, we formulate the coverage problem for continuous and discrete environments. We consider a set of m robots in an environment with k different event types. Each event type is measured with a sensor from a set of k sensors `SensorSet`, and a robot i is equipped with a subset of these sensors, i.e., $S_i \subseteq \texttt{SensorSet}$. The objective is to position the robots in the environment minimize the total sensing cost of k event types.

4.2.1 Continuous Environments

Consider a convex environment $\mathcal{X} \subset \mathbb{R}^2$. The density function of an event type j is $\phi_j : \mathcal{X} \to \mathbb{R}_+, j \in [k]$, which represents the measure of information or the probability of an event type occurring over $\mathcal{X}$. Let $q_i \in \mathcal{X}$ be the position of robot i in the environment and $Q = \{q_1, \dots, q_m\}$. The sensing cost of a sensor j, denoted by f^j, is a non-decreasing function of the distance of the robot to the measured point. Thus, robot i is assigned to measure the events of type $j \in S_i$ at the points closest to q_i, i.e., for each event type $j \in S_i$ robot i measures the events in the Voronoi cell

$$\texttt{Vor}_i^j = \{p \in \mathcal{X} |\ ||p - q_i|| \leq ||p - q_r||, j \in S_i \cap S_r, \forall r \in [m]\}.$$

In this chapter, in contrast to the previous chapter, we do not restrict the sensing function to sub-additive sensing functions.

For simplicity we let `Vor` denote the set of all the Voronoi cells. Now we define the heterogeneous deployment problem in continuous environments as follows:

Problem 4.2.1 (Coverage in Continuous Environments)**.** *Given a set of m robots with dynamics $\dot{q}_i = u_i\ \forall i \in [m]$ where u_i is the control input, find a set of locations $Q =$*

$\{q_1, \dots, q_m\}$ *that minimizes the total sensing cost function, i.e.,*

$$\mathcal{H}(Q, \texttt{Vor}) = \sum_{i=1}^{m} \sum_{j \in S_i} \int_{\texttt{Vor}_i^j} f^j(||p - q_i||)\phi_j(p)dp. \tag{4.1}$$

The sensing cost measure $\mathcal{H}(Q, \texttt{Vor})$ is the total sensing cost of k event types over the environment by the robots located at $\mathcal{P}$. Contrary to the formulation in [44], by defining k Voronoi partitions for the environment, we ensure that an event of type j at location q is sensed by the closest robot with the required sensor. Moreover, the definition of $\mathcal{H}(\mathcal{P}, \texttt{Vor})$ captures different sensing cost function f^j for each event type j. Observe that with $k = 1$, the cost measure $\mathcal{H}(\mathcal{P}, \texttt{Vor})$ becomes the sensing cost for homogeneous robots as proposed in [17]. Figure 4.1 illustrates an instance of the coverage problem with two event types.

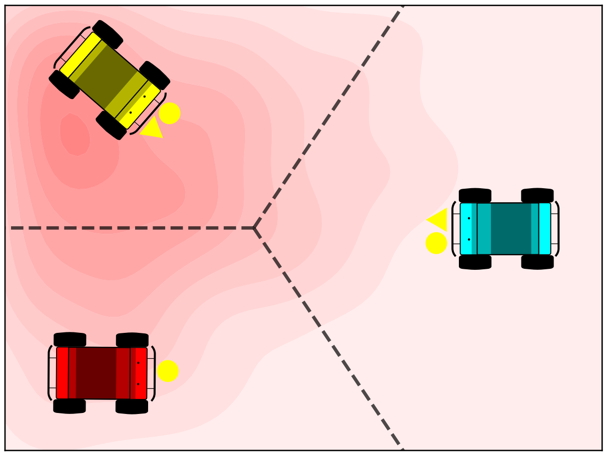
In [44], the authors considered the same heterogeneous coverage problem, but with the objective

$$\mathcal{H}_{het}(\mathcal{P}) = \kappa \sum_{i \in [m]} \int_{\texttt{Vor}_i} ||p - q_i||^2 \phi_{S_i} dq + (1 - \kappa) \sum_{i \in [m]} \int_{\mathcal{X}} ||p - q_i||^2 \phi_{S_i} dq$$

where $\kappa \in (0, 1]$, $\texttt{Vor}_i$ is the Voronoi partition generated by the positions of all robots and $\phi_{S_i} = \sum_{j \in S_i} \phi_i^j$. To illustrate the differences with the proposed objective in Equation (4.1), Figure 4.2 shows a coverage problem on a line with three event types. Each robot senses the event type of the same color. The colored polygons represent the density function of each event type. As shown in Figures 4.2a and 4.2b, the objective in [44] has locally optimal configurations in which one event type is covered sub-optimally. This is in contrast to the configuration shown in Figures 4.2c, where a partition is generated for each event type.

4.2.2 Discrete Environments

In the discrete case, the environment is represented by an undirected graph $G = (\mathcal{V}, \mathcal{E}, c)$. The graph could represent a road-map of an environment with obstacles. Let $\mathcal{V}$ be the set of vertices, and let $\mathcal{E}$ be the edge set, which represents the paths between vertices. The cost function $c : \mathcal{E} \to \mathbb{R}_+$ assigns a cost $c(e)$ for traversing each edge $e \in \mathcal{E}$. The events occur on the vertices of the graph, and the function ϕ_u^j gives the mass of event type j at vertex $u \in \mathcal{V}$. Let $d(u, v)$ be the length of the shortest path from u to v on graph G. The goal is to deploy robots to a set of vertices of the graph. We do not consider deployments


(a) Event density of event type 1, i.e., ϕ_1. The sensor capable of sensing this event is shown with a circle on the robots.

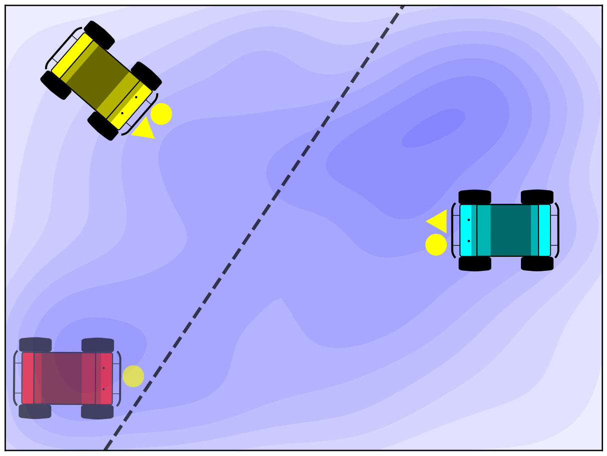
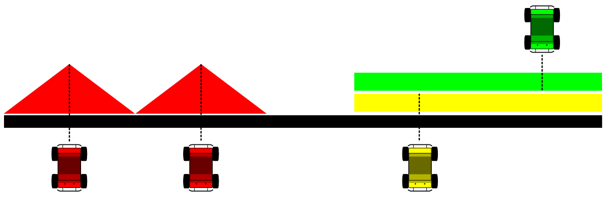
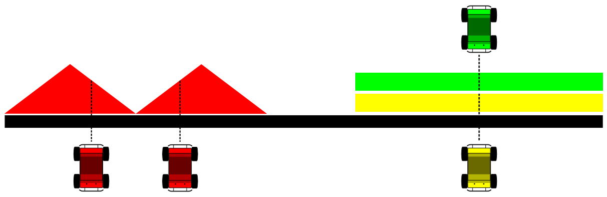

(b) Event density of event type 2. The sensor capable of sensing this event is shown with a triangle on the robots.

Figure 4.1: The Voronoi partitions of an environment for each event type. A robot considered in a Voronoi partition if it is equipped with the required sensor for the event type.


(a) Locally optimal configuration for [44] with $\kappa = 1$


(b) Locally optimal configuration for [44] with $\kappa < 1$

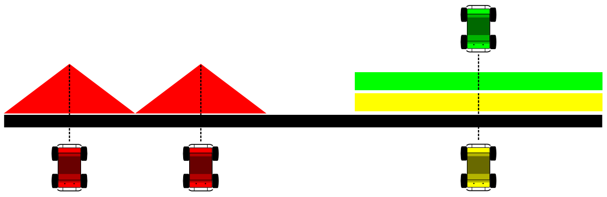

(c) Locally optimal configuration for Equation (4.1)

Figure 4.2: Instance of coverage of three event types on a line. Triangles (resp. rectangle) show the density function for event type 1(resp. 2 and 3). Robots are capable of sensing the event types with the same color.

in which robots are located on the edges of the graph, and this is motivated by the result on the properties of locating robots on vertices given in Lemma 4.4.1.

An event of type j is assigned to the closest robot among the robots that are equipped with the sensor type j. If there exists an event with two equally close robots $i, j \in [m]$, we assign the event to robot i if $i > j$ and robot j otherwise. The subset $W_i^j(\mathcal{P})$ is the set of vertices assigned to robot i to sense events of type $j \in S_i$ on those vertices. Observe that W_i^j is the discrete analogue of $\texttt{Vor}_i^j$.

For simplicity, we let W denote the set of all the subsets W_i^j. Now we define the heterogeneous deployment problem in discrete environments as follows:

Problem 4.2.2 (Coverage in Discrete Environments)**.** *Given a set of m robots find a set of locations on the graph $\mathcal{P} = \{p_1, \dots, p_m\}$ that minimizes the total sensing cost function*

$$\mathcal{H}(\mathcal{P}, W) = \sum_{i=1}^{m} \sum_{j \in S_i} \sum_{v \in W_i^j} f^j(d(q_i, v))\phi_v^j. \tag{4.2}$$

Observe that the objective function is the adaptation of the objective function (4.1) to discrete environments. Also note that with $k = 1$, the cost measure $\mathcal{H}(\mathcal{P}, W)$ becomes the sensing cost for homogeneous robots as proposed in [1].

4.3 Gradient Descent Algorithm for Continuous Environments

In this section, we provide a distributed control law for robots with a proof of convergence to a local minimum of $\mathcal{H}(\mathcal{P}, \texttt{Vor})$ in Problem 4.2.1.

A well-known approach to minimize the sensing cost for each robot to ascend the gradient of $\mathcal{H}$. Let $\delta\texttt{Vor}_i^j$ be the boundary of the Voronoi cell $\texttt{Vor}_i^j$, and let $\delta\texttt{Vor}_{ik}^j$ be the common boundary of the Voronoi cells $\texttt{Vor}_i^j$ and $\texttt{Vor}_k^j$. Let $\mathbf{n}_{ik}^j(q)$ be the unit normal to $\delta\texttt{Vor}_{ik}^j$ at q in the outward direction of $\texttt{Vor}_i^j$. Finally, we define the neighbors of a robot i in Voronoi partition j as $\mathcal{N}_i^j = \{l \in [n] | \delta\texttt{Vor}_i^j \cap \delta\texttt{Vor}_l^j \neq \emptyset\}$.

Then we establish the results on the derivative of $\mathcal{H}$ with respect to the position of robots in Lemma 4.3.1. The results in Lemma 4.3.1 is an extension to the homogeneous case in [17] and the proof follows closely.

Lemma 4.3.1. *For differentiable sensing functions f^j, the derivative of $\mathcal{H}$ with respect to the position of robot i is*

$$\frac{\partial \mathcal{H}}{\partial q_i} = \sum_{i=1}^{m} \sum_{j \in S_i} \int_{Vor_i^j} \frac{\partial f^j(||p - q_i||)}{\partial q_i} \phi_j(q) dq.$$

Proof. By the Leibnitz theorem [67] we have,

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial q_i} =& \sum_{i=1}^{m} \sum_{j \in S_i} \Big(\int_{\mathtt{Vor}_i^j} \frac{\partial f^j(||p - q_i||)}{\partial q_i} \phi_j(q) dq + \sum_{l \in \mathcal{N}_i^j} \int_{\mathtt{Vor}_i^j} f^j(||p - q_i||) \mathbf{n}_{il}^j(q) \frac{\partial(\delta \mathtt{Vor}_{ik}^j)}{\partial q_i}(q) \phi_j(q) dq \\ &+ \sum_{l \in \mathcal{N}_i^j} \int_{\mathtt{Vor}_i^j} f^j(||p - q_i||) \mathbf{n}_{li}^j(q) \frac{\partial(\delta \mathtt{Vor}_{ik}^j)}{\partial q_i}(q) \phi_j(q) dq \Big).\end{aligned}$$

Observe that $\mathbf{n}_{il}^j = -\mathbf{n}_{li}^j$ for all $l \in \mathcal{N}_i^j$. Then the result follows immediately. □

An interesting observation from this is that unlike the homogeneous version of the coverage problem, two robots can share a location in $\mathcal{X}$ if they do not have sensors in common, i.e., the derivative is defined for any $\mathcal{P} \in \mathcal{X}^m \setminus Q_m$ where Q_m is the set of points in which robots with common sensor type overlap, i.e.,

$$Q_m = \{\{q_1, \dots, q_m\} \in \mathcal{X}^m | \exists i, j \in [m] \; q_i = q_j, i \neq j, S_i \cap S_j \neq \emptyset\}.$$

To derive a distributed control law, we first rewrite the derivative of $\mathcal{H}$ in the following form.

$$\frac{\partial \mathcal{H}}{\partial q_i} = 2 \sum_{j \in S_i} \int_{\mathtt{Vor}_i^j} (q_i - p) \frac{df^j(x)}{d(x^2)} \Big|_{||p - q_i||} \phi_j(q) dq. \tag{4.3}$$

Let the mass and centroid of a Voronoi cell respectively be

$$M_{\mathtt{Vor}_i} = \int_{\mathtt{Vor}_i} \frac{df}{d(x^2)} \Big|_{||p - q_i||} \phi(p) dp, \; C_{\mathtt{Vor}_i} = \frac{1}{M_{\mathtt{Vor}_i}} \int_{\mathtt{Vor}_i} p \frac{df}{d(x^2)} \Big|_{||q_i - p||} \phi(p) dp.$$

Then we have $\frac{\partial \mathcal{H}}{\partial q_i} = 2 \sum_{j \in S_i} M_{\mathtt{Vor}_i^j}(q_i - C_{\mathtt{Vor}_i^j})$. Now consider the following simple control law

$$u_i = -c_i \frac{1}{\sum_{j \in S_i} M_{\mathtt{Vor}_i^j}} \sum_{j \in S_i} M_{\mathtt{Vor}_i^j}(q_i - C_{\mathtt{Vor}_i^j}) \quad \forall i \in [n], \tag{4.4}$$

where c_i is a positive gain such that $q_i + u_i$ remains in the convex set $\cap_{j \in S_i} \mathtt{Vor}_i^j$. Robot i is the generator of a cell in each of the Voronoi partitions of the events in S_i, and the control law u_i moves the robot i to the average of the centroids of its Voronoi cells. Observe that computing this control law only requires the communication between robot i and its neighbors in each Voronoi partition. We assume that the communication range is sufficient for the robots to communicate with their neighboring robots.

The following shows the result on the convergence of the proposed distributed control law in Equation (4.4).

Proposition 4.3.2 (Continuous-time Lloyd Descent)**.** *Under control law* (4.4)*, the robots converge to the union of Q_m and the set of critical points of $\mathcal{H}$.*

Proof. Under control law (4.4), if c_i of robot i at some time instance becomes zero then robot i is on the boundary of $\cap_{j \in S_i} \mathtt{Vor}_i^j$ and coinciding with another robot. Therefore, the robots have converged to Q_m. Otherwise, the robots starting from a collision free configuration in $\mathcal{X}^m \setminus Q_m$ remain inside $\mathcal{X}^m \setminus Q_m$ at any time instance, then the set $\mathcal{X}^m \setminus Q_m$ is a positively invariant set under control law (4.4). The derivative of $\mathcal{H}$ with respect to time is

$$\frac{d}{dt}\mathcal{H}(Q, \mathtt{Vor}) = \sum_{i=1}^{m} \frac{\partial \mathcal{H}}{\partial q_i} \dot{q}_i = -\sum_{i=1}^{m} \frac{2c_i}{\sum_{j \in S_i} M_{\mathtt{Vor}_i^j}} (\frac{\partial \mathcal{H}}{\partial q_i})^2.$$

Therefore, observe that the direction of control law (4.4) coincides with the gradient of $\mathcal{H}$. The rest of the proof follows from the proof of Proposition 3.1 in [68], which uses LaSalle's Invariance Principle to show at each step of the algorithm $\mathcal{H}$ monotonically increases and converges to the largest invariant set contained in

$$C_m = \{P \in D^m | (\frac{\partial \mathcal{H}(Q)}{\partial q_i})^2 = 0, \forall i \in [n]\} \qquad \square$$

Although control law (4.4) converges to a set consisting of Q_m, simulation results in Section 4.5 show that by resolving collisions with local a controller, the system remains in $\mathcal{X}^m \setminus Q_m$ and converges to the set of critical points of $\mathcal{H}$.

Remark 4.3.3 (Unknown Density Functions). Consider the scenario where the robots do not have access to the density functions of the event types and they only observe events of different types arriving over time. A control law is proposed in [66] for the homogeneous case which drives the closest robot to the observed event and converge to a locally optimal positions. In the heterogeneous case, each observed event $Z = [z^d, z^c]$ is a random vector consisting of a discrete component $z^d \in [k]$, denoting the observed event type, and a continuous component $z^c \in \mathcal{X}$ representing the location of the event. The events of different types arrive with equal frequencies and according to their spatial density functions. We assume that the relative importance of the event types, denoted by $\Phi_j = \int_{\mathcal{X}} \phi_i(q)dq$ is known. By extending the results in [66], the control law become

$$q_{i,t+1} = \begin{cases} q_{i,t} - \gamma_t \Phi_j \frac{df^j}{dx^2}\Big|_{||z_t^c - q_{i,t}||} (z_t^c - q_{i,t}) & \text{if } z_t^d \in S_i, z_t^c \in \mathtt{Vor}_i^{z_t^d} \\ q_{i,t} & \text{otherwise.} \end{cases} \tag{4.5}$$

Observe that the expected value of the direction $\frac{df^j}{dx^2}|_{x=||z_t^c - q_{i,t}||}(z_t^c - q_{i,t})$ coincides with the direction of the derivative of $\mathcal{H}$ with respect to q_i in Equation (4.3). The control law (4.5) drives the closest robot with the required equipment towards the observed event. The convergence of the control law to locally optimal positions arrive from the proof of Theorem 3 in [66]. •

4.4 Gradient Descent Algorithm for Discrete Environments

In this section, we discuss the coverage problem for heterogeneous robots in a discrete environment with multiple event types (Problem 4.2.2 in Section 4.2). Given a graph, the goal is to position the robots on the vertices of the graph such that the total sensing function $\mathcal{H}$ is minimized.

We define two configurations of the robots on the graph. The first configuration $Q \subset \mathcal{V}$ is a subset of size m of the vertices of the graph which represents positioning the robots on the vertices. The second configuration $D \subset E \times [0,1]$ represents the placement of the robots on the edges. For a placement of robot i on edge (u,v), i.e., $q_i = ((u,v),\gamma)$ (see Figure 4.3), parameter γ is the fraction of the path from q_i to v along the edge (u,v). The distance of robot i from vertex $w \in \mathcal{V}$ is $d'(q_i,w) = \min\{\gamma c(u,v) + d(u,w), (1-\gamma)c(u,v) + d(v,w)\}$.

For a configuration D, each event is assigned to the closest robot with required equipment. Then total sensing function for configuration D and partition $W(D)$ is

$$\mathcal{H}'(D, W(D)) = \sum_{i=1}^{m} \sum_{j \in S_i} \sum_{v \in W_i^j} f^j(d'(q_i, v))\phi_v^j. \tag{4.6}$$

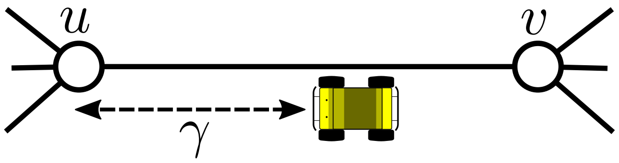


Figure 4.3: Robot located on an edge of a graph.

We provide a gradient descent control law to position the robots in the graph minimizing Equation (4.2). To motivate the approach, first, we provide the following result on the optimal solution of the discrete problem.

Lemma 4.4.1. *For any placement of the robots in the graph D, there exists another placement of the robots on the vertices Q such that*

$$\mathcal{H}'(D, W(D)) \leq \mathcal{H}(Q, W(Q)).$$

Proof. For any placement of the robots in the graph, if there exists a robot located on the edge of the graph, we create another placement without decreasing the total sensing cost. Figure 4.3 illustrates an instance in which a robot i is located on the edge (u, v). For each event type $j \in S_i$, we partition the vertices in W_i^j into two subset where the first subset $W_{i,u}^j$ consists of the vertices in W_i^j such that the shortest path from q_i contains u and similarly the second subset $W_{i,v}^j$ consists of the vertices in W_i^j such that the shortest path for the robot to reach the vertex passes through v. The total event mass on the first subset is $\sum_{j \in S_i} \sum_{w \in W_{i,u}^j} \phi_w^j$ and similarly for the second subset is $\sum_{j \in S_i} \sum_{w \in W_{i,v}^j} \phi_w^j$. Then we move the robot to the vertex with a larger total event mass. Observe that moving the robot to the vertex with a larger total event mass can only increase the total sensing cost. $\square$

This result motivates us to consider the cases where the robots are located on the vertices of the graph. The set of admissible controls for robot i, $\mathcal{U}_i$, is limited to $\cup_{j \in S_i} W_i^j(\mathcal{P}_t)$ to ensure that the robots are required to communicate with only their neighboring robots. The robots i, k are called neighbors if there exists j such that the sets W_i^j, W_k^j share in edge on graph G. Observe that the control set $\mathcal{U}_i$ is a non-empty set since it contains the current location of the robot. We say the robots are in a *locally optimal configuration* if there is no robot can take a control input in its control set to improve the sensing cost unilaterally.

Now we provide the control law on the graph as follows:

$$p_{i,t+1} = \arg\max_{u \in \mathcal{U}_i} \sum_{j \in S_i} \sum_{w \in W_i^j(\mathcal{P}_t)} f^j(d(u,w))\phi_w^j. \tag{4.7}$$

Control law (4.7) drives the robots to a configuration on the graph minimizing the total sensing cost of the events in the partitions dominated by the robot. This resembles the control law in Equation (4.4). The following lemma provides the results on the convergence of the control law (4.7).

Lemma 4.4.2. *Under the control law (4.7), the robots converge to a set of locally optimal locations for the total sensing problem in discrete environments.*

Proof. By the definition of the control law in Equation (4.7) we move robot i to the new position such that it minimizes the sensing cost for the vertices in the partitions of robot i, i.e., $W_i^j(\mathcal{P}_t), j \in S_i$. Considering a fixed partition, by any robot relocating on the graph, the total sensing cost improves, i.e.,

$$\mathcal{H}(\mathcal{P}_t, W(\mathcal{P}_t)) \leq \mathcal{H}(\mathcal{P}_{t+1}, W(\mathcal{P}_t)).$$

Observe that the sensing cost is minimized when each event is assigned to the closest robot. Then by updating the partition for the new positions of the robots we have

$$\mathcal{H}(\mathcal{P}_{t+1}, W(\mathcal{P}_t))) \leq \mathcal{H}(\mathcal{P}_{t+1}, W(\mathcal{P}_{t+1})).$$

Therefore, the cost improves with each step of the control law and the algorithm converges to a locally optimal solution of the discrete total sensing cost problem. □

4.5 Experimental Results

In this section, we evaluate the performance of the control laws for both continuous and discrete environments.

4.5.1 Continuous Environment

The performance of the proposed control law for continuous environments is evaluated with four experiments. Each experiment consists of 8 GRITSBots [69] with different sensing

	θ_j	`SensorSet`
Experiment 1	$\theta_j = 1\ j \in [k]$	$S_1 = S_7 = \{3,4\}, S_2 = \{2,4\},\ S_3 = \{1,2,3\},$ $S_4 = \{1,3\}, S_8 = \{4\}, S_5 = \{1,2,3,4\}, S_6 = \{1,2\}$
Experiment 2	$\theta_j = 1\ j \in [k]$	$S_i = \{1,2\}\ i \leq 4,\ \ S_i = \{3,4\}\ i \geq 5$
Experiment 3	$\theta_j = 1\ j \in [k]$	$S_i = \{j \in [k] \| j <= i\}$
Experiment 4	$\theta_j = j\ j \in [k]$	$S_i = [k]$

Table 4.1: Parameters of the experiments

capabilities. The proposed control law is implemented on the Robotarium [70] using both simulation and physical experiments. The density function for each event type j is given by a bivariate normal distribution, i.e.,

$$\phi_j(q) = \frac{\theta_j}{2\pi\sqrt{|\Sigma|}} \exp\big(-\frac{1}{2}(q-\tau_j)^T \Sigma^{-1}(q-\tau_j)\big),$$

where τ_j is the mean and $\Sigma = [0.1, 0; 0, 0.1]$ is the covariance matrix. The parameter θ_j is a scaling factor to represent the importance of each event type. The sensing function is for all event types $f^j(x) = x^2, j \in [k]$. Table 4.1 shows the parameters of each experiment.

In Figure 4.4, the sensing cost of the proposed control law is compared to that of the Heterogeneous Lloyd's algorithm proposed in [44]. The results are the average of 500 instances of Experiment 3 with different uniformly random initial locations for the robots. The mean values $\tau_j, j \in [k]$ of density functions for each instance are generated randomly in a 1×1 environment. The dashed lines represent the total sensing cost in Equation (4.1) for both algorithms. Figure 4.4 show the significant improvement of the sensing cost for each event type with respect to the Heterogeneous Lloyd's algorithm. The percentage improvement is the ratio of the difference between the sensing cost of the Heterogeneous Lloyd's algorithm and the sensing cost of the proposed algorithm to the sensing cost of the proposed algorithm. Figure 4.5 shows the total sensing cost for the proposed algorithm and the Heterogeneous Lloyd's algorithm. The comparison of the two algorithms on different experiments are given in Table 4.2. The proposed control law outperforms the existing algorithm in total sensing cost and in the sensing cost of each event type. Figure 4.6 show the paths of robots covering four events, starting from a random initial positions.

Figure 4.7 illustrates the final configuration of the robots in the physical implementation of the proposed algorithm in the Robotarium with 8 robots for an instance of Experiment 1. In the experiment, the proposed algorithm improved the total sensing cost by 87.7% in 1 minute.

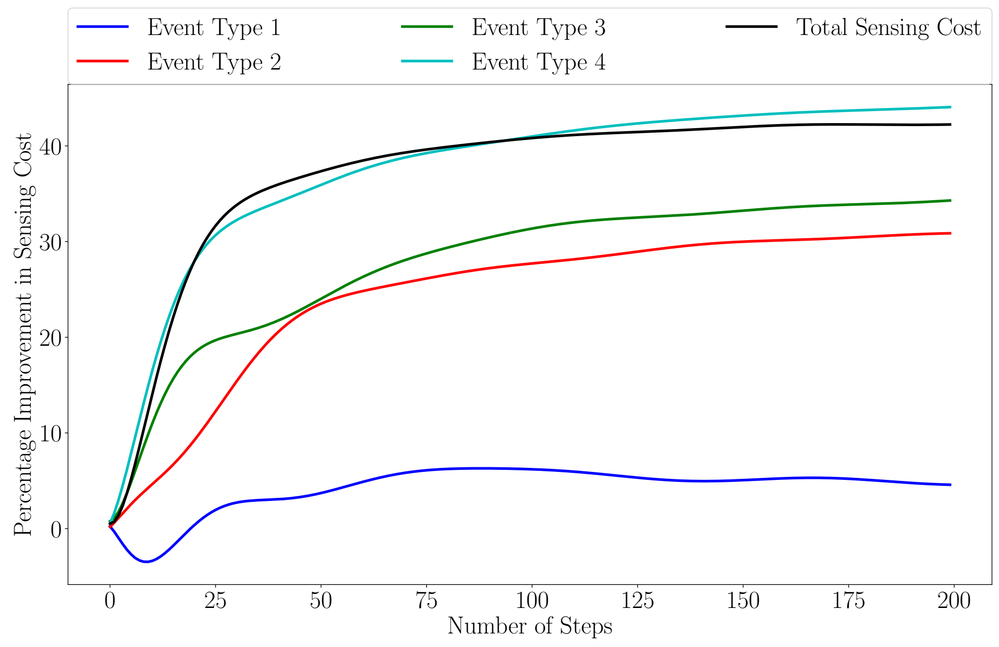


Figure 4.4: Performance of the proposed control law for Experiment 3

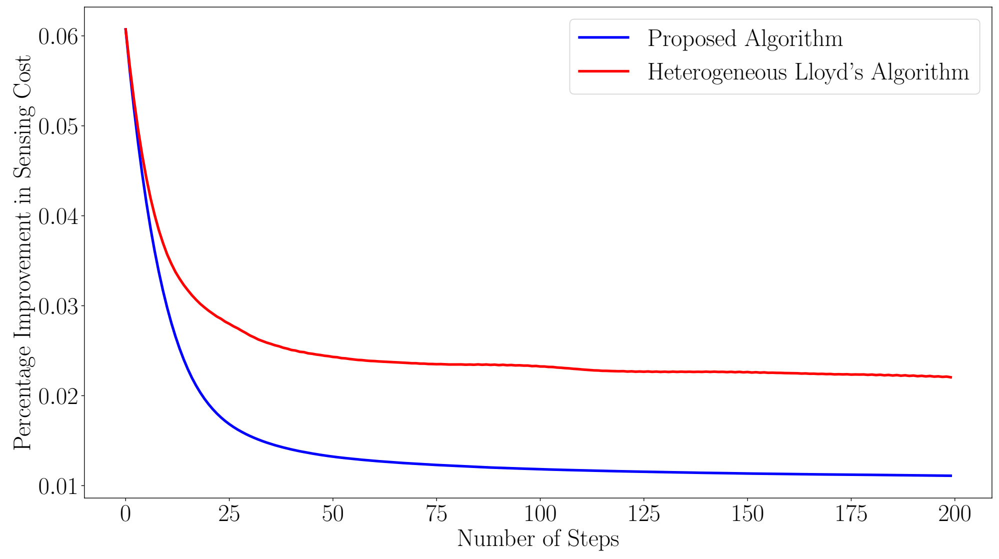
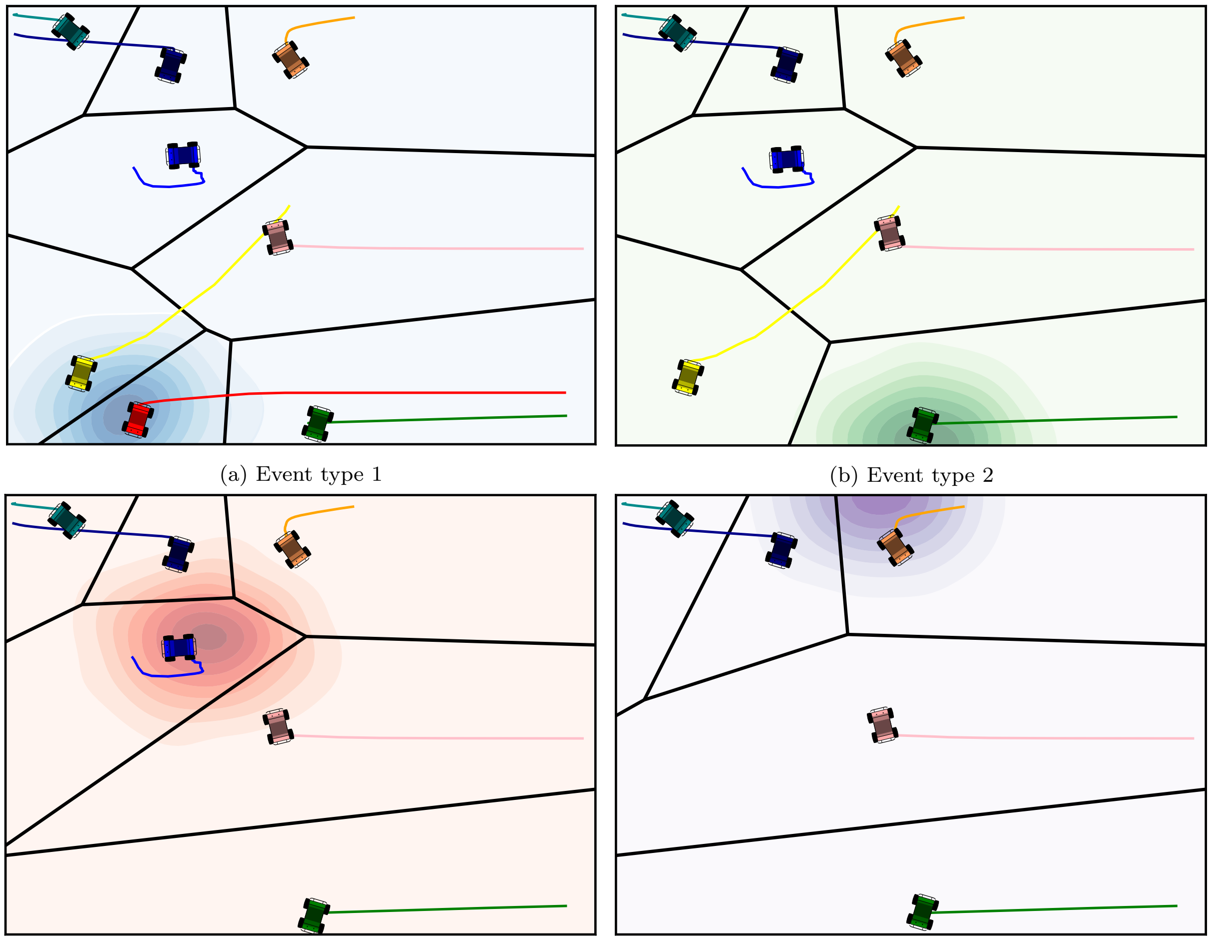


Figure 4.5: The comparison of the total sensing cost for the proposed algorithm and the Heterogeneous Lloyd's algorithm with environment settings of Experiment 3

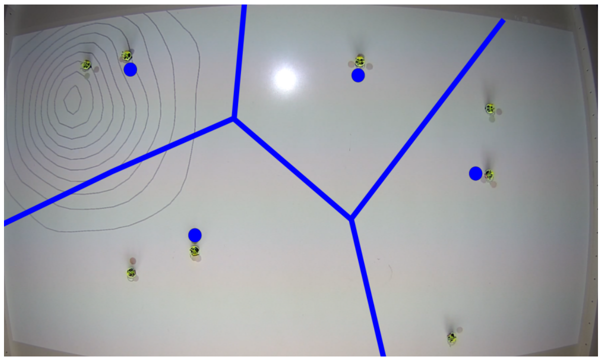

(a) Event type 1 (b) Event type 2

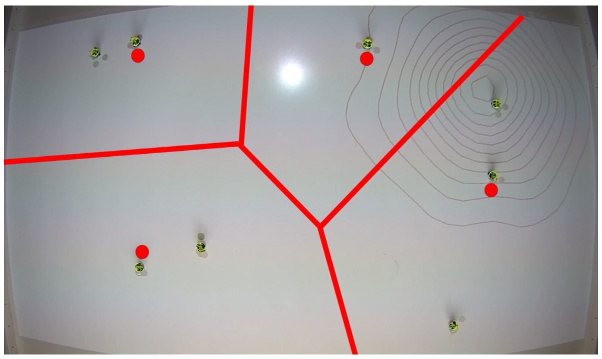
(c) Event type 3 (d) Event type 4

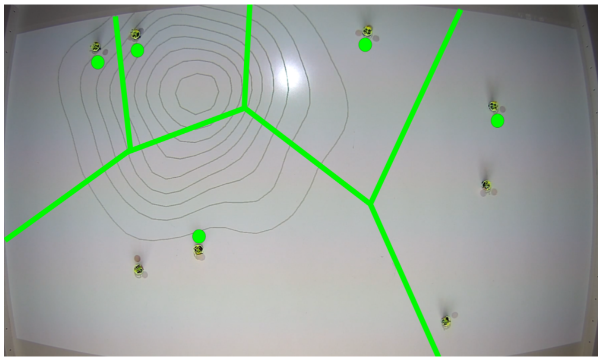
Figure 4.6: Paths of eight robots covering four event types with the proposed algorithm.

	Event 1		Event 2		Event 3		Event 4		Total Sensing	
	Avg.	σ	Avg.	σ	Avg.	σ	Avg.	σ	Avg.	σ
Experiment 1	57.3	7.1	56.3	8.6	56.3	7.9	42.2	6.5	55.3	5.0
Experiment 2	57.8	4.3	57.1	3.9	57.8	4.3	57.9	4.4	58.1	2.1
Experiment 3	7.2	5.0	32.7	4.5	36.6	4.2	46.1	3.2	43.5	1.8
Experiment 4	34.8	5.5	46.0	4.4	49.8	5.0	52.6	7.3	45.4	3.2

Table 4.2: Average percentage improvement of the sensing cost for the proposed algorithm compared to Heterogeneous Lloyd's algorithm over 500 instances for each experiment. The mean deviation from average is denoted by σ.


(a) Event type 1


(b) Event type 2


(c) Event type 3

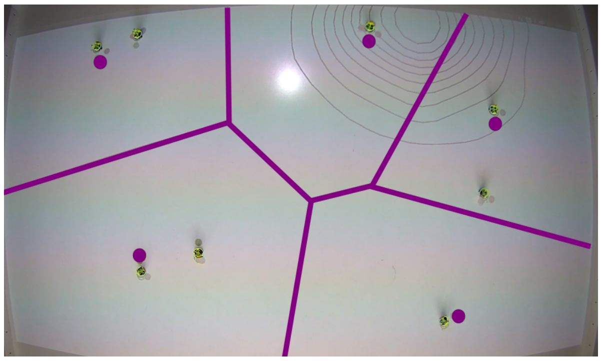

(d) Event type 4

Figure 4.7: Eight robots with different sensing capabilities in the Robotarium with the environment size of 2×3.2 meters. The density functions of each event type are shown via contours. The colored circles next to the robots show the generators of each Voronoi partition.

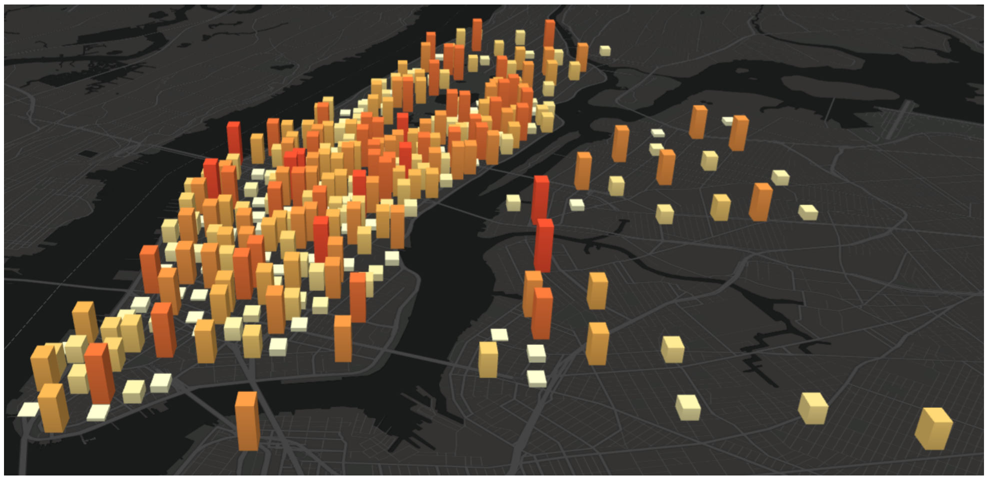
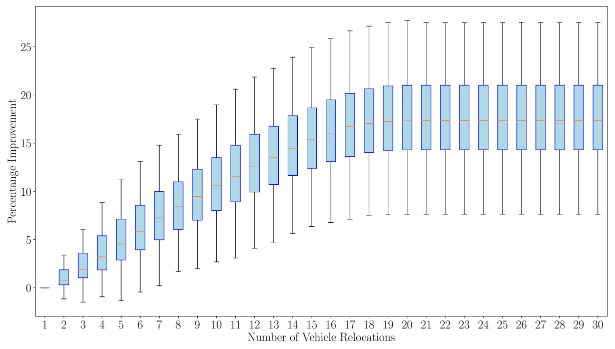


Figure 4.8: Pick-up locations in Manhattan N.Y. and a sample distribution represented with a bar chart

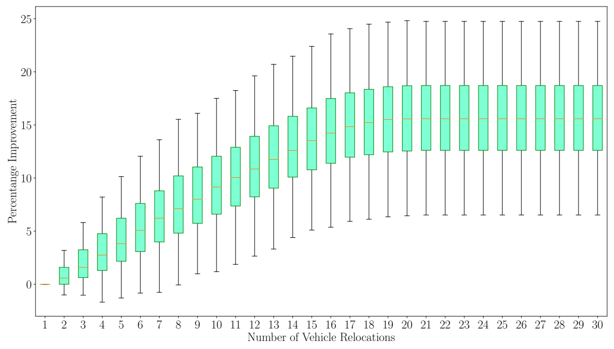
4.5.2 Discrete Environment

In the discrete environment, we evaluate the performance of the proposed control law in a deployment problem motivated by transportation applications. Figure 4.8 shows 500 randomly generated pick-up locations in Manhattan. Requests for rides arrive at the pick-up locations. Each request has a capacity requirement in $\{2, 3, 4, 5\}$, giving the size of the group seeking a ride. There are 5 vehicles of each capacity $\{2, 3, 4, 5\}$ requirement. There are then four event types, one for events with each capacity requirement. Observe that a vehicle with capacity i can serve any request with a group size of $k \leq i$. The objective is to position the vehicles in the environment to minimize the time to respond to a request. Note, here we are considering only the initial deployment problem, which is to place the vehicles at locations to best respond to an initial request.

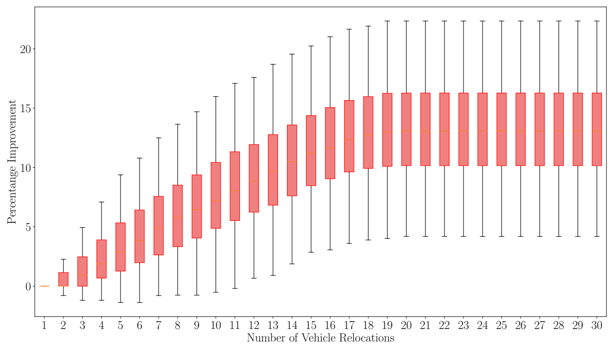
Figure 4.9 shows the improvement in the total service quality for different event types and the total service quality. The results are the average of 1000 randomly generated mass functions. The lines show the mean value and the shaded area represent the first and third quartiles of each data set. Notice that the event type with capacity 2 is serviced by all the vehicles, and the algorithm shows the best improvement for this event type.



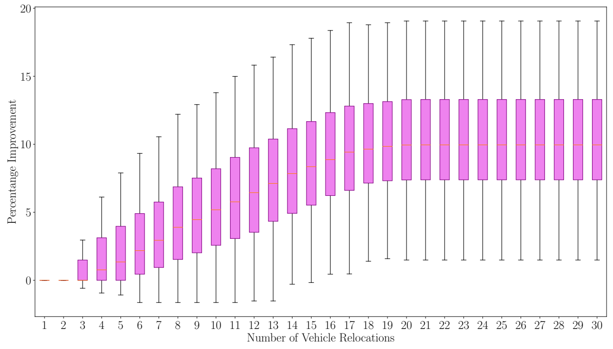
(a) Improvement in expected response time of ride-requests of size 2



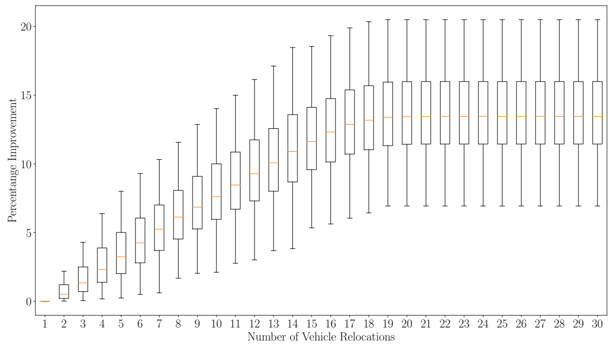
(b) Improvement in expected response time of ride-requests of size 3



(c) Improvement in expected response time of ride-requests of size 4



(d) Improvement in expected response time of ride-requests of size 5



(e) Improvement in the total expected response time of ride-requests

Figure 4.9: Improvement in the expected response time of the ride-requests with different sizes

Chapter 5

Re-Deployment of Multiple Service Robots

In service applications, robots are tasked with responding to requests for service that arrive sequentially over time [16]. For example, in a hospital setting [6] a fleet of robots may be used to assist patients by traveling to their locations. A key aspect in such applications is where the robots should wait (i.e., their deployment locations) to optimally respond to the next service request. After a request has been serviced, the robots can redeploy to re-optimize their positions for future requests. There is an inherent trade-off between the expected response time for a service request and the cost incurred to redeploying robots between successive requests.

In this chapter, we focus on the problem of deploying a set of robots to service tasks arriving sequentially in an environment. The robots' motion in the environment is captured as a road-map (i.e., graph), and each task arrives at a node of the graph according to a known probability distribution. A group of k robots move on a common road-map, and tasks arrive on the vertices of the road-map. A task is serviced by a robot traveling to the task location. At each task arrival, we consider minimizing the response time and the redeployment cost. The robots choose to redeploy to another location between task arrivals. We consider this objective on a horizon of the next N task arrivals.

The contributions of this chapter are as follows. Our first contribution is to formulate the re-deployment problem as a multi-stage optimization. We then cast the problem as a dynamic program (DP), and show that in the single robot case, a simple greedy policy is optimal. Then we propose two simple policies for the multiple robot problem with theoretical performance guarantees. In the first policy, built on the existent α-approximation algorithm [31] considering only one arrival, we provide a 5α-approximation algorithm for the two-stage problem. In the second policy, the planning horizon is extended to the infinity. We show that if robots are initiated in the k-median, then the proposed policy is a 2α-approximation in a setting prioritizing the expected service cost. Finally, we evaluate performance of both random environments and a workplace floor plan.

This chapter is organized as follows. In Section 5.2, we formulate the problem as a dynamic program. In Section 5.3, we provide a policy for the single robot problem. Section 5.4 consists of upper and lower bounds for the optimal value and two policies for the multi-robot problem. Finally in Section 5.5, we provide benchmarking results.

5.1 Related Work

The task allocation problem has been the subject of extensive research [66, 71, 72]. Some of the most related work is [66, 71], where tasks are assigned dynamically to the robots in free space, and the objective is to deploy the robots such that the response time is minimized. In contrast, we focus on dynamically assigning tasks for robots moving on a roadmap, and look to minimize both response time and redeployment cost.

The facility location problem [55, 28] is the problem of installing facilities in a set of locations with a fixed cost of opening a facility. Demands arrive at the different locations and the objective is to minimize the time to respond to the demand and the total cost of opening facilities. A special case of the facility location problem, is the k-median problem [28] where the cost of opening facilities is zero.

An extension to the facility location problem is the mobile facility location problem (MFL), introduced in [29]. The objective is to move the facilities while minimizing the total movement cost and the response time. The two main differences between our problem and the MFL are 1) robots service tasks by visiting the task location, thus the configuration changes with each arrival and 2) MFL considers just the next arrival and plans the next waiting configuration for a single-ahead stage. In contrast, we plan the next configuration of the robots for a horizon of N task arrivals.

Online algorithms are presented in the literature for the facility location problem [73, 74]. The results are also extended to MFL with stochastic demands [75, 76]. A related problem is the k-server [77] problem in which tasks arrive sequentially over time, and each task is serviced by visiting the corresponding task location. However, the problem is defined on a metric space, rather than a road-map, and the objective is to minimize the server travel over the worst-case set of task arrivals. In contrast, we consider a known spatial distribution for service requests, capturing scenarios where prior information is available on the frequency of service requests.

Another closely related area of research is dynamic vehicle routing (DVR) [14]. In DVR tasks arrive sequentially over time according to a stochastic process. The most closely related results are on light-load policies, where the arrival rate of tasks is low. However, these problems consider only the service quality as a metric, and look at arrivals in the Euclidean plane rather than roadmaps.

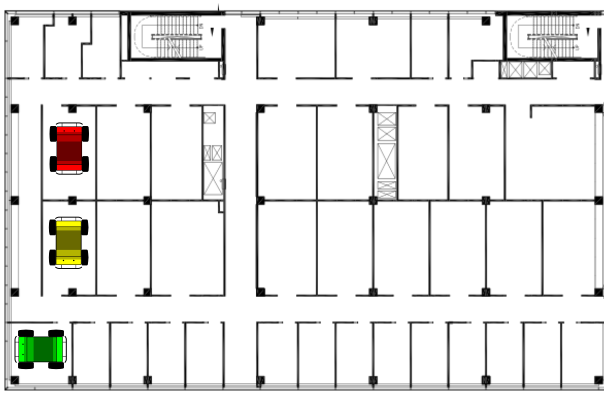
5.2 Problem Formulation

Consider a set of m robots and a set of locations $\mathcal{V}$ for the robots to wait and tasks to arrive. Let $G = (\mathcal{V}, \mathcal{E}, c)$ be a metric graph on vertices $\mathcal{V}$, let $\mathcal{E}$ be the edge set and $c : \mathcal{E} \to \mathbb{R}_+$ be the cost function defined on the edge set satisfying the triangle inequality. A vertex of the graph represents a state of the robots, e.g. position and heading for the Dubins car, and cost of an edge represents the time robot takes to travel between the states.

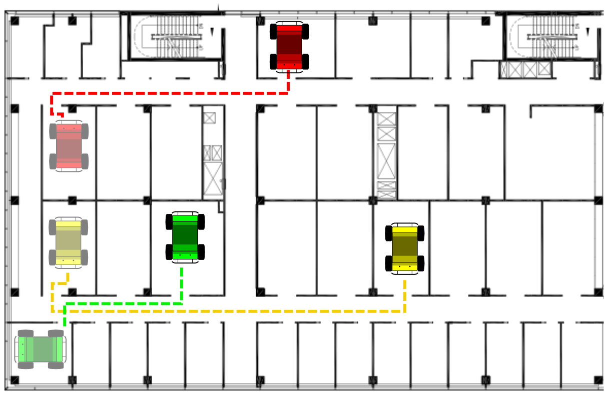
A subset of $\mathcal{V}$ occupied by the robots at a time stage is called a configuration and it is denoted by $Q \in \mathcal{Q}$ where $\mathcal{Q}$ is the set of all the possible configurations, i.e., all subsets of $\mathcal{V}$ with size k.

The probability that a task arrives on vertex $v \in \mathcal{V}$ is $p_a(v) \geq 0$, where $\sum_{v \in \mathcal{V}} p_a(v) = 1$. Task arrivals occur sequentially, with the time between arrivals sufficient for the robots to reconfigure between arrivals. We explore the effect of relaxing this assumption in Section 5.5. This is analogous to light load in DVR [14]. The task locations are independent and identically distributed. A robot is assigned to the task after the task arrival and services by visiting its location.

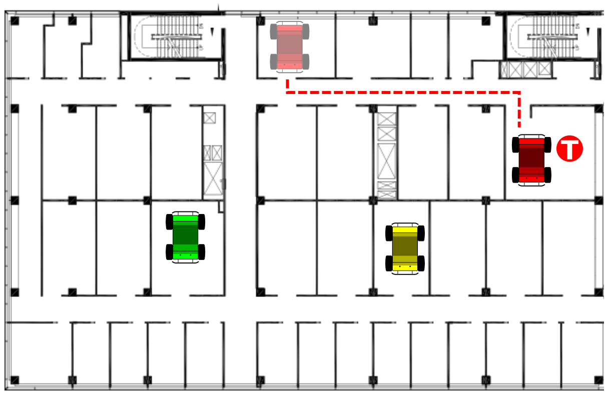
We consider a multi-objective problem of minimizing a linear combination of the response time and the redeployment cost, where the latter part captures energy consumption, or additional travel. The redeployment cost between two configurations is denoted by $\text{Assgn}(Q_1, Q_2)$. The minimum cost redeployment between two configurations is the minimum cost assignment of the vertices of two sets Q_1 and Q_2. Let $d(Q, u)$ be the closest distance from the vertices of Q to u, then we define the response time of a configuration as the expected distance to the next arrival, i.e., $\sum_{u \in \mathcal{V}} d(Q, u)$. Figure 5.1 illustrates the two stages of the problem. Robots after servicing task $i - 1$ at v are at configuration $Q_{i-1,v}$ (Figure 5.1a), then they redeploy to configuration Q_i waiting for the next task to arrive (Figure 5.1b). Task i arrives at u and the closest robot moves to service the task (Figure 5.1c), and finally the robots redeploy to Q_{i+1} (Figure 5.1d).



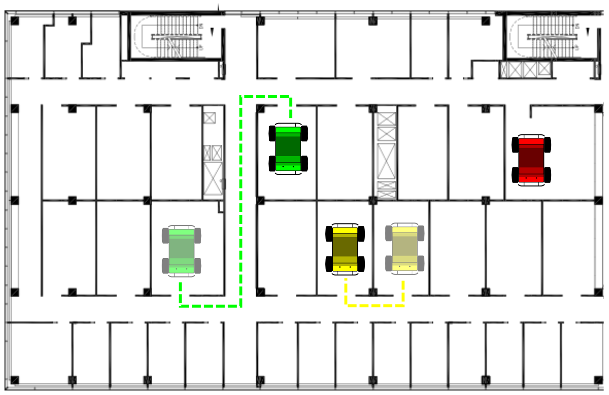
(a) Configuration $Q_{i-1,v}$, serviced a task and transitioning to the new waiting configuration.



(b) Configuration Q_i, waiting for task arrival.



(c) Task t_i arrives at u, transition to configuration $Q_{i,u}$.



(d) New waiting configuration Q_{i+1}

Figure 5.1: Demonstrating a single stage of the problem. (a) Configuration of the robots after servicing task t_{i-1} at v (b) New waiting location of the robots for the new task arrival (c) Task arrival and the closest robot deployed to service the task (d) New waiting location of the robots for the next arrival.

The cost incurred at each stage i is linear combination of the service time and the redeployment cost, i.e., $\text{Assgn}(Q_{i-1}, Q_i) + \beta \sum_{u\in\mathcal{V}} p_a(u)d(Q_{i,u})$, where β is a user-defined variable.

Observe that the optimal configuration at each stage depends on the previous arrivals. To account for the future arrivals at each stage, we formualte the problem as a problem of minimizing the discounted stage costs over N next task arrivals. Let $V_i(Q)$ be the expected cost incurred during stages $i, \dots, N$ at current configuration Q, then we can write the deployment problem as follows:

$$V_i(Q_{i-1}) = \text{Assgn}(Q_{i-1}, Q_i) + \beta \sum_{u\in\mathcal{V}} p_a(u)d(Q_i, u) + \gamma \sum_{u\in\mathcal{V}} p_a(u)V_{i+1}(Q_{i,u}). \tag{5.1}$$

Note that if N approaches infinity, then the problem becomes a discounted factor infinite horizon problem. For small values of β and γ, the problem corresponds to the case where minimizing the transition cost between the configurations is dominant. For instance, with $\beta = 0$ the optimal policy at each stage is $Q_i = Q_{i-1}$. In the case of large β and $\gamma = 0$, the optimal policy at stage i approaches the k-median solution [28], i.e., $\min_{Q_i\in\mathcal{Q}} \sum_{u\in\mathcal{V}} p_a(u)d(Q_i, u)$. In [30], the authors show that when the number of robots k is an input then even the single-stage problem is NP-hard, which in turn implies that Problem (5.1) is NP-hard.

In the next section, we provide a policy for the single robot case, and in Section 5.4 we discuss multiple robots problem.

5.3 Optimal Policy for Single Robot

In this section, we provide the optimal policy for the single robot case. The greedy approach to the problem for a single robot is to plan for a single stage without considering future arrivals. Let $\Pi_1(Q_{i-1})$ be the greedy policy at configuration Q_{i-1}, which returns the minimizer of the single stage, i.e.,

$$\Pi(Q_{i-1}) = \text{argmin}_{Q_i\in\mathcal{Q}} \text{Assgn}(Q_{i-1}, Q_i) + \beta \sum_{u\in\mathcal{V}} p_a(u)d(Q_i, u). \tag{5.2}$$

In Lemma 5.3.1, we show that the greedy policy is optimal. Consider a planning horizon of N stages and let $\langle Q_1^*, \dots, Q_N^*\rangle$ be the sequence of optimal configurations of the robot. Let $\langle u_1, \dots, u_N\rangle$ be the sequence of tasks, then we establish the following result.

Lemma 5.3.1. *Given a single robot, the greedy policy in* (5.2) *is optimal.*

Proof. Proof by induction. At the final stage N, the problem becomes the single stage problem, therefore, the greedy policy is optimal at stage N. By induction, assume that the greedy policy returns the optimal for $V_{i+1}(Q)$ for all $Q \in \mathcal{Q}$. For $k = 1$, the configuration $Q_{i,v}$ is independent of the previous choice of the configuration Q_i and it is only dependent on the vertex of task t_i. Since tasks are independent and identically distributed, then regardless of the choice of Q_i at stage i, the expected cost of future arrivals is only dependent on the probability distribution. By the induction hypothesis, the greedy algorithm returns the optimal value of $V_{i+1}(Q_{i.v})$ for all $Q_{i,v}$. Therefore, the optimal policy for Problem (5.1) with single robot is the minimizer of

$$\text{argmin}_{Q_i \in \mathcal{Q}} \text{Assgn}(Q_{i-1}, Q_i) + \beta \sum_{u \in \mathcal{V}} p_a(u) d(Q_i, u).$$

This is the greedy policy in (5.2). $\square$

Complexity of greedy policy: The greedy policy considers all the vertices in $\mathcal{V}$. The $\text{Assgn}(Q_{i-1}, Q_i)$ function is $O(1)$ and $\sum_{u \in \mathcal{V}} p_a(u) d(Q_i, u)$ is $O(|\mathcal{V}|)$. Therefore, the total run-time of the greedy policy at each stage is $O(|\mathcal{V}|^2)$.

5.4 Multiple Robots

In this section, we investigate the optimal policy for the multiple robots. In Section 5.3, the optimal policy at each arrival is independent of the previous choices of the configurations, and thus the greedy policy is optimal. However, this is not the case for the multiple robots.

Example 5.4.1. Consider an instance of the problem with two robots on a graph with vertices located at $\{(0,0), (0,2), (2,0), (2,2), (1,1)\}$ in the plane (see Figure 5.2a). Let $\beta = 5$, $\gamma = 0.9$ and equal probability on the vertices, then for any configuration that does not contain the mid-vertex, the single stage optimal policy is to stay at the configuration, however, the optimal policy for infinite horizon is to move one of the robots to the mid-point (see Figure 5.2b).

The infinite horizon problem, namely Problem (5.1) is a stationary problem, i.e. the optimal policy at each configuration is equal for every stage i. Thus, the problem can be written as a dynamic program as follows:

$$V(Q) = \min_{Q' \in \mathcal{Q}} \{\text{Assgn}(Q, Q') + \beta \sum_{u \in \mathcal{V}} p_a(u) d(Q', u) + \gamma \sum_{u \in \mathcal{V}} p_a(u) V(Q'_u)\} \quad \forall Q \in \mathcal{Q}. \quad (5.3)$$

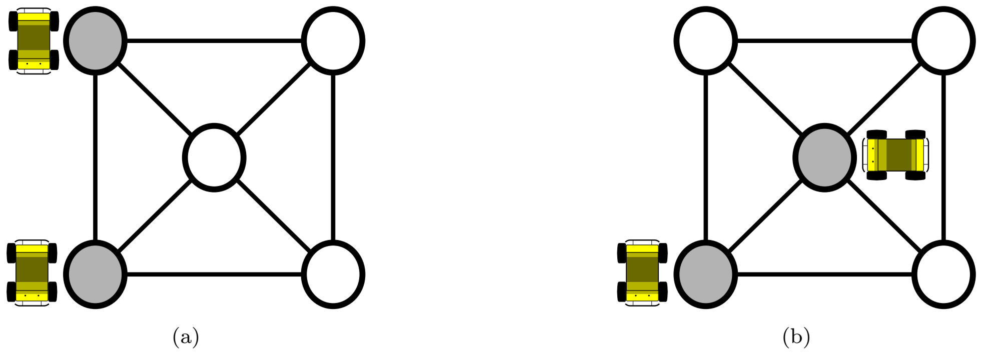


Figure 5.2: Example of different solutions of single-stage optimal and infinite horizon optimal solutions. The dark vertices are occupied with the robots. (a) shows the initial configuration and the optimal solution of single-stage. (b) shows the optimal policy for the given initial configuration in infinite horizon.

The dynamic program (DP) is solved via several well-known approaches such as policy iteration, value iteration, or linear programming. Observe that for an instance of k robots and n tasks locations, there exist $\binom{n}{k}$ configurations – exponential in the number of robots. The main drawback of the dynamic programming approaches is that finding the optimal policy for a single configuration requires solving DP (5.3) for all the configurations, which is not computationally tractable for large instances.

In this section, we propose two simple policies and we provide some results on the performance of them. First we provide bounds on the optimal value of V, which is essential in the performance analysis of the two policies.

5.4.1 Bounding the Value Function

In this section, we provide bounds on the optimal value function V of configurations for the infinite horizon problem. These bounds are essential in our analysis of the simple policies. For simplicity we define the following notations:

$$\mathcal{D}(Q) := \sum_{u \in \mathcal{V}} p_a(u) d(Q, u); \text{ and } \mathrm{StageCost}(Q_1, Q_2) := \mathrm{Assgn}(Q_1, Q_2) + \beta \mathcal{D}(Q_2).$$

The term $\mathcal{D}(Q)$ is the expected response time of the robots at configuration Q, and $\mathrm{StageCost}(Q_1, Q_2)$ is the stage cost for redeployment from Q_1 to Q_2.

The following results provides an upper-bound on the optimal value for a given configuration.

Lemma 5.4.2 (Upper-bound)**.** *The cost of the optimal policy for $Q \in \mathcal{Q}$ is upper-bounded by*

$$V(Q) \leq \min_{Q' \in \mathcal{Q}} \{\text{StageCost}(Q, Q') + \gamma \frac{\beta + 1}{1 - \gamma} \mathcal{D}(Q')\}.$$

Proof. For any $Q' \in \mathcal{Q}$ we have,

$$V(Q) \leq \text{StageCost}(Q, Q') + \gamma \sum_{v \in \mathcal{V}} p_a(v) V(Q'_v). \tag{5.4}$$

Therefore, we need to upper-bound $V(Q'_v)$ for all $v \in \mathcal{V}$. Note that $V(Q'_v)$ is the cost of the optimal policy for the configuration Q'_v. Then we have,

$$V(Q'_v) \leq \text{StageCost}(Q'_v, Q') + \gamma \sum_{u \in \mathcal{V}} p_a(u) V(Q'_u).$$

Note that $\text{Assgn}(Q'_v, Q) = d(Q, v)$, thus we have,

$$V(Q'_v) \leq d(Q', v) + \beta \mathcal{D}(Q') + \gamma \sum_{u \in \mathcal{V}} p_a(u) V(Q'_u).$$

Therefore, we have,

$$\begin{aligned}
\sum_{v \in \mathcal{V}} p_a(v) V(Q'_v) &\leq \sum_{v \in \mathcal{V}} p_a(v) \Big(d(Q', v) + \beta \mathcal{D}(Q') + \gamma \sum_{u \in \mathcal{V}} p_a(u) V(Q'_u) \Big) \\
&\leq \sum_{v \in \mathcal{V}} p_a(v) d(Q', v) + \beta \sum_{v \in \mathcal{V}} p_a(v) \mathcal{D}(Q') + \gamma \sum_{v \in \mathcal{V}} p_a(v) \sum_{u \in \mathcal{V}} p_a(u) V(Q'_u) \\
&\leq (\beta + 1) \sum_{v \in \mathcal{V}} \mathcal{D}(Q') + \gamma \sum_{v \in \mathcal{V}} p_a(u) V(Q'_u).
\end{aligned}$$

Finally we have,

$$\sum_{v \in \mathcal{V}} p_a(v) V(Q'_v) \leq \frac{\beta + 1}{1 - \gamma} \mathcal{D}(Q') \tag{5.5}$$

The result follows directly from Equations (5.4) and (5.5). □

Let $\Pi_1^* : \mathcal{Q} \to \mathcal{Q}$ be the optimal policy for the single stage problem, i.e.,

$$\Pi_1^*(Q) = \arg\min_{H \in \mathcal{Q}} \operatorname{Assgn}(Q, H) + \beta \mathcal{D}(H) \tag{5.6}$$

and let M be the minimizer of the k-median problem, i.e, $M = \arg\min_{Q \in \mathcal{Q}} \mathcal{D}(Q)$. Let Q^* be the configuration associated with the optimal policy at Q, i.e., $Q^* = \Pi^*(Q)$, then we establish following lower-bound on the optimal value.

Lemma 5.4.3 (Lower-bound)**.** *The cost of the optimal policy for $Q \in \mathcal{Q}$ is lower-bounded by*

$$V(Q) \geq \operatorname{StageCost}(Q, \Pi_1^*(Q)) + \gamma \frac{\beta}{1-\gamma} \mathcal{D}(M).$$

Proof. Since moving to Q^* is the optimal action in Q, then we have,

$$V(Q) = \operatorname{StageCost}(Q, Q^*) + \gamma \sum_{v \in \mathcal{V}} p_a(v) V(Q_v^*). \tag{5.7}$$

By the definition of Π_1^*, we have, $\operatorname{StageCost}(Q, \Pi_1^*(Q)) \leq \operatorname{StageCost}(Q, Q^*)$. Then to prove the result, it suffices to show that

$$\sum_{v \in \mathcal{V}} p_a(v) V(Q_v^*) \geq \frac{\beta}{1-\gamma} \mathcal{D}(M).$$

Let $\Pi^*(Q)$ be the function takes a configuration as input and returns the optimal configuration to visit. Then $\Pi^*(Q)_u$ is configuration where the closest robot at $\Pi^*(Q)$ to u has moved to u. Thus we have,

$$V(Q_v^*) \geq \beta \mathcal{D}(M) + \gamma \sum_{u \in \mathcal{V}} p_a(u) V(\Pi^*(Q_v^*)_u).$$

For the simplicity of the notation, let $M = \Pi^*(Q_v^*)$, then we can write similar inequalities for each M_u as follows:

$$V(M_u) \geq \beta \mathcal{D}(M) + \gamma \sum_{w \in \mathcal{V}} p_a(w) V(\Pi^*(M_u)_w).$$

Therefore, we have,

$$V(Q_v^*) \geq \beta \mathcal{D}(M) + \gamma \beta \mathcal{D}(M) + \gamma^2 \sum_{u \in \mathcal{V}} p_a(u) \sum_{w \in \mathcal{V}} p_a(w) V(\Pi^*(M_u)_w).$$

Observe that we can repeatedly write the similar inequalities for each stage, then we have,

$$V(Q_v^*) \geq \beta \sum_{u \in \mathcal{V}} p_a(u) d(M, u)(1 + \gamma + \gamma^2 + \dots) = \frac{\beta}{1-\gamma} \mathcal{D}(M). \tag{5.8}$$

Finally, the result is immediate by Equations (5.7) and (5.8). □

In the rest of this section, we provide our two simple policies for deployment of a system of multiple robots.

5.4.2 Policy I: Approximation Algorithm for Two-stage-Horizon

First, from the problem definition in Section 5.2, given a configuration Q_{i-1}. the two-stage problem is to minimize:

$$\min_{Q_i \in \mathcal{Q}} \text{StageCost}(Q_{i-1}, Q_i) + \gamma \sum_{u \in \mathcal{V}} p_a(u) \text{StageCost}(Q_{i,u}, \Pi_1(Q_{i,u})) \tag{5.9}$$

where Π_1 is a policy defined on $\Pi_1 : \mathcal{Q} \to \mathcal{Q}$. With a simple observation, we can show that Π_1 is Π_1^*. Our algorithm for the two-stage problem is built on the existent algorithms for the single-stage problem with multiple robots. We provide the algorithm for Two-stage-Horizon in Algorithm 5.1, and we prove approximation results in Theorem 5.4.4.

Built on the approximation algorithms for the single-stage problem in [31], our algorithm for the two-stage problem consists of three single stage-problems. The algorithm evaluates the solutions to the single-stage problems in Line 7 and returns the better solution. Function Single-Stage(β, Q) in Algorithm 5.1 returns the solution to the single-stage problem, i.e., Equation (5.6). The run-time of the algorithm is dictated by the existent single-stage algorithm [31].

In Section 5.5, we show on an extensive set of experiments that the Two-stage-Horizon policy outperforms the Single-Stage policy in a wide range of β and γ values.

Now we establish the following result on the problem.

Algorithm 5.1

```
1: function TWO-STAGE-HORIZON($G, \beta, \gamma, Q$)
2:     if $\beta < 1$ then
3:         return SINGLE-STAGE($2\gamma\beta, Q$)
4:     else
5:         $H_1 \leftarrow$ SINGLE-STAGE($\frac{\beta+\gamma}{1+\gamma}, Q$)
6:         $H_2 \leftarrow$ SINGLE-STAGE($\beta + \gamma + \beta\gamma, Q$)
7:         Evaluate $H_1$ and $H_2$ for Problem (5.9) and return the minimally-valued configuration.
8:     end if
9: end function
```

Theorem 5.4.4. *Suppose there exists an α-approximation algorithm for the single-stage problem, then Algorithm 5.1 is a 5α-approximation algorithm for the two-stage problem.*

To prove the theorem, we are required four intermediate results.

We divide our analysis into two cases i) $\beta < 1$ and ii) $\beta \geq 1$. First we establish the following observation on the single-stage problem with $\beta < 1$.

Observation 5.4.5. *If $\beta < 1$, then $\Pi_1^*(Q) = Q$ for all $Q \in \mathcal{Q}$.*

Proof. The statement $\Pi_1^*(Q) = Q$ is equivalent to $\beta \sum_{u\in\mathcal{V}} p_a(u)d(Q,u) \leq \text{Assgn}(Q,F) + \beta \sum_{u\in\mathcal{V}} p_a(u)d(F,u)$, for all $Q, F \in \mathcal{Q}$. Now consider a task at u and the closest robot to u in configuration Q, namely $\sigma_Q(u)$, then we have $d(u, \sigma_Q(u)) \leq d(u, v), \ \forall v \in Q$.

Let $v \in Q$ be the vertex matched with $\sigma_F(u)$ in the assignment $\text{Assgn}(F,Q)$. Therefore, we have,

$$d(u, \sigma_Q(u)) \leq d(u, v) \leq d(v, \sigma_F(u)) + d(u, \sigma_F(u)), \tag{5.10}$$

where the last inequality is due to the triangle inequality. By taking the expectation on both sides of Equation (5.10) we have,

$$\mathcal{D}(Q) \leq \text{Assgn}(Q,F) + \mathcal{D}(F).$$

Therefore, the result follows immediately from $\beta < 1$. □

According to Observation 5.4.5, the optimal policy for the single-stage problem with $\beta < 1$ is to stay at the configuration and wait for the next task arrival.

Under Observation 5.4.5, the two stage problem becomes

$$\min_{Q_i \in \mathcal{Q}} \text{StageCost}(Q_{i-1}, Q_i) + \gamma\beta \sum_{v \in \mathcal{V}} p_a(v)\mathcal{D}(Q_{i,v}). \tag{5.11}$$

Now consider the following single-stage problem:

$$\min_{Q_i \in \mathcal{Q}} \text{StageCost}(Q_{i-1}, Q_i) + 2\gamma\beta\mathcal{D}(Q_i) \tag{5.12}$$

Observe that Problem (5.12) is the single-stage problem in Line 3 of Algorithm 5.1. Now we provide the following result on the quality of the solution obtained from Problem (5.12) as a solution for Problem (5.11).

Lemma 5.4.6. *Suppose there exists an α-approximation algorithm for single-stage problem, then the solution obtained from Problem (5.12) provides an $\alpha(1+2\gamma)$-approximation for Problem (5.11).*

Proof. Let Q_i be the solution of the α-approximation algorithm to Problem (5.12), and Q_i^* be the optimal solution to Problem (5.11), then we have,

$$\text{StageCost}(Q_{i-1}, Q_i) + 2\gamma\beta\mathcal{D}(Q_i) \leq \alpha(\text{StageCost}(Q_{i-1}, Q_i^*) + 2\gamma\beta\mathcal{D}(Q_i^*)). \tag{5.13}$$

Observe that by the triangle inequality we have $\sum_{v \in \mathcal{V}} p_a(v)\mathcal{D}(Q_{i,v}) \leq 2\gamma\beta\mathcal{D}(Q_i)$, then by Equations (5.13) it holds that,

$$\begin{aligned}\text{StageCost}(Q_{i-1}, Q_i) + \gamma\beta \sum_{v \in \mathcal{V}} p_a(v)\mathcal{D}(Q_{i,v}) \leq \\ \alpha(1+2\gamma)\big(\text{StageCost}(Q_{i-1}, Q_i^*) + \gamma\beta \sum_{u \in \mathcal{V}} p_a(u)\mathcal{D}(Q_{i,u}^*)\big).\end{aligned}$$ □

Hence, Algorithm 5.1 is a 3α-approximation for the two-stage problem with $\beta < 1$. Consider the following problem:

$$\min_{Q_i, Q_{i+1} \in \mathcal{Q}} \text{StageCost}(Q_{i-1}, Q_i) + \gamma\mathcal{D}(Q_i) + \gamma\text{StageCost}(Q_i, \Pi_1(Q_i)). \tag{5.14}$$

Now we establish the following result on Problem (5.14).

Lemma 5.4.7. *Suppose there exists an η-approximation algorithm for Problem* (5.14)*, then there exist $(1+2\gamma/\beta)\eta$-approximation algorithm for Problem* (5.9)*.*

Proof. Consider Q_i, Q_{i+1} as the configurations returned by the approximation algorithm for Problem (5.14) and Q_i^* be the optimal configuration for Problem (5.9). By the assumption of the Lemma, we have,

$$\begin{aligned}&\mathrm{StageCost}(Q_{i-1}, Q_i) + \gamma\mathcal{D}(Q_i) + \gamma\sum_{u\in\mathcal{V}} p_a(u)\mathrm{StageCost}(Q_i, \Pi_1(Q_i)) \leq \\ &\eta\big(\mathrm{StageCost}(Q_{i-1}, Q_i^*) + \gamma\mathcal{D}(Q_i^*) + \gamma\sum_{u\in\mathcal{V}} p_a(u)\mathrm{StageCost}(Q_i^*, \Pi_1^*(Q_i))\big).\end{aligned} \tag{5.15}$$

And also observe that by the triangle inequality we have,

$$\begin{aligned}&\mathrm{StageCost}(Q_{i-1}, Q_i^*) + \gamma\mathcal{D}(Q_i^*, u) + \gamma\sum_{u\in\mathcal{V}} p_a(u)\mathrm{StageCost}(Q_i^*, \Pi_1^*(Q_{i,u})) \leq \\ &\mathrm{StageCost}(Q_{i-1}, Q_i^*) + 2\gamma\mathcal{D}(Q_i^*, u) + \gamma\sum_{u\in\mathcal{V}} p_a(u)\mathrm{StageCost}(Q_{i,u}^*, \Pi_1^*(Q_{i,u})).\end{aligned} \tag{5.16}$$

Finally by Equations (5.15) and (5.16) we have,

$$\begin{aligned}&\mathrm{StageCost}(Q_{i-1}, Q_i) + \gamma\mathcal{D}(Q_i) + \gamma\sum_{u\in\mathcal{V}} p_a(u)\mathrm{StageCost}(Q_i, \Pi_1(Q_i)) \leq \\ &\eta(1+\frac{2\gamma}{\beta})\big(\mathrm{StageCost}(Q_{i-1}, Q_i^*) + \gamma\sum_{u\in\mathcal{V}} p_a(u)\mathrm{StageCost}(Q_i^*, \Pi_1^*(Q_{i,u}))\big).\end{aligned}$$

□

Note that Problem (5.14) is not a single-stage problem, and similar to the previous analysis, we will define two alternative problems to Problem (5.14), and then we will evaluate the quality of the solutions of the problems. Consider the following two problems:

$$\min_{Q_i\in\mathcal{Q}} \mathrm{Assgn}(Q_{i-1}, Q_i) + (\beta+\gamma+\beta\gamma)\mathcal{D}(Q_i) \tag{5.17}$$

$$\min_{Q_i\in\mathcal{Q}} \mathrm{Assgn}(Q_{i-1}, Q_i) + \frac{\beta+\gamma}{1+\gamma}\mathcal{D}(Q_i) + \frac{\gamma}{1+\gamma}\mathrm{StageCost}(Q_{i-1}, \Pi_1(Q_{i-1})) \tag{5.18}$$

Observe that in Problem (5.18), the first two terms and the third term are two independent single-stage problems, therefore, it can be approximately solved under assumption of Theorem 5.4.4 with two independent calls to the single-stage approximation algorithm.

Lemma 5.4.8. *Suppose there exists an α-approximation algorithm for single-stage problem, then*

(i) a solution to Problem (5.17) provides an $\alpha\beta$-approximation for Problem (5.14); and

(ii) a solution to Problem (5.18) provides an $\alpha(1+2\gamma)$-approximation for Problem (5.14).

Proof. (i) Suppose Q_i be the solution obtained from the single-stage approximation algorithm for Problem (5.17). By the triangle inequality we have,

$$\text{StageCost}(Q_{i-1}, Q_i) + \gamma\mathcal{D}(Q_i) + \gamma\text{StageCost}Q_i, \Pi_1(Q_i)) \leq \text{Assgn}(Q_{i-1}, Q_i) + (\beta + \gamma + \beta\gamma)\mathcal{D}(Q_i). \tag{5.19}$$

Also observe that by the triangle inequality we have,

$$\begin{aligned}\text{Assgn}(Q^*_{i-1}, Q^*_i) + (\beta + \gamma + \beta\gamma)\mathcal{D}(Q^*_i) \leq \text{StageCost}(Q_{i-1}, Q^*_i) + \gamma\mathcal{D}(Q^*_i, u) \\ + \beta\gamma\text{StageCost}(Q^*_i, \Pi^*_1(Q_i)).\end{aligned} \tag{5.20}$$

Finally, by applying the α-approximation factor of single-stage algorithm to Equations (5.19) and (5.20) we have,

$$\begin{aligned}&\text{StageCost}(Q_{i-1}, Q_i) + \gamma\mathcal{D}(Q_i) + \gamma\text{StageCost}(Q_i, \Pi_1(Q_i)) \leq \\ &\alpha\beta\big(\text{StageCost}(Q^*_{i-1}, Q^*_i) + \gamma\mathcal{D}(Q^*_i) + \gamma\text{StageCost}(Q^*_i, \Pi^*_1(Q^*_i))\big).\end{aligned}$$

(ii) Suppose Q_i and $\Pi_1(Q_{i-1})$ be the solutions obtained from the single-stage approximation algorithm for the two independent single-stage problems in Problem (5.17). By triangle inequality, we have

$$\begin{aligned}\text{StageCost}(Q_{i-1}, Q_i) + \gamma\mathcal{D}(Q_i) + \gamma\text{StageCost}(Q_i, \Pi_1(Q_i)) \leq \\ (1+\gamma)\text{Assgn}(Q_{i-1}, Q_i) + (\beta + \gamma)\mathcal{D}(Q_i) + \gamma\text{StageCost}(Q_{i-1}, \Pi_1(Q_{i-1})).\end{aligned} \tag{5.21}$$

Observe that by the triangle inequality we have,

$$\begin{aligned}(1+\gamma)\text{Assgn}(Q^*_{i-1}, Q^*_i) + (\beta + \gamma)\mathcal{D}(Q^*_i) + \gamma\text{StageCost}(Q^*_{i-1}, \Pi^*_1(Q^*_{i-1})) \leq \\ \text{StageCost}(Q_{i-1}, Q^*_i) + \gamma\mathcal{D}(Q^*_i) + \gamma\text{StageCost}(Q^*_i, \Pi^*_1(Q_i)) + 2\gamma\text{Assgn}(Q_{i-1}, Q^*_i).\end{aligned} \tag{5.22}$$

Finally, by applying the α-approximation factor of single-stage algorithm to Equations (5.21) and (5.22) we have,

$$\begin{aligned}&\text{StageCost}(Q_{i-1}, Q_i) + \gamma\mathcal{D}(Q_i) + \gamma\text{StageCost}(Q_i, \Pi_1(Q_i)) \leq \\ &\alpha(1+2\gamma)\big(\text{StageCost}(Q_{i-1}, Q_i^*) + \gamma\mathcal{D}(Q_i^*) + \gamma\text{StageCost}(Q_i^*, \Pi_1^*(Q_i))\big). \qquad \square\end{aligned}$$

With these results we now prove Theorem 5.4.4.

Proof of Theorem 5.4.4. Algorithm 5.1 evaluates the solution for the two problems in Line 7 and returns the better solution. An immediate result of Lemmas 5.4.7 and 5.4.8, is that the approximation factor for Algorithm 5.1 with $\beta \geq 1$ is

$$\min\{\alpha\beta, \alpha(1+2\gamma)\}(1+2\frac{\gamma}{\beta}) \leq \alpha(1+4\gamma) \leq 5\alpha. \qquad \square$$

A $3+o(1)$- approximation algorithm for single-stage problem and k-median is provided in [31]. Built on this result, Algorithm 5.1 is a $9+o(1)$ approximation algorithm for the Two-stage-Horizon with $\beta < 1$ and $15+o(1)$ with $\beta > 1$.

5.4.3 Policy II: Move to k-median

An intuitive policy for Problem (5.1) is to move the robots to the k-median solution in between the arrivals. In fact it is shown to be the optimal policy in the light load if the transition cost between the configurations is negligible compared to the service cost [14]. Let Π_{median} denote this policy, and $V_{\text{median}}(Q)$ be the value function under this policy. Then we establish the following result.

Lemma 5.4.9. *Suppose an α-approximation algorithm for k-median returns Q, then the value function under Π_{median} is at most $\alpha(1+\gamma/\beta)$ times the optimal, i.e.,*

$$V_{\text{median}}(Q) \leq \alpha(1+\frac{\gamma}{\beta})V(Q).$$

Proof. First observe that the value function under this policy, namely $V_{\text{median}}(Q)$, is as follows:

$$V_{\text{median}}(Q) = \text{StageCost}(Q, Q) + \gamma\sum_{u\in\mathcal{V}} p_a(u)V_{\text{median}}(Q_u).$$

Also notice that the $V_{\text{median}}(Q_u)$ is bounded as follows, where the equality is ensured if Q is the only k-median.

$$V_{\text{median}}(Q_u) \leq \text{StageCost}(Q_u, Q) + \gamma \sum_{v \in \mathcal{V}} p_a(v) V_{\text{median}}(Q_v).$$

Therefore, With the same stages in the proof of Lemma 5.4.2, we have

$$V_{\text{median}}(Q) \leq \frac{\beta + \gamma}{1 - \gamma} \mathcal{D}(Q).$$

Also from Lemma 5.4.3 we have, $V(Q) \geq \frac{\beta}{1-\gamma}\mathcal{D}(M)$, where M is the actual k-median. Since Q is an α-approximation for the k-median,

$$V_{\text{median}}(Q) \leq \alpha(1 + \frac{\gamma}{\beta})V(Q). \qquad \square$$

In fact, the bound shows that the value for this policy approaches the optimal for large β. This result captures the optimal policy results for DVR in [14]. However, the policy can be arbitrarily sub-optimal for small values of β. For instance, consider a problem with $\beta = 0$ and $\gamma > 0$. In this case $V(Q) = 0$ while $V_{\text{median}}(Q) > 0$ due to the transitions to the k-median between task arrivals.

The policy does not provide desired behavior for small values of β. In the next section, we provide simulation results for two policies in the infinite horizon setting and evaluate their performance for different values of γ and β.

5.5 Simulation Results

In this section, we evaluate the performance of the two policies in Section 5.4 in random environments and a sample work place (see Figure 5.3). To simulate the real-world applications, we also evaluate the performance of the policies in environments with unknown probability distribution and high task arrival rates.

In all the experiments we consider the infinite horizon problem, and the expected cost at each Q for policy Π is obtained by solving the following set of equations, known as policy evaluation:

$$V(Q) = \text{StageCost}(Q, \Pi(Q)) + \gamma \sum_{u \in \mathcal{V}} V(\Pi(Q)_u) \quad \forall Q \in \mathcal{Q}.$$

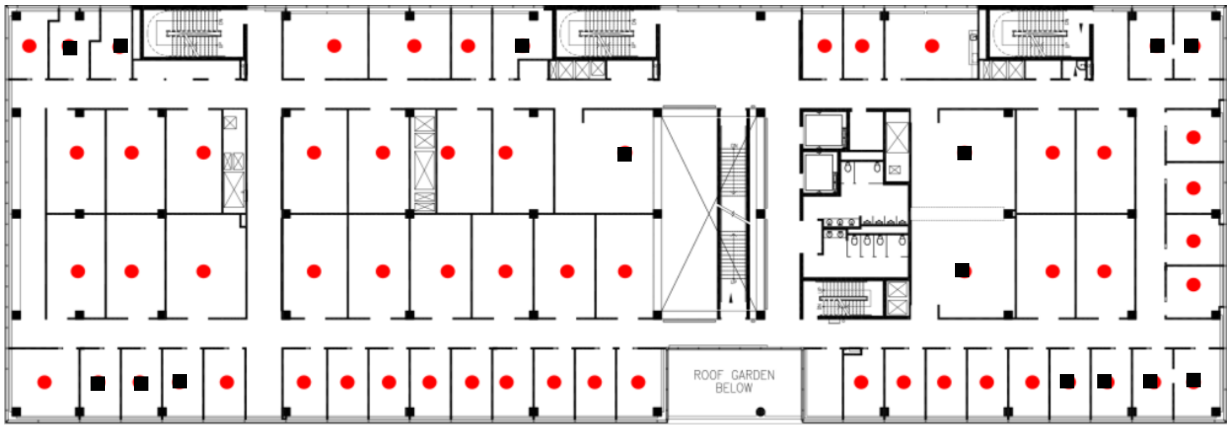
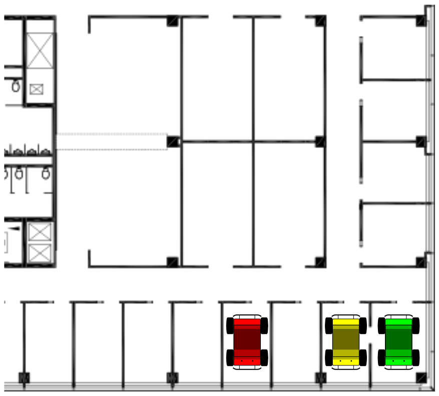


Figure 5.3: Floor plan of E5 building at University of Waterloo. Floor plan contains 62 task locations (circles and squares) and 15 potential waiting locations for the robots (squares) (b) A stage of the two-stage policy performing a task and redeploying.

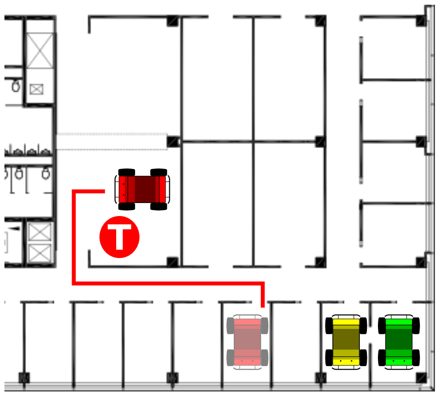
Figure 5.4 shows two stage of the Two-Stage-Horizon policy in a part of the workplace floor plan with $\beta = 5$ and $\gamma = 0.9$. The circles represent the tasks. In stage 1 the red robot services a task and to redeploy and the green robot moves to the new location. In stage 2, the yellow robot services a task while the green robot moves to the new waiting location.

Random Environment: In this experiment, we compare the average expected cost of all configurations obtained from different policies to the optimal value from DP in random environments. In each instance, 4 robots are servicing a set of 20 tasks randomly generated in the Euclidean plane of size 10×10. The probability of a task arriving in each location is independent and identically distributed. Figure 5.5 shows the average error percentage of three policies for different values of β and $\gamma = 0.9$. For each β, a set of 2000 random instances are generated and each box depicts the median, first quartile and third quartile. The DP is solved via the policy iteration method with an average time of 894.4 seconds on an Intel Core$i$5 @3.6Ghz processor. The average computation time of the Two-stage-Horizon policy and the Move-to-Median policy for a single configuration are 0.0078 seconds and 0.0027 seconds, respectively.

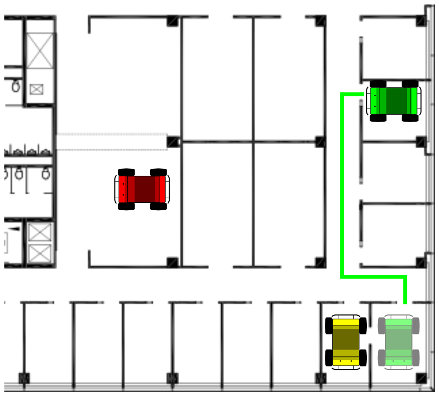
Figure 5.5a illustrates the asymptotic improvement in the performance of the Move-to-Median policy as β increases. In contrast, the greedy Single-Stage policy performs almost optimally when the deployment cost and response time are equally weighted. However, the Two-stage-Horizon policy provides a solution within 5% of the optimal for all β, out-performing Move-to-Median policy at $\beta = 2$ and single stage policy at $\beta = 5$



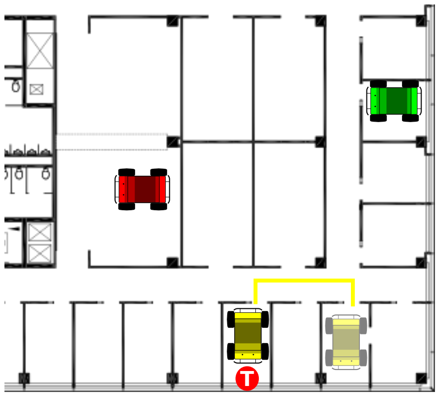
(a) Initial configuration



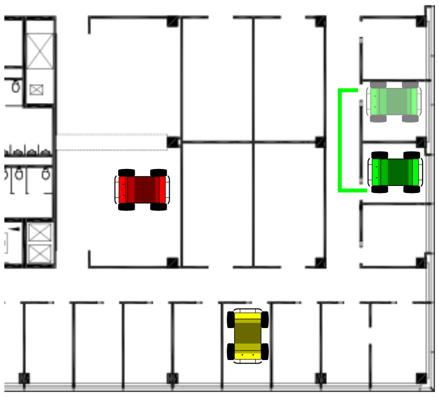
(b) A task arrives and the closest robot (red) is deployed



(c) Robots relocate and wait for the next arrivals

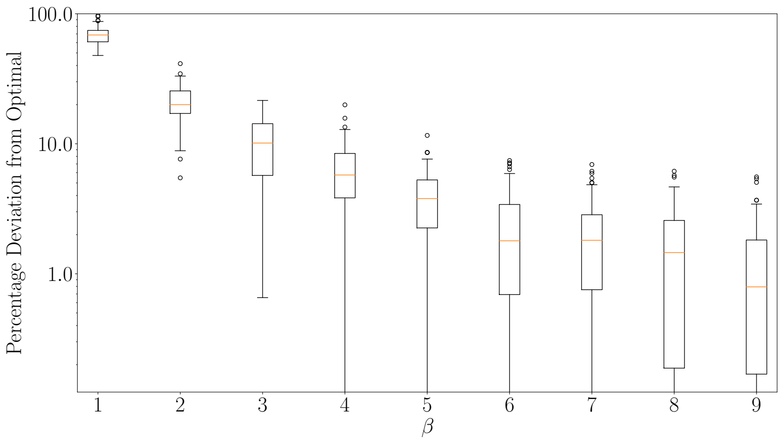


(d) A task arrives and the closest robot (yellow) is deployed

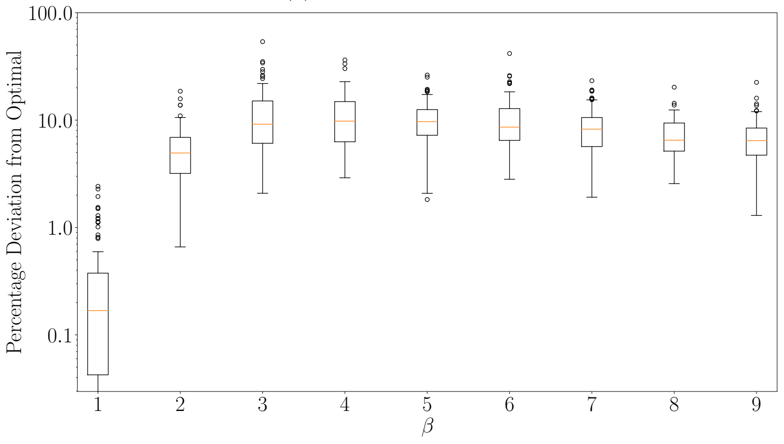


(e) Robots relocate and wait for the next arrivals

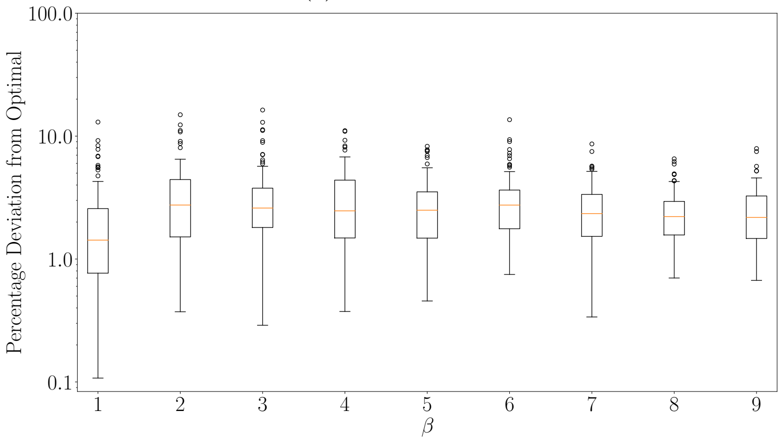
Figure 5.4: Two stages of the two-stage policy performing two tasks and two re-deployments.



(a) Move-to-Median



(b) Single-Stage



(c) Two-stage-Horizon

Figure 5.5: Average deviation from optimal for different policies

by 18.1% and 10.1% on average, respectively.

We observed the similar behavior of the three policies when varying γ. For small γ, the single-stage policy provides near-optimal solutions. As γ increases, the Move-to-Median policy improves in quality. Similar to the previous experiment, the Two-stage-Horizon policy provides better solutions in for a large range of γ values.

Unknown Probability Distribution: In this experiment, the spatial probability distribution of the tasks is unknown to the robots. The robots, starting with a uniform Dirichlet prior (with all parameters equal to 2), and update their estimate of the distribution according to the maximum a posteriori estimator [78]:

$$\hat{p}_u = \frac{\text{number of arrivals at } u + 1}{\text{number of arrivals} + \text{number of task locations}}.$$

The environment is a floor plan with 62 task locations and 5 robots. In some applications, e.g. hospital setting, the robots cannot stay at the task locations, therefore, the potential waiting location of the robots are limited to 15 locations marked with squares in Figure 5.3. The probability distribution is drawn from a Dirichlet distribution where the task locations are weighted with their x-coordinates, i.e. the probability of a task is higher for vertices in the right side.

We compare performance to a policy that has access to the true distribution over 80 task arrivals. Figure 5.6 shows the average total stage costs for $\beta = 5$ and 10 over 15000 instances. Observe that a poor estimation of the probability distribution in the first few arrivals propagates proportionally with the value of β. The Move-to-Median policy is equivalent to the two-stage policy for sufficiently large β. Therefore, the Move-to-Median is more vulnerable to error in the estimate.

Different Arrival Times: In Section 5.2, we assumed that the time between arrivals is sufficient for the robots to reconfigure between the arrivals. Now we repeat the experiment with $\beta = 5$ and $\gamma = 0.9$ on the environment given in Figure 5.3 for different task arrival rates. The time to service a task at u is the difference between the arrival time and the time that a robot visits the task location. We perform 15,000 simulations, each containing a sequence of 80 tasks arrivals according to the Poisson process with parameter λ, where λ represents the number of tasks arrive in a time unit. The *time unit* on the x-axis is the expected time to service a task from the k-median and return, i.e. $\mathcal{D}(M)$. Figure 5.7 shows the average stage cost of the two policies. The stage cost increases exponentially with λ. Notice that when there are approximately 2 arrivals per time unit, and thus new tasks frequently arrival while the robots are redeploying, the average stage cost increases by less than a factor of two from the cost for $\lambda \approx 0$ (i.e., the regime where our assumption holds).

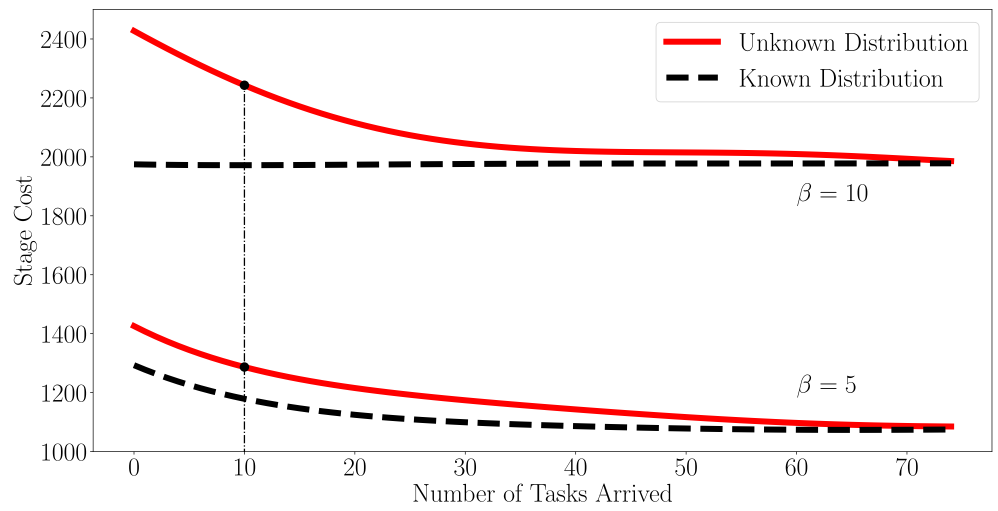


Figure 5.6: TWO-STAGE-HORIZON with unknown probability distribution.

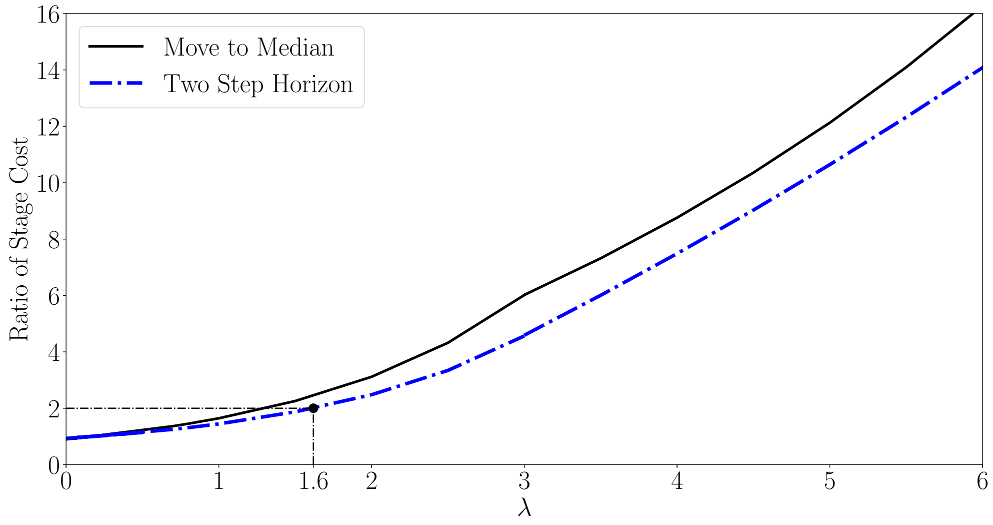


Figure 5.7: Stage cost of TWO-STAGE-HORIZON and MOVE-TO-MEDIAN policies as a function of task arrival rate.

Chapter 6

Re-Balancing Self-Interested Drivers in Transportation Networks

In recent years, ride-sourcing services such as UberX and Lyft have emerged as an alternative mode of urban transportation. The compelling feature of these services compared to the conventional taxi services is the improved service quality such as the expected wait time for pick-up [79]. The wait-time of a pick-up for a ride request is a function of the distribution of the drivers in the environment. Therefore, the objective of the ride-sourcing company is to distribute the drivers in the environment to improve the service quality. However, the ride-sourcing company does not have control over the positions of the drivers as they are self-interested units maximizing their objectives. Therefore, the challenge is to ensure the service quality by indirect controls on the positions of the drivers.

The surge pricing in high demand areas is an instance of a conventional indirect control method for both re-balancing the drivers and increasing the supply of drivers which is employed by Uber. Although this control method reduces the expected response time of servicing the requests by drawing more drivers to the high demand areas, it might draw drivers away from lower demand areas, resulting in higher wait times in those areas and more imbalance [80, 81].

The problem of servicing ride requests in ride-sourcing networks consists of two major problems: 1) assignment of the current ride requests to the drivers; and 2) rebalancing of the drivers for the future ride-requests. The main focus of this chapter is the latter where we re-balance a subset of drivers to service ride-requests arriving sequentially in an environment.

The environment is represented by a graph and the ride-requests arrive on the nodes of

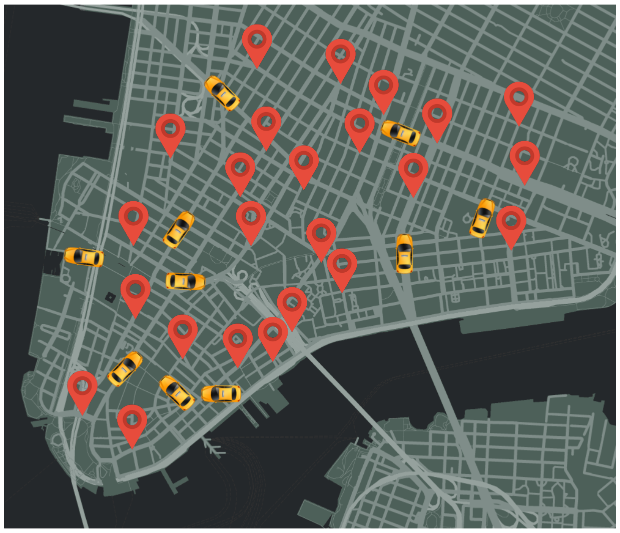


Figure 6.1: A set of drivers in the ride-sharing system and a set of locations with high probability of ride request arrival.

the graph according to a known arrival rate (see Figure 6.1). The drivers are self-interested units maximizing their expected profit by choosing their location in the environment to wait for the next ride request. Hence, the ride-sourcing company has no direct control over the waiting locations of the drivers. Therefore, the objective of the ride-sourcing company is to incentivize the drivers to relocate to a set of waiting locations that maximize the service quality.

The contributions of this chapter are three-fold. First, we formulate the ride-sharing problem with self-interested units and two global objectives; minimizing the expected wait time, and minimizing the maximum wait time of the ride request. We prove the NP-hardness of these problems. Second, we propose two indirect control methods to relocate the drivers in the environment: 1) sharing the location of all drivers with a subset of drivers, and 2) paying the drivers to relocate. Third, we develop novel algorithms for each objective and control method combination. For the problem of minimizing the expected wait-time under information sharing, we propose an LP-rounding algorithm that provides near-optimal solutions in an extensive set of experiments. We provide a 3-approximation algorithm for the problem of minimizing the maximum wait-time under information sharing. To find the optimal payment in the second control method, we first cast the problem as a game between the drivers and the service provider, where the service provider seeks to minimize a linear combination of the total amount paid and the global objective. We provide 3-approximation algorithms to find the optimal control for both global objectives.

Finally, we evaluate the performance of the proposed control methods on real-world ride-sharing data for the two global objectives.

This chapter is organized as follows. In Section 6.2, we formulate the problem of minimizing the expected or the maximum wait-time of customers with self-interested drivers. In Section 6.3, we propose the first indirect control method to relocate the drivers based on sharing the information on the drivers with a subset of them. In Section 6.4.1, we provide the second control method based on incentive pay. Finally in Section 6.5, we evaluate the performance of the proposed control methods on an extensive set of experiments with real-world ride-sharing data.

6.1 Related Work

The problem of dispatching taxis to service ride requests arriving sequentially over time has been the subject of extensive research [82, 83, 84, 85]. These studies focus on policies to optimally assign the ride requests to the taxis. In contrast, we focus on the waiting locations of the drivers that minimize the expected wait time or the maximum wait-time of the customers. We assume the ride requests are assigned in a first-come-first-serve fashion to the closest available driver which is a common method employed by the ride-sharing companies [86].

The problem of rebalancing service units in the environment has been studied for various applications. In the mobility-on-demand problem (MOD) [11, 87, 33, 34], a group of vehicles are located at a set of stations. The customers arrive at the stations, hire vehicles for a ride, and then drop the vehicles off at the station closest to their destination. The objective is to balance the vehicles at the stations to minimize the expected wait time of the customer. In comparison to MOD, we consider the customer wait-time as the time between the request arrival and the pick-up time, which incorporates the distance of the closest available vehicle to the pick-up location. In [88], the authors focus on the intersection of the MOD systems and the public transportation where the ride-sharing company, costumers and the municipal transportation authority are self-interested units. In this study, the authors provide a pricing scheme for the ride-sharing company to maximize service quality. The MOD methods consider the macroscopic aspect of the ride-sharing problem where a flow formulation is provided to approximate the average number of vehicles to relocate from a station to others. In contrast to the MOD approaches, our results capture the microscopic aspect of ride-sharing, focusing on the movement of individual drivers.

A conventional method for relocating the drivers in a ride-sharing network is by the surge pricing method in high-demand areas. The problem of pricing ride requests in a ride-sharing system has been studied recently in the literature [35, 36, 37]. In [36], the authors study the ride-sharing problem in a ride-sharing network where the service units are a combination of the self-interested drivers and autonomous vehicles. The authors propose a pricing scheme and a payment method for the self-interested drivers to rebalance in the network. However, increasing the price of rides in high-demand areas to incentivize the drivers to relocate to those areas decreases the demand [36]. In contrast, we propose an indirect control method for relocating the drivers by sharing information on the position of the drivers to a subset of them and steer them towards the areas with higher demand and lower supply.

The facility location problem [55, 28] and its extension to the mobile facility location problem (MFL) [29] is the problem of distributing facilities in a set of locations to respond to the demands arriving at different locations. The objective is to minimize the time to respond to the demands and the total cost of opening facilities. A special case of the facility location problem is the k-median problem [28] where the number of open facilities is limited and the cost of opening a facility is zero. In Chapter 5, we addressed a multi-stage MFL problem where we relocate a set of autonomous vehicles to minimize expected response time for future requests in a receding horizon manner. However, a key difference in MFL problems is that the objective of the service units are aligned with the service provider, and thus the waiting locations of service units can be directly controlled.

A closely related problem is that of Voronoi games on graphs [89] where requests arrive on the vertices of a graph and the objective of each self-interested service unit is to maximize the number vertices assigned to them. This work casts the problem as a game between the service units prove that the problem of finding the pure Nash equilibrium on general graphs is NP-hard. In [90], the authors provide the best response strategy for each driver and they approximate Nash equilibria. These studies focus on the strategies of self-interested service units. In contrast, we focus on finding the optimal policy for the ride-sharing company to optimally respond to the ride requests.

6.2 Problem Formulation

In this section, we provide a detailed description of the environment model for the ride-sharing problem and the models for the self-interested drivers and the service provider.

6.2.1 Environment Model

Consider a complete metric graph $G = (\mathcal{V}, \mathcal{E}, c)$ where the vertex set $\mathcal{V}$ represents the pick-up and drop-off locations, the edge set $\mathcal{E}$ is the set of connections between the vertices and the function $c : \mathcal{E} \to \mathbb{R}_+$ assigns a travel time to each edge on the graph. There are m drivers in the environment responding to the ride requests arriving over time. The drivers wait on a subset of the vertices for the next request, which we call the configuration of the drivers Q. The set of all configurations of the drivers is denoted by $\mathcal{Q}$.

The ride requests arrive at each vertex u of the graph according to an independent process. Upon a request arrival, the closest driver to the vertex of the request is assigned to service the request. Let $p_a(u)$ denote the arrival probability, which represents the likelihood of ride request arriving at vertex u. Let the drop-off probability $p_d(\text{dropoff} = w|\text{pickup} = v)$ be the probability of a request with pickup location v and a drop-off location w.

6.2.2 Self-interested Drivers' Model

We assume that the drivers in the system act in their self-interest to maximize their profit. A driver i might be aware of the position of a subset of other drivers which we call the information of driver i and denote by I_i. The information I_i is the set of known positions of other drivers. For instance, each driver may be aware of the location of the other drivers in its vicinity. We assume that the information $I_i = \emptyset$ for all drivers unless it is provided by the centralized service provider.

Driver i's *perception of her expected profit* is a function of her information I_i on the location of other drivers, environment parameters such as arrival times, drop-off location probabilities, her current waiting location q_i and the period of working time B_i, denoted by $\mathcal{M}_i(u, B_i, I_i)$. Each driver is a self-interested unit, and thus will wait at a location that maximizes its expected profit, i.e.,

$$\arg\max_{u \in \mathcal{V}} -\sigma c(q_i, u) + \mathcal{M}_i(u, B_i - c(q_i, u), I_i), \tag{6.1}$$

where σ is the cost per minute of driving.

The remainder of this chapter and the proposed main control methods do not rely on any specific form of function $\mathcal{M}_i$. We do assume, however, that the ride-sharing company has access to this function, obtained through data of driver behavior. In Section 6.5.1 we present one potential model of $\mathcal{M}_i$, which is then used for simulating the two control methods.

6.2.3 Service Provider's Model

In addition to the objective of each driver, we consider a global objective $J(Q)$ to maximize the service quality. In this chapter, we focus on two global objectives in servicing tasks: 1) the expected wait-time and 2) the maximum wait-time of the ride requests.

The expected wait-time objective can be expressed as

$$J_{\exp}(Q) = \sum_{u \in \mathcal{V}} \min_{q_i \in Q} p_a(u) c(q_i, u). \tag{6.2}$$

Under global objective (6.2), the desired configuration of the drivers concentrates on the regions with higher arrival rates.

An alternative global objective is to minimize the maximum wait-time over all ride requests, i.e.,

$$J_{\max}(Q) = \max_{u \in \mathcal{V}} \min_{q_i \in Q} c(q_i, u). \tag{6.3}$$

Under this objective, the drivers provide a more uniform service quality at different locations regardless of the arrival rate at the locations.

The main challenge in optimizing the global objective is that the drivers are self-interested units and the service provider does not have any direct control over the configuration of the vehicles. Therefore, the service provider is not able to minimize the expected response time or the maximum response time to the requests directly. The two indirect control methods proposed in this chapter incentivize the drivers to relocate to desired waiting locations. The first control method exploits the dependency of the expected profit of the drivers on their information I_i. The service provider can share information on the location of drivers with a subset of them to manipulate their decision towards relocating to a desired waiting location. We refer to this as the *sharing information* control method. The second proposed control method, incentivizes the drivers to relocate to desired waiting locations with payments, which we refer to as the *pay-to-control* method. These control methods are applicable to various models of driver behavior $\mathcal{M}$. Table 6.1 summarizes the results provided on the proposed two control methods and the two global objectives.

In the following sections, we provide a detailed description of the two control methods and propose algorithms to find near-optimal controls.

Control method	J_{exp}	J_{max}
information sharing	LP-rounding	3-approx. algorithm
pay-to-control*	3-approx. algorithm	3-approx. algorithm

Table 6.1: Summary of algorithms proposed. *For the pay-to-control method we minimize a linear combination of the total amount paid and the global objective.

6.3 Control by Sharing Information

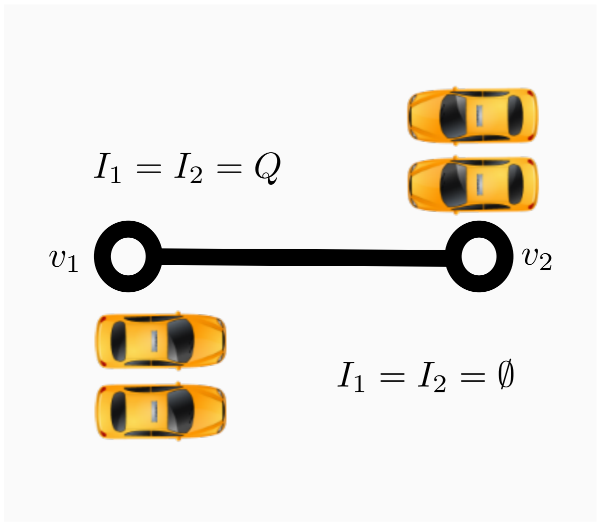
The method proposed in this section exploits the fact that the decision of the self-interested drivers on their optimal waiting location (see Equation (6.1)) is a function of the information provided to the driver regarding the position of the other drivers, i.e., I_i.

First we provide an example to demonstrate the importance of the information of the drivers in controlling their configuration in the environment.

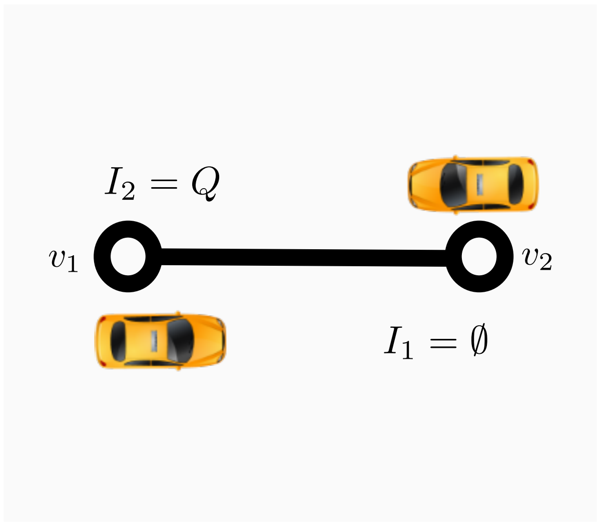
Example 6.3.1. Consider a ride-sharing system with two ride request locations and two vehicles (see Figure 6.2). The locations are unit distance apart and the arrival rate at locations v_1 and v_2 are 0.1 and 0.2, respectively. The cost of relocation for the drivers is $= 0$ and there is a driver model $\mathcal{M}_i > 0$. The vehicles are initially located at v_1 and will relocate to the best waiting location, namely optimizing Equation (6.1). Figure 6.2a shows the two scenarios where both vehicles are provided the same information, i.e., $I_1 = I_2 = \emptyset$ and $I_1 = I_2 = Q$. Note that the configuration of the vehicles when they are provided the same information is the worst possible configuration for the global objective J_{exp}. However, illustrated in Figure 6.2b, providing the information to a subset of the vehicles results in the optimal configuration for the global objective. •

6.3.1 Formal Definition and Complexity Class

The information sharing problem consists of 1) deciding the subset of drivers we share information with, and 2) deciding what information to share with each driver. The number of different information sets is exponential in the number of drivers. In this work we simplify 1) and 2) into a binary decision for each driver; either share full information or share no information. Theorem 6.3.4 shows that finding the optimal information control even with this binary decision is NP-hard. Moreover, our experiments on real-world ride-sharing data in Section 6.5 suggests that even limiting to the binary decision, we achieve significant improvement in the service quality.



(a) Equal information sharing



(b) Partial information sharing

Figure 6.2: Ride-sharing problem with two vehicles and two request arrival locations.

Remark 6.3.2 (Alternate information sets). The approach proposed in this section does not require that the binary choice is between the empty set and the full information set. One can replace this with any two subsets of the information set: For example, for each driver, the binary decision could be between the empty set and the set of all other driver positions within a certain radius. •

Let $q'_{i,Q}$ (resp. $q'_{i,\emptyset}$) be the new waiting location selected by driver i from Equation (6.1) with information $I_i = Q$ (resp. $I_i = \emptyset$). Let $F_i = \{q'_{i,Q}, q'_{i,\emptyset}\}$ be the set of candidate waiting locations for driver i. The formal definition of the problem of sharing information as follows:

Problem 6.3.3. *Consider a metric graph $G = (\cup_{i=1}^{m} F_i \cup \mathcal{V}, \mathcal{E}, c)$. Find a new configuration Q' by picking only one vertex from each F_i, i.e., such that $|Q' \cap F_i| = 1$ for each i, while minimizing the global objective $J_{\exp}(Q')$ or $J_{\max}(Q')$.*

Figure 6.3 shows an instance of the information-sharing problem. The green vertices are the waiting locations of the drivers if they have no information on the position of the other drivers, and the red vertices are the waiting locations of the drivers if the information is provided to each driver by the service provider. Let Q' be the solution to Problem 6.3.3 in which if $q_{i,Q} \in Q'$ then the driver i is provided complete information of the position of the other drivers, and no information is available for driver i if $q'_{i,\emptyset} \in Q'$.

First, we analyze the complexity of Problem 6.3.3 for the two global objectives (6.2)

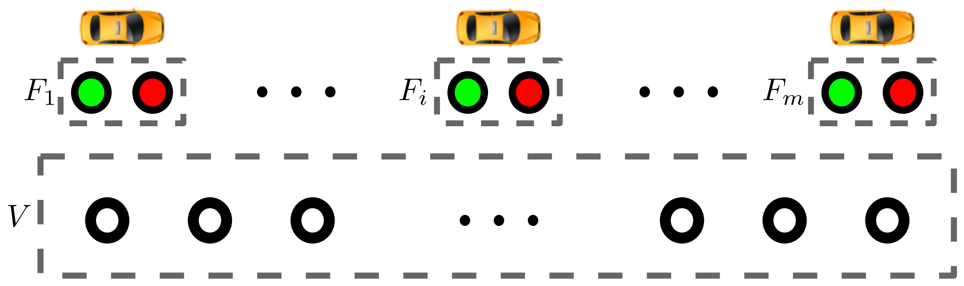


Figure 6.3: An instance of Problem 6.3.3. The green vertices represent the desired waiting location of each driver if $I_i = \emptyset$, and the red vertices represent the waiting location of the drivers if $I_i = Q$.

and (6.3), and then we provide our algorithm for each of the global objectives.

Theorem 6.3.4. *The problem of finding the optimal information sharing control, i.e. Problem 6.3.3, with either of the objectives of minimizing the expected wait-time $J_{\exp}$ or minimizing the maximum wait-time $J_{\max}$ is NP-hard.*

Proof. First we prove the NP-hardness of Problem 6.3.3 for minimizing the expected wait-time $J_{\exp}$ with a reduction from CNF-SAT [91] (see Section 2.4.5) as follows:

Consider an instance of CNF-SAT with n Boolean variables and m clauses. We will reduce this problem to Problem 6.3.3.

(i) Let $\cup_{i=1}^{m} F_i$ contain $2n$ vertices, partitioned into n sets of size two. The set F_i contains two vertices, $\{v_i^T, v_i^F\}$, where v_i^T will correspond to setting the ith SAT variable to true (i.e., the positive literal) and v_i^F will correspond to setting it to false (i.e., the negative literal).

(ii) We let $\mathcal{V}$ contain m vertices, one representing each clause in the SAT formula.

(iii) Let $\mathcal{E}$ contain an edge for each $v \in \cup_{i=1}^{m} F_i$ and $w \in \mathcal{V}$.

(iv) For each $e = (v, w) \in \mathcal{E}$, we set its cost to 1 if the literal v appears in the clause w, and 2 if the literal does not. Note that the costs are metric.

(v) Let $p_a(u) = 1/m$ for each $u \in \mathcal{V}$.

Now, we solve the instance of Problem 6.3.3. If it returns a subset of $\cup_{i=1}^m F_i$ with cost exactly 1, then for each clause $c \in \mathcal{V}$, there is literal in $\cup_{i=1}^m F_i$ with edge cost of 1 to c. This implies that the literal chosen from each subset in the partition of $\cup_{i=1}^m F_i$ gives a satisfying truth assignment for the SAT instance. If the subset returned has a cost greater than 1, then there exists a clause $w \in \mathcal{V}$ for which every chosen literal has an edge cost of 2. Thus, this clause is not satisfied and no satisfying instance exists.

The proof of hardness for Problem 6.3.3 with the objective $J_{\max}$ follows the same steps with exception of step (v) where $p_a(u) = 1$ for each $u \in \mathcal{V}$. □

6.3.2 Minimizing The Expected Wait Time

Given that Problem 6.3.3 is NP-hard for both objectives, we turn our focus to sub-optimal algorithms. In particular, we provide a simple Linear Program (LP)-rounding algorithm for Problem 6.3.3. Although we do not provide bounds on the performance of the LP-rounding algorithm, we evaluate the performance of the algorithm on an extensive set of real-world ride-sharing data in Section 6.5 and we show that the proposed algorithm is on average within 0.021% of the optimal.

First we cast Problem 6.3.3 as an integer linear program (ILP), then we propose a rounding algorithm based on the solution to the relaxation of the ILP. Let integer parameter $x_{u,v} \in \{0,1\}$ denote the assignment of a request at v to a driver at vertex u if $x_{u,v} = 1$, and $x_{u,v} = 0$ otherwise. Let the integer parameter $y_u \in \{0,1\}$ for all $u \in \cup_i^m F_i$ represent if there is a driver assigned to wait for next request arrival at u. Then we write the ILP for Problem 6.3.3 as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{v\in\mathcal{V}} \sum_{u \in \cup_i^m F_i} p_a(v) c(u,v) x_{u,v} && (6.4a)\\
\text{subject to} \quad & \sum_{u \in \cup_i^m F_i} x_{u,v} \geq 1, \forall v \in \mathcal{V}, && (6.4b)\\
& x_{u,v} \leq y_u, \forall v \in \mathcal{V}, \forall u \in \cup_i^m F_i, && (6.4c)\\
& y_u + y_v = 1, F_i = \{u,v\}, i \in [m], && (6.4d)\\
& y_u, x_{u,v} \in \{0,1\}, \forall v \in \mathcal{V}, \forall u \cup_i^m F_i. && (6.4e)
\end{aligned}$$

By constraint (6.4b), a feasible solution assigns each request location to a driver. Equation (6.4c) ensures that a request is assigned to u only if there is a driver located at u, and finally Equation (6.4d) shows that in a feasible solution only one of the candidate waiting locations is chosen from each subset F_i, which represent that either the information is provided to a driver or otherwise.

Now we propose our LP-rounding algorithm for Problem 6.3.3. Let $(\mathbf{x}', \mathbf{y}')$ be the solution to the LP relaxation of ILP (6.4). Without loss of generality for all $F_i = \{u, v\}, i \in [m]$, let $y_u \geq 1/2$ and $y_v \leq 1/2$. Given solution $(\mathbf{x}', \mathbf{y}')$ we construct an integer solution to ILP (6.4) by setting $y_u = 1$ for each vertex u with $y'_u > 1/2$ and $y_v = 0$. In a case, $F_i = \{u, v\}$ and $y'_u = y'_v = 1/2$, we set $y_u = 1$ where u is the optimal waiting location of driver i with $I_i = \emptyset$. Then we assign each vertex $v \in \mathcal{V}$ to the closest vertex u in $\cup_{i=1}^{m} F_i$ with $y_u = 1$ by setting $x_{u,v} = 1$. Note that the constructed solution (x, y) satisfies the constraint of ILP (6.4), therefore, it is a feasible solution to Problem 6.3.3. Also, observe that the optimal objective value to the LP relaxation is a lower-bound on the optimal value of ILP (6.4) and provides a bound on the performance of the LP-rounding algorithm.

In the solution to the information-sharing problem, if driver i is selected to receive information on the location of drivers, a snapshot of the location of drivers is presented to driver i and the driver can calculate their expected profit based on complete information. This method employed at each time step and presents information to a driver if there is an opportunity to improve the expected response time.

6.3.3 Minimizing The Maximum Wait-Time

In this section, we propose an algorithm for Problem 6.3.3 with the objective of minimizing the maximum wait-time. We also prove that the solution provided by the proposed algorithm is within a factor of 3 of the optimal control.

The proposed algorithm makes a guess on the optimal maximum wait time of the demands on the vertices, removes the edges longer than the guess, then tries to find a subset of $\cup_i^m F_i$ in the resulting graph such that there is an edge between any $v \in \mathcal{V}$ and a vertex in the subset.

Let T be our guess for the maximum wait-time in the optimal solution of Problem 6.3.3. Let $H = (\mathcal{V}, \mathcal{E}_H)$ be the induced graph of G by deleting the edges longer than T, i.e., $E_H = \{e \in \mathcal{E} | c(e) \leq T\}$. Let $\mathcal{N}(v)$ denote the vertices in $\cup_{i \in [m]} F_i$ with an edge incident to vertex v in graph H. At any step of the algorithm, if there exist a $v \in \mathcal{V}$ such that $|\mathcal{N}(v)| = 0$, then we reached a *conflict*, and we increase our guess T. The approximation algorithm for Problem 6.3.3 with the objective of minimizing the maximum time consists of the following two subroutines:

Subroutine I: For any vertex v with $|\mathcal{N}(v)| = 1$, in a sequential manner, we add the vertex $u \in \mathcal{N}(v)$ to our solution. At each step, we declare the vertices in $\mathcal{V}$ within distance $3T$ of u as serviced. Also, we remove the vertex $w = F_i \setminus \{u\}$ and all the edges incident to it, and we update $\mathcal{N}(v)$ for all $v \in \mathcal{V}$. If at any stage of this process, there is a v with $|\mathcal{N}(v)| = 0$, then there exists a conflict and we increase our guess T.

Subroutine II: Assuming that Subroutine I is completed without any conflicts, then at the start of the Subroutine II of the algorithm all the vertices in $\mathcal{V}$ that are not serviced at the end of Subroutine I have $|\mathcal{N}(v)| \geq 2$. Starting from an arbitrary vertex $v \in \mathcal{V}$, we add a vertex $u \in \mathcal{N}(v)$ to the solution and declare all the vertices in $\mathcal{V}$ within distance $3T$ of u as serviced. We also, remove $w = F_i \setminus \{u\}$ and all the edges incident to it, and we update $\mathcal{N}(v)$ for all $v \in \mathcal{V}$. Then we execute the Subroutine I for all the vertices with $|\mathcal{N}(v)| = 1$. Subroutine II continues until all the vertices in $\mathcal{V}$ are serviced.

The algorithm performs a binary search on $T \in [\min_{u,v\in\mathcal{V}} c(u,v), \max_{u,v} c(u,v)]$ to find the optimal maximum wait-time T^*. Observe that there is no conflict in Subroutine I for any $T \geq T^*$, therefore, in the course of the binary search, the algorithm will reach $T = T^*$ and return no conflicts. Prior to the result on the solution quality, we show the following result on Subroutine II.

Lemma 6.3.5. *There is no conflict in Subroutine II.*

Proof. By contradiction assume there is a conflict in Subroutine II. Therefore, at some step of the execution, there is a vertex v such that $|\mathcal{N}(v)| = 0$. Prior to this step, $|\mathcal{N}(v)| = 1$, and there should have been another vertex w with $|\mathcal{N}(w)| = 1$, otherwise the algorithm would have added the vertex in $\mathcal{N}(v)$ to the solution. Observe that the event of $|\mathcal{N}(w)| = 1$ and $|\mathcal{N}(v)| = 1$ shows that at the start of Subroutine II, $\mathcal{N}(w) \cap \mathcal{N}(v) \neq \emptyset$. Therefore, the location that is added to the solution in $\mathcal{N}(w)$ will service v with $3T$. Thus, v would have been marked as serviced prior to conflict. $\square$

Theorem 6.3.6. *The proposed algorithm is a 3-approximation algorithm for Problem 6.3.3 with global objective $J_{\max}$.*

Proof. By contradiction, assume that the algorithm returns a conflict with $T > T^*$. By Lemma 6.3.5, the conflict can only happen in Subroutine I. Let H^* be the induced graph by removing edges longer that T^* in $\mathcal{E}$. Since $T > T^*$, then $\mathcal{N}_{H^*}(v) \subseteq \mathcal{N}(v)$. Therefore, if there is a conflict in H, there is a conflict in H^*, which is a contradiction. $\square$

6.4 Pay to Control

In this section, we propose our second in-direct control method to relocate the drivers. The idea is that to incentivize a driver to relocate to a desired location, the difference between the driver's expected profit at the waiting location from Equation (6.1) and the expected profit at the desired waiting location needs to be compensated. First, we pose the problem between the drivers and the service provider as a game. Then we provide our algorithms for finding the optimal policy of the service provider.

6.4.1 Service Provider's Game

Let d_i be the incentive per unit distance offered to driver i. Let $Q = \{q_1, \dots, q_m\}$ (resp. $Q' = \{q'_1, \dots, q'_m\}$) be the configuration of the drivers before (resp. after) the incentive pay. The game between the drivers and service provider consists of the following:

(i) A set of m players and a service provider,

(ii) An action set A_i for each driver i, which is the waiting locations in the graph, i.e. $A_i = V\ \forall i \in [m]$. The action set of the service provider is $\mathcal{Q}$; and

(iii) The profit function of the service provider is

$$h(Q') = \sum_{i \in m} d_i \sigma c(q_i, q'_i) + \beta J(Q'),$$

where $J \in \{J_{\text{exp}}, J_{\text{max}}\}$ and $\beta \geq 0$ is a user-defined parameter that indicates the importance of the service quality for the service provider with respect to the incentive pay. For a small value of β, the incentive pay is in the priority, thus the service provider will offer the waiting locations close to the driver's desired waiting location, however, for large values of β, the service provider accepts high incentive pay to relocate the drivers to the configuration with minimum expected response time.

(iv) The profit of driver i is the maximum of the expected profit of the offered waiting location with incentive pay and the expected profit of the waiting location from Equation (6.1), i.e.,

$$\max\{(d_i - 1)\sigma c(q_i, q'_i) + \mathcal{M}_i(q'_i, B_i - c(q_i, q'_i), I_i), \\ \max_{u \in \mathcal{V}} -\sigma c(q_i, u) + \mathcal{M}_i(u, B_i - c(q_i, u), I_i)\}.$$

This is an instance of a leader-follower game [92]. The service provider offers an incentive based on its utility and the drivers as followers either take the offer or reject it. The service provider is aware of the best action of the drivers given any action taken by the service provider (i.e., incentive pay and the offered waiting location). The objective is to find the optimal strategy for the service provider to minimize a linear combination of the incentive pay and the expected response time by relocating the drivers to the desired configuration.

Driver i will accept the offer by the service provider to relocate to q_i' only if the offered incentives surpass the best-expected profit of the driver. Since the profit functions of the drivers are known to the service provider, then the minimum d_i in which the drivers will accept the offer to move to configuration Q' is

$$\begin{aligned} d_i = & \frac{\max_{u\in\mathcal{V}} -\sigma c(q_i,u) + \mathcal{M}_i(u, B - c(q_i,u), I_i)}{\sigma c(q_i,q_i')} \\ & - \frac{\mathcal{M}_i(q_i', B_i - c(q_i,q_i'), I_i)}{\sigma c(q_i,q_i')} + 1. \end{aligned} \tag{6.5}$$

In the equation above, $\max_{u\in\mathcal{V}} -\sigma c(q_i,u) + \mathcal{M}_i(u, B - c(q_i,u), I_i)$ is the maximum expected profit of the driver i by relocating to a new waiting location, and $\mathcal{M}_i(q_i', B_i - c(q_i,q_i'), I_i)$ is the expected profit of driver i by waiting at the location q_i' offered by the service provider. Knowing this minimum d_i, the objective of the service provider becomes

$$\begin{aligned} h(Q') = & \sum_{i\in m} \sigma c(q_i,q_i') + \beta J(Q') \\ & - \sum_{i\in m} \mathcal{M}_i(q_i', B_i - c(q_i,q_i'), I_i) \\ & + \sum_{i\in m} \max_{u\in\mathcal{V}} -\sigma c(q_i,u) + \mathcal{M}_i(u, B - c(q_i,u), I_i). \end{aligned} \tag{6.6}$$

Remark 6.4.1 (Equilibrium). The optimal solution to the problem $\min_{Q'} h(Q')$ is the equilibrium of the leader-follower game between the service provider and the drivers. Since any other configuration will increase the cost function of the service provider. In addition, By Equation (6.5), waiting in a location other than the one suggested by the service provider will decrease driver's expected profit. •

6.4.2 Minimizing The Expected Wait-Time

First, observe that finding the optimal configuration Q' in Equation (6.6) is independent of $\sum_{i\in m} \max_{u\in\mathcal{V}} -\sigma c(q_i,u) + \mathcal{M}_i(u, B - c(q_i,u), I_i)$. Therefore, the problem of minimizing the utility function of the service provider h with $J(Q) = \sum_{u\in\mathcal{V}} \min_{q'\in Q} c(u,q')$, i.e., global objective (6.2), has the mobile facility location (MFL) problem as a special case where $\mathcal{M}_i(v, B_i - c(u,v)) = 0$ for all $u, v \in \mathcal{V}$ and $i \in [m]$. The MFL is a well-known NP-hard problem [31] where given a metric graph $G = (F \cup D, \mathcal{E}, c)$, mapping $\mu : D \to \mathbb{R}_+$ and a subset $Q \subseteq F \cup D$ of size m. The objective is to find a subset $Q' = \{q'_1, \ldots, q'_m\} \subseteq F$ minimizing $\sum_{i\in[m]} c(q_i, q'_i) + \sum_{u\in D} \mu_u \min_{q'\in Q'} c(u,q')$.

We now propose a constant factor approximation for the minimum pay-to-control problem, namely minimizing Equation (6.6). Let $w_{q'_i,I_i} = \frac{1}{\sigma} \max_{u\in\mathcal{V}} \sigma c(q_i,u) - \mathcal{M}_i(u, B - c(q_i,u)) + \mathcal{M}_i(q'_i, B_i - c(q_i,q'_i), I_i)$, then the utility function of the service provider becomes

$$h(Q') = \sum_{i\in[m]} \left(c(q_i,q'_i) - w_{q'_i,I_i}\right) + \beta \sum_{u\in\mathcal{V}} p_a(u) \min_{i\in[m]} c(q'_i,u).$$

The algorithm follows by a reduction from the minimum pay-to-control problem to MFL. Given an instance of the minimum pay-to-control problem we construct an MFL instance as follows:

(i) A graph $G = (Q \cup F \cup \mathcal{V}, \mathcal{E}, c')$ where F is the set of possible waiting locations for the drivers

(ii) There is an edge between $q_i \in Q$ and $q' \in F$ with cost $c'(q_i,q') = \frac{1}{2}\left(c(q_i,q') - w_{q',I_i}\right)$,

(iii) There is an edge between $q' \in F$ and $v \in \mathcal{V}$ with cost $c'(q',v) = c(q',v)$.

(iv) The objective is to find a subset of $Q' \subseteq F$ with $|Q'| = m$ such that minimizes

$$C(Q') = \sum_{i\in m} c'(q_i,q'_i) + \frac{\beta}{2} \sum_{u\in\mathcal{V}} p_a(u) \min_{i\in[m]} c'(q'_i,u).$$

Figure 6.4 shows the constructed MFL instance. Suppose Q' is a solution to the MFL instance, we let Q' be the solution of the minimum pay-to-control problem and provide the following result on the cost of the solution.

Theorem 6.4.2. *Given an α-approximation algorithm for the MFL problem, the reduction above provides an α-approximation for the pay-to-control problem with objective $J_{\exp}$ and any $\beta > 0$.*

Proof. For any $Q' \subseteq F$, by the construction of the MFL instance, we have $h(Q') = 2\sigma C(Q')$. Therefore, given an α-approximation algorithm for the MFL problem, and Q' obtained from

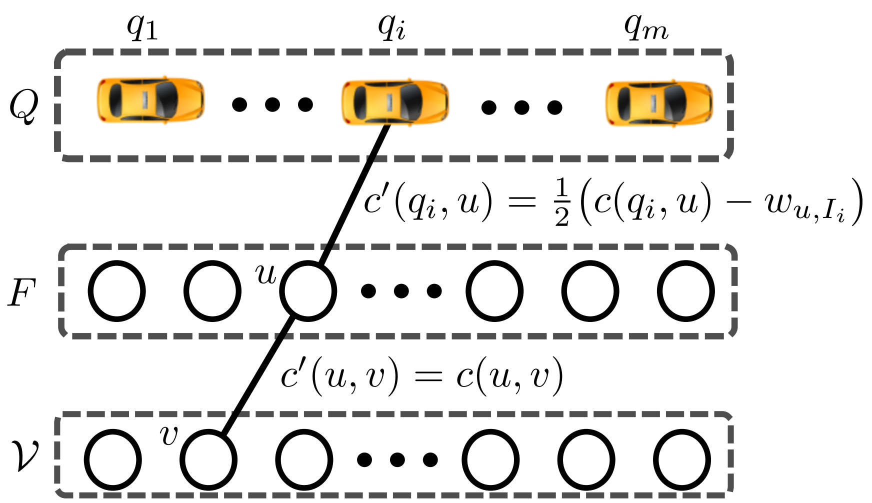


Figure 6.4: Constructed MFL instance for optimizing the utility of the service provider. An instance of the edges between the subsets is shown with their respective costs.

the constructed MFL instance, we select Q' as a solution to the minimum pay-to-control problem. Therefore,

$$h(Q') = 2\sigma C(Q') \le 2\alpha\sigma \min_{Q^* \subseteq F} C(Q^*) = \alpha \min_{Q^* \subseteq F} h(Q^*). \qquad \square$$

By the result of Theorem 6.4.2, the $3 + o(1)$-approximation algorithm for the MFL problem in [31] applies to the minimum pay-to-control problem.

6.4.3 Minimizing The Maximum Wait-Time

In this section, we propose an algorithm for minimizing the service provider's utility function $h(Q)$ in Equation (6.6) where the measure of the service quality is the maximum wait-time, i.e., $J_{\max}(Q)$. Therefore, the utility function of the service provider becomes

$$\begin{aligned} h(Q') = &\sum_{i\in m} \sigma c(q_i, q_i') + \beta \max_{u\in\mathcal{V}} \min_{q'\in Q'} c(u, q') \\ &- \sum_{i\in m} \mathcal{M}_i(q_i', B_i - c(q_i, q_i'), I_i) \\ &+ \sum_{i\in m} \max_{u\in\mathcal{V}} -\sigma c(q_i, u) + \mathcal{M}_i(u, B - c(q_i, u), I_i). \end{aligned} \tag{6.7}$$

Similar to the previous section, observe that the minimization of $h(Q')$ (see Equation (6.6)) is independent of the term $\sum_{i \in m} \max_{u \in \mathcal{V}} -\sigma c(q_i, u) + \mathcal{M}_i(u, B - c(q_i, u), I_i)$. Therefore, the problem of minimizing the service provider's utility function has the metric k-center problem [39] as a special case with $\sigma = 0$ and $\sum_{i \in m} \mathcal{M}_i(q_i', B_i - c(q_i, q_i'), I_i) = 0$. The metric k-center problem is a well-known NP-hard problem.

Now consider the graph $G = (\mathcal{V}, \mathcal{E}, c)$ on the demand vertices. Without the loss of generality, we assume $c(e_1) \le c(e_2) \le \ldots \le c(e_{|\mathcal{V}|^2})$ where $e_i \in \mathcal{E}$ for all $i \in \{1, \ldots, |V|^2\}$. Now consider a set of sub-graphs $G_1, \ldots, G_{|\mathcal{V}|^2}$ on the demand vertices where $G_i = (\mathcal{V}, \mathcal{E}_i, c)$ with $\mathcal{E}_i = \{e \in \mathcal{E} | c(e) \le c(e_i)\}$.

The proposed algorithm for this problem is built on the approximation algorithm for the metric k-center problem in [39]. For each sub-graph G_i, we construct a graph $H_i^2 = (\mathcal{V}, \mathcal{E}_{H^2}^i)$ as follows:

(i) Add the demand vertices to H_i^2,

(ii) add edges in $\mathcal{E}_i$ to $\mathcal{E}_{H^2}^i$, and

(iii) add edge (u, v) to $\mathcal{E}_{H^2}^i$ if there exists $w \in \mathcal{V}$ such that edges (u, w) and (w, v) are in $\mathcal{E}_i$.

For each graph H_i^2, starting from an empty set S_i, we greedily add a vertex in $u \in \mathcal{V} \setminus S_i$ to S_i if there is no edge between u and any vertex in S_i. We continue adding the vertices to S_i until all the vertices in $\mathcal{V} \setminus S_i$ have an edge incident to a vertex in S_i. The set S_i is called a *maximal independent set*. We call a maximal independent set S_i *valid*, if $|S_i| \le m$. Then we construct a bipartite graph for each valid S_i as follows:

(i) Add vertices in Q,

(ii) add vertices in S_i, and

(iii) add an edge between $q_k \in Q$ and $s_j \in S_i$ with cost $\min_{u \in \mathcal{V}_{s_j}} c(q_k, u) - w_{u, I_k}$ where

$$\mathcal{V}_{s_j} = \{u \in \mathcal{V} | c(u, s_j) \le c(e_i)\}.$$

After constructing the graph, we find the optimal assignment in the resulting bipartite graph using the Hungarian algorithm [41]. Let $\mathrm{Assgn}(Q, S_i)$ represent the cost of the optimal assignment for each valid S_i. Let $\mathrm{Assgn}(Q, S') + \beta c(e')$ be the smallest value among the valid independent sets. Let $s_j \in S'$ be the vertex assigned to $q_k \in Q$ is the

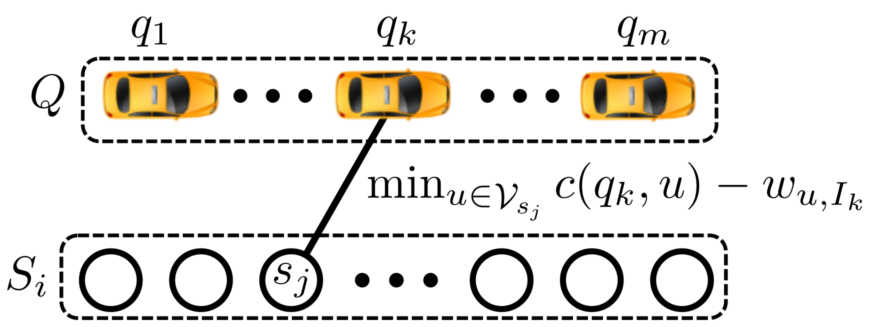


Figure 6.5: Constructed bipartite graph for the problem of minimizing service provider's utility with the global objective of minimizing the maximum wait-time.

solution of the assignment problem. Then we add $q'_j = \arg\min_{u\in\mathcal{V}_{s_j}} c(q_k,u) - w_{u,I_k}$ to the final solution.

Let Q^* be the optimal solution to the problem of minimizing the service provider's utility function, let T^* be the corresponding maximum wait-time in the optimal solution and let Q' be the configuration obtained from the proposed algorithm. Prior to providing the result on the performance of the proposed algorithm, we prove the following result on the steps of the algorithm.

Lemma 6.4.3. *If the maximal independent set S_i is not a valid independent set, i.e. $|S_i| > m$, then $T^* \geq c(e_i)$.*

Proof. Suppose, $T^* < c(e_i)$, then for each vertex in $v \in \mathcal{V}$ there is a vertex in $q \in Q^*$, denoted by $\ell_{Q^*}(v)$, with $c(q,v) \leq T^*$. Observe that for any $s_j, s_k \in S_i$, we have $\ell_{Q^*}(s_j) \neq \ell_{Q^*}(s_k)$, otherwise, there is an edge between s_j, s_k in graph H_i^2. This is a contradiction since S_i is a maximal independent set. Therefore, for each vertex in S_i, there is a unique vertex in Q^*. This is a contradiction, since $|Q^*| = m$. □

Theorem 6.4.4. *The proposed algorithm provides a 3-approximation for the pay-to-control problem with objective $J_{\max}$ and any $\beta > 0$.*

Proof. In the course of the algorithm, we have considered the graph G_i where $c(e_i) = T^*$. By Lemma 6.4.3, we have $|S_i| \leq m$. Also note that for each $s_j \in S_i$, there is a vertex $q_k \in Q^*$ in $\mathcal{V}_{s_j}$. Observe that, while constructing the bipartite graph, we considered the minimum payment cost from each $q \in Q$ to the vertices in $\mathcal{V}_{s_j}$, therefore,

$$\mathrm{Assgn}(Q, Q') \leq \mathrm{Assgn}(Q, Q^*).$$

Since for each $v \in \mathcal{V}$, there is a vertex $s_j \in S_i$ such that $c(s_j, v) \leq 2T^*$, therefore, there is a vertex $q_k \in Q'$ such that $c(q_k, v) \leq c(q_k, s_j) + c(s_j, v) \leq 3T^*$. Finally we have,

$$\begin{aligned} h(Q') &= \mathrm{Assgn}(Q, Q') + \beta J_{\max}(Q') \leq \mathrm{Assgn}(Q, Q') + 3\beta T^* \\ &\leq 3\mathrm{Assgn}(Q, Q^*) + 3\beta T^* = 3h(Q^*). \end{aligned} \qquad \square$$

6.5 Simulation Results

In this section, we evaluate the performance of the two proposed control methods on real-world ride-sharing data for yellow taxis in Manhattan, N.Y. [93]. In our experiments, we consider the ride requests for a randomly chosen day 10/06/2016, since the data-set for the year 2016 is the latest available data-set that provides the coordinates of pick-up and drop-off locations.

To reduce the complexity of the large data set with 401464 pick-up locations, we clustered the ride requests using K-means algorithm [94] into 500 clusters. Figure 6.6 shows the clustered pick-up locations and the arrival rates at each cluster is represented with a bar. The drop-off probability $p_d(w|v)$ of a ride request with the source at cluster v and destination w is obtained from the average number of ride requests assigned to cluster v with destination closest to the center of cluster w.

We consider two global objectives: 1) the expected wait-time, and 2) the maximum wait-time and two scenarios: 1) random initial configuration where the initial location of the drivers are selected uniformly randomly, and 2) jammed initial configuration where the drivers are initialized at the 20 closest locations to the Rockefeller center.

Observe that the proposed algorithms to find the controls are applicable to various driver models for $\mathcal{M}$. In the following section, we provide the driver model $\mathcal{M}$ used in the simulations.

6.5.1 Drivers' model

A driver model is a function of evaluating the expected profit of different locations at each time instance. Prior to introducing our driver model, we provide the parameters in the model.

The *pick-up probability* of a location v as seen by driver i is the probability that a ride request at v is assigned to driver i before any other ride request. Consider a configuration

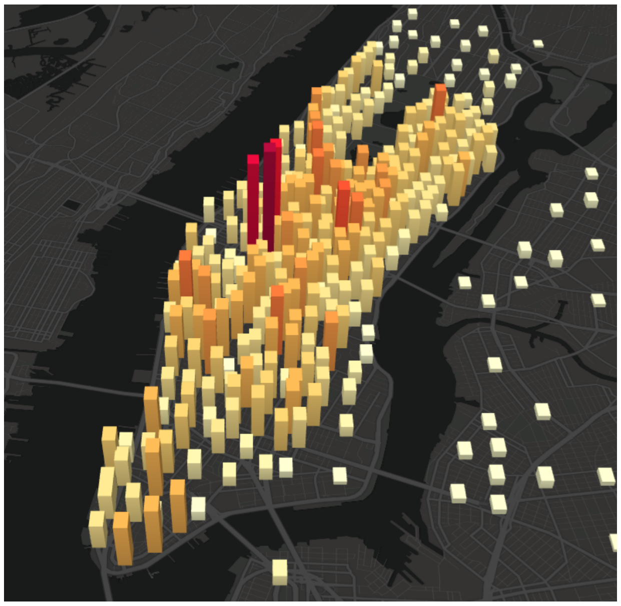


Figure 6.6: The set of pick-up and drop off locations in Manhattan. The bars at the locations of clusters represent the ride request arrival rates.

Q of the drivers and a vertex u, where driver i is the kth closest driver to the vertex location u. Since a request is assigned to the closest available driver, then driver i can expect to be assigned to the kth request arriving at u. Let $S_i(v,u)$ denote the number of drivers closer to v than driver i located at vertex u. Note that $S_i(v,u)$ is a function of the information of driver i regarding the position of other drivers. Figure 6.7 illustrates an instance with three vehicles and a pick-up location. Given the full information of the vehicle positions, S_{red} at pick-up location is 2 and S_{red} is zero if $I_{\text{red}} = \emptyset$.

We assume that the ride requests arrive at each location v according to a Poisson process with arrival rate λ_v. Then, the probability that a request at v is assigned to driver i at u as expected by driver i before other requests is approximated by

$$p_i(v,u) = \Pi_{w\in\mathcal{V}} \sum_{k=S_i(v,u)+1}^{S_i(v,u)+S_i(w,u)+1} \binom{\sum_{w\in\mathcal{V}} S_i(w,u)+1}{k}$$
$$\Big(\frac{\lambda_v}{\lambda_v+\lambda_w}\Big)^k \Big(\frac{\lambda_w}{\lambda_v+\lambda_w}\Big)^{S_i(v,u)+S_i(w,u)+1-k}.$$

Let σ' be the fare per unit time of servicing a request, then the profit of a ride with pick-off at v and drop-off at w for driver i located at u is $\sigma' c(w,v) - \sigma c(u,v)$.

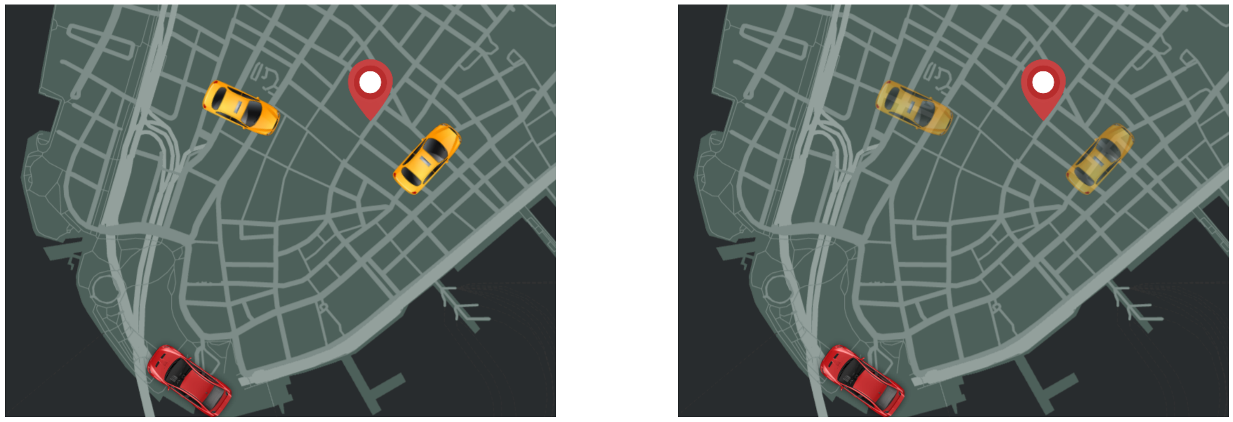

(a) $I_{\text{red}} = Q$ (b) $I_{\text{red}} = \emptyset$

Figure 6.7: Environment with three drivers and a pick-up location v. (a) The red car has full information on the position of the other drivers, therefore, $S_{\text{red}}(v, q_{\text{red}}) = 2$, (b) The positions of the other drivers are not known to the driver of red car, thus, $S_{\text{red}}(v, q_{\text{red}}) = 0$.

Now we define the expected profit of driver i located at u and working for B_i period of time as follows:

$$\begin{aligned}\mathcal{M}_i(u,B_i,I_i) = \sum_{v,w\in\mathcal{V}} p_d(w|v)\big[p_i(v,u)\big(\max\{0,\sigma' c(w,v)\\ - \sigma c(u,v) + \mathcal{M}_i(w, B_i - c(u,v) - c(v,w), I_i)\}\big)\big],\end{aligned} \tag{6.8}$$

where $\mathcal{M}_i(u, 0, I_i) = 0$ for all $u \in \mathcal{V}$, drivers $i \in [m]$ and $I_i \subseteq Q$. Note that, $\mathcal{M}_i(w, B_i - c(u,v) - c(v,w), I_i)$ represents the expected profit of driver i after drop-off at w. The conditional probability of the drop-off location is the common knowledge of the drivers and the ride-sharing companies based on prior customer data.

Intuitively, the expected profit in Equation (6.8) represents the total expected profit of servicing requests with pick-up location at v and drop-offs at w only if the

$$\sigma' c(w,v) - \sigma c(u,v) + \mathcal{M}_i(w, B_i - c(u,v) - c(v,w), I_i) \geq 0.$$

Observe that finding $\mathcal{M}_i(u, B_i, I_i)$ for all $u \in \mathcal{V}$ and a given B_i is performed in polynomial time, however, since the drivers are not going through the calculation of $\mathcal{M}_i(u, B_i, I_i)$ for all $u \in \mathcal{V}$, we assume that they have access to the expected profit of the vertices by experience.

Since the calculation of the expected profit for drivers for each time step is computationally expensive, we trained a Random Forest Regressor [95] implemented by [94] to approximate the values of $\mathcal{M}$ for each number of vehicles in the system with training data over 10000 instances with work-day B_i of 15 average length rides, fare $\sigma' = \$0.81$ per kilometer [86] and driving cost $\sigma = \$0.15$ per kilometer [96].

6.5.2 Information Sharing

We begin by evaluating the performance of the information sharing control method. Observe that the number of unique locations in the set $\cup_i^m F_i$ (see Section 6.3) improves the possibility of providing better solutions for both global objectives. Also, note that the drivers that are located at the same location given the same information have the same candidate waiting location. Therefore, for all but one of these drivers, we replace the set $F_i = \{q'_{i,Q}, q'_{i,\emptyset}\}$ with the set $F_i = \{q'_{i,R}, q'_{i,\emptyset}\}$ where R is a random subset of the full information Q.

Figure 6.8 shows the percentage improvement in the expected wait time for different number of vehicles using the partial information sharing control method of Section 6.3.2 in the jammed and random initial configuration scenarios. The results are the average of 200 instances for a varying number of drivers in each scenario. The boxes show the first, second and third quartiles of each set of experiments. Observe that the information sharing algorithm improves the expected wait time of the ride requests by approximately 20% in jammed initial configuration scenario. Also observe that the improvement in the expected wait-time with randomly distributed drivers in the environment is minimal, since the randomly distributed drivers provide a close to optimal expected-wait time, especially in an environment where the arrival rates are not heavily concentrated in a certain area. Therefore, the possibility to improve the expected-wait time with the information sharing algorithm is limited. The expected wait time of the solution obtained from the LP-rounding algorithm of Section 6.3 on this set of experiments is on average within 0.021% of optimal. The maximum deviation from the optimal solution is 0.58%. We obtain these bounds by comparing the solution of the LP-rounding algorithm to that from the LP relaxation, which provides an upper bound on the error from optimal.

Figure 6.9 illustrates the percentage improvement in the maximum wait-time of the ride requests using the partial information sharing control method of Section 6.3.3. Note that the proposed algorithm improves the maximum wait-time of the ride requests by approximately 25% in the jammed scenario and 5% in the random initial configuration scenario.

Figure 6.10 shows the expected response time of a set of 80 drivers responding to 300 requests arriving over time with the jammed initial configuration. The figure summarizes

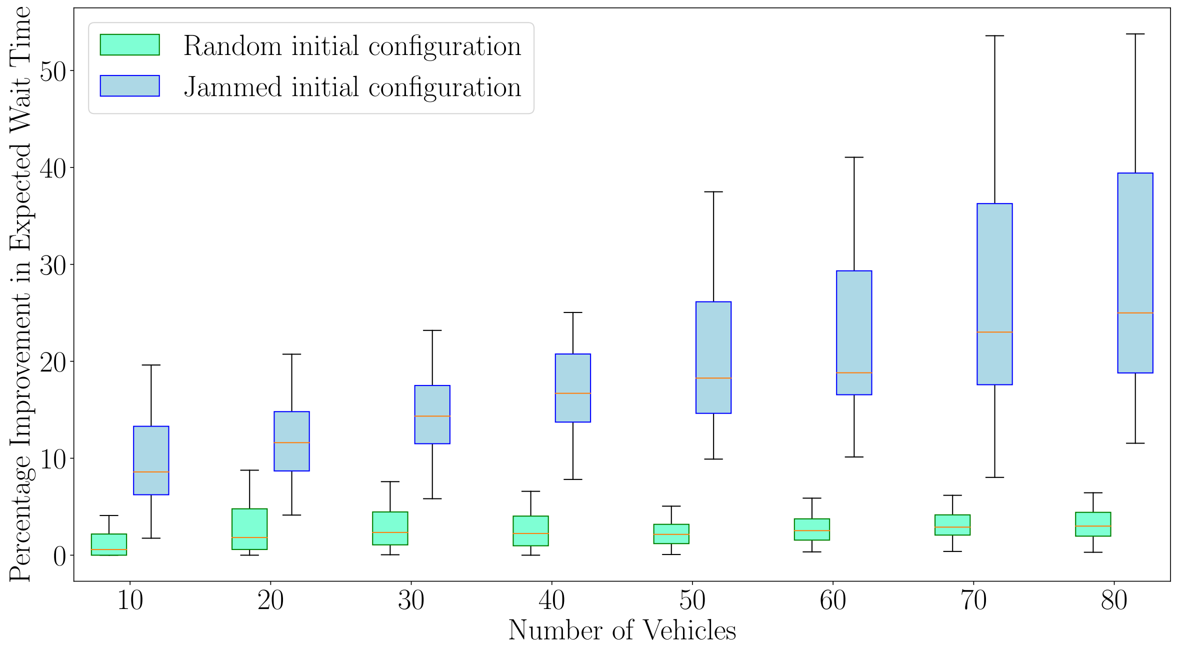


Figure 6.8: Improvement in J_{exp} by sharing partial information

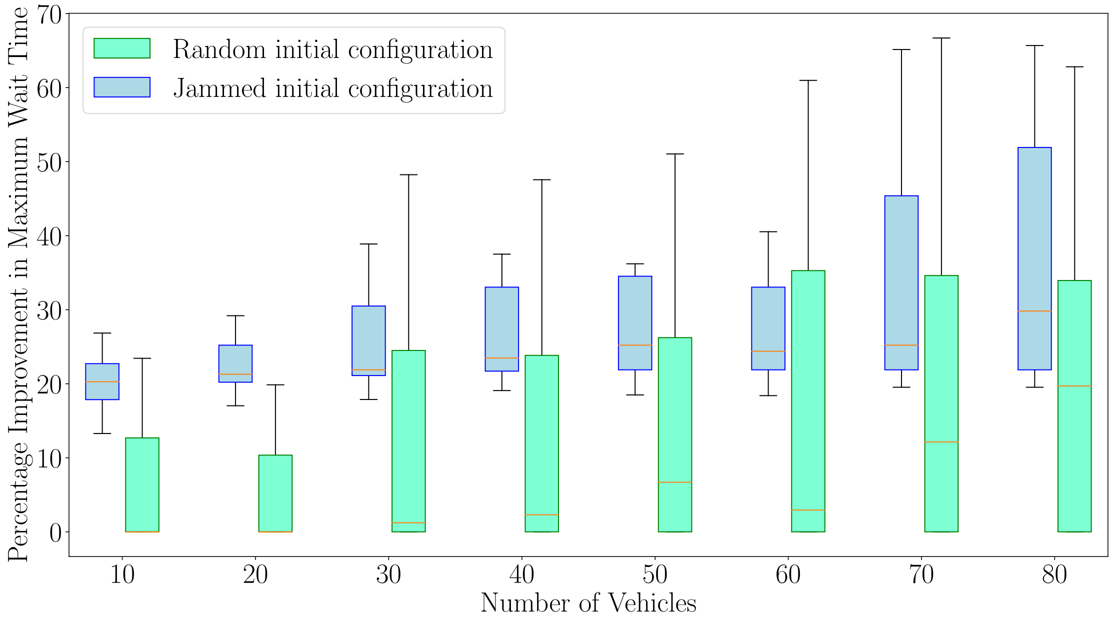


Figure 6.9: Improvement in $J_{\max}$ by sharing partial information

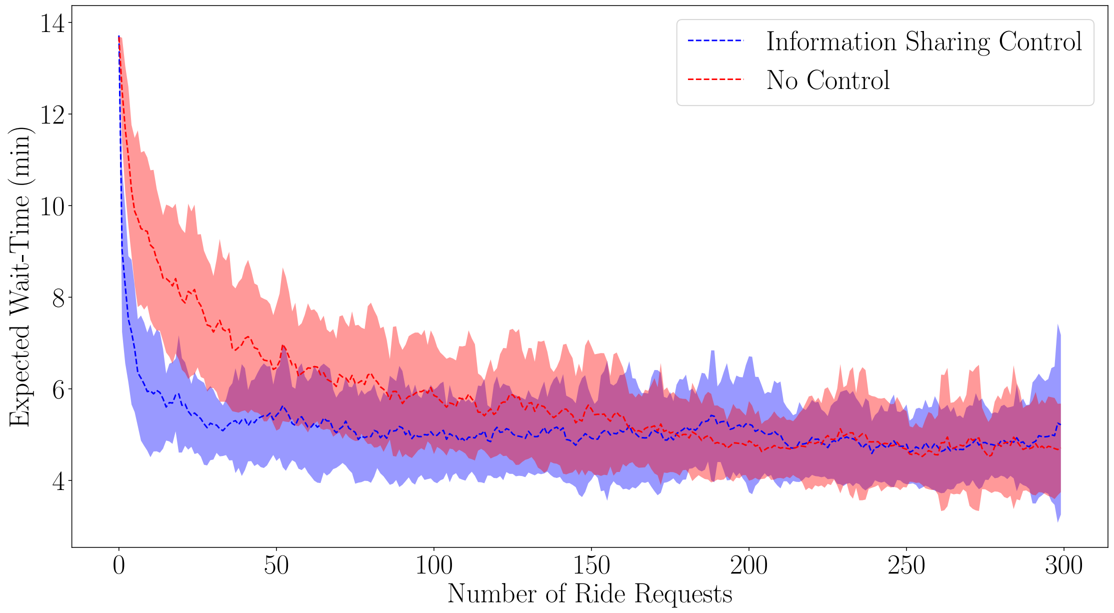


Figure 6.10: The expected wait-time of ride requests in a system of 80 drivers executing 300 ride requests arriving over time under the partial information control method.

100 experiments with 300 randomly generated requests for each experiment. The lines represent the average and shaded areas represent the first and third quartiles. In the transient, the information sharing algorithm improves the expected response time rapidly. However, in the steady-state, the performance is very similar to the no control case. Thus, the information sharing method serves to more quickly disperse the drivers from the initial jammed configuration. The information-sharing algorithm is applied whenever the expected wait time is larger than 6 minutes. Figure 6.11 shows a similar experiment with the objective of minimizing the maximum wait-time. Observe that the proposed algorithm improves the maximum wait-time in the initial jammed configuration and maintains the quality service over the course of responding to 300 requests. To limit the number of information sharing control inputs to the system, we only use the information sharing method when the maximum wait time in the current driver configuration is larger than 30 minutes.

6.5.3 Pay to Control

In this section, we evaluate the performance of the pay-to-control method for the jammed and random initial configuration scenarios. Figure 6.12a illustrates the improvement in

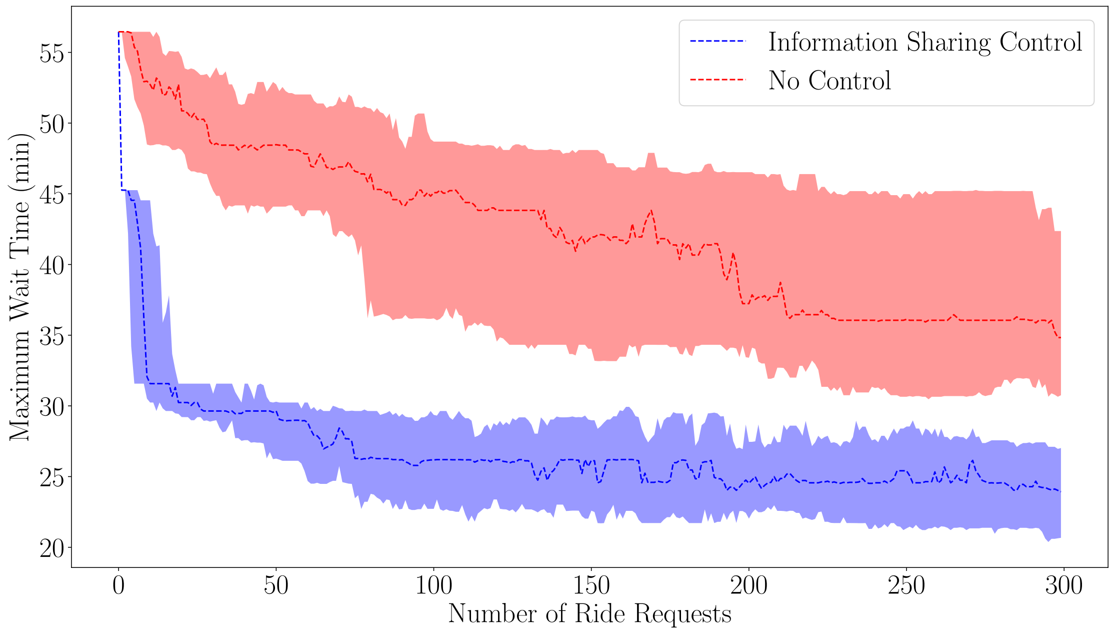
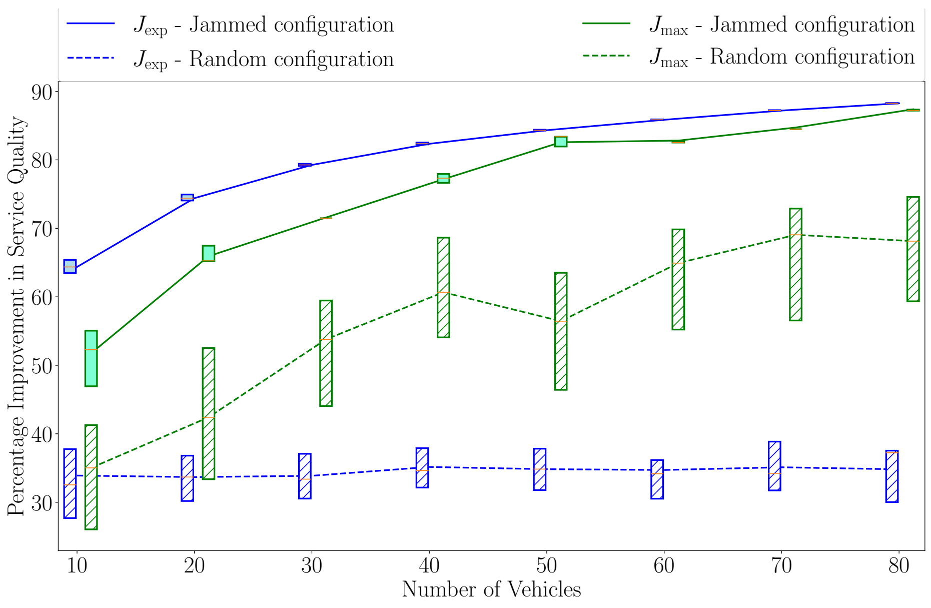


Figure 6.11: The maximum wait-time of ride requests in a system of 80 drivers executing 300 ride requests arriving over time under the partial information control method.

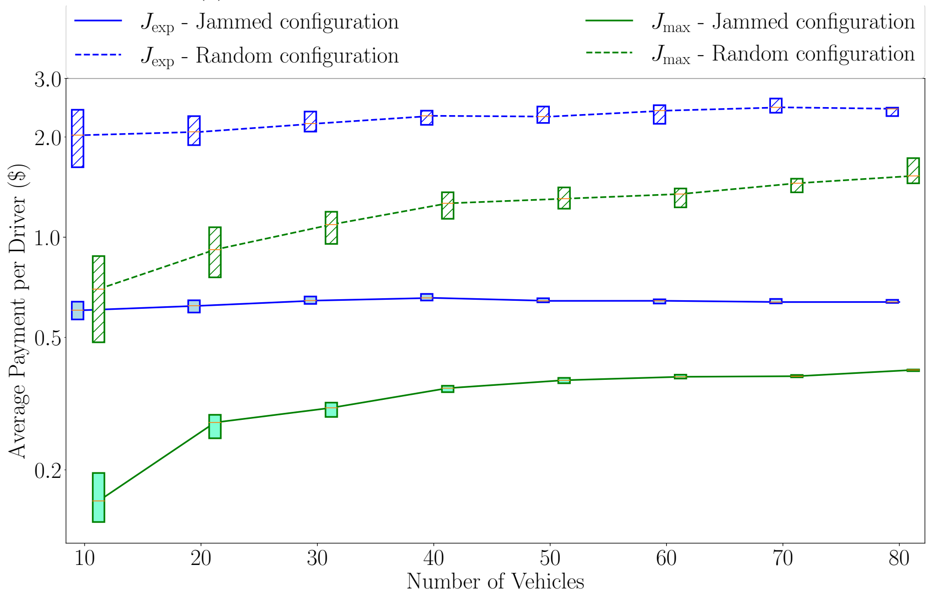
the expected wait-time and the maximum wait-time for the two scenarios with $\beta = 100$. Note that the proposed algorithm improves the expected wait-time by approximately 75% for the jammed scenario and 35% for the random scenario. Also observe that the proposed algorithm improves the maximum wait-time by approximately 70% in the jammed scenario and 50% in the random scenario.

Figure 6.12b shows the payment per driver for the two objectives and the two scenarios. Observe that the expected profit of the drivers in the jammed scenario is smaller than the expected profit of the drivers in the random scenario, therefore, the amount paid to convince the drivers to relocate to desired waiting locations in the jammed scenario is significantly smaller compared to the amount paid to the drivers in the random scenario.

Figure 6.13 shows the expected response time of a set of 80 drivers responding to 300 requests arriving over time with the jammed initial configuration. The results are an average of 100 experiments with 300 randomly generated requests for each experiment. The lines represent the average and shaded areas represent the first and third quartiles. The pay-to-control is applied to the system if the expected wait time is greater than 3 minutes. Observe that the proposed algorithm maintains the low expected wait-time in the course of servicing 300 ride requests. The average pay to maintain the low expected wait-time



(a) The percentage improvement in the service quality



(b) The amount paid to drivers

Figure 6.12: Improvement in the global objectives and the amount paid to drivers using pay-to-control method. The blue represents the results for the objective J_{exp} and the green shows the results for the objective J_{max}. The solid bars represent the jammed initial configuration and the hatched bars represent the random initial configuration.

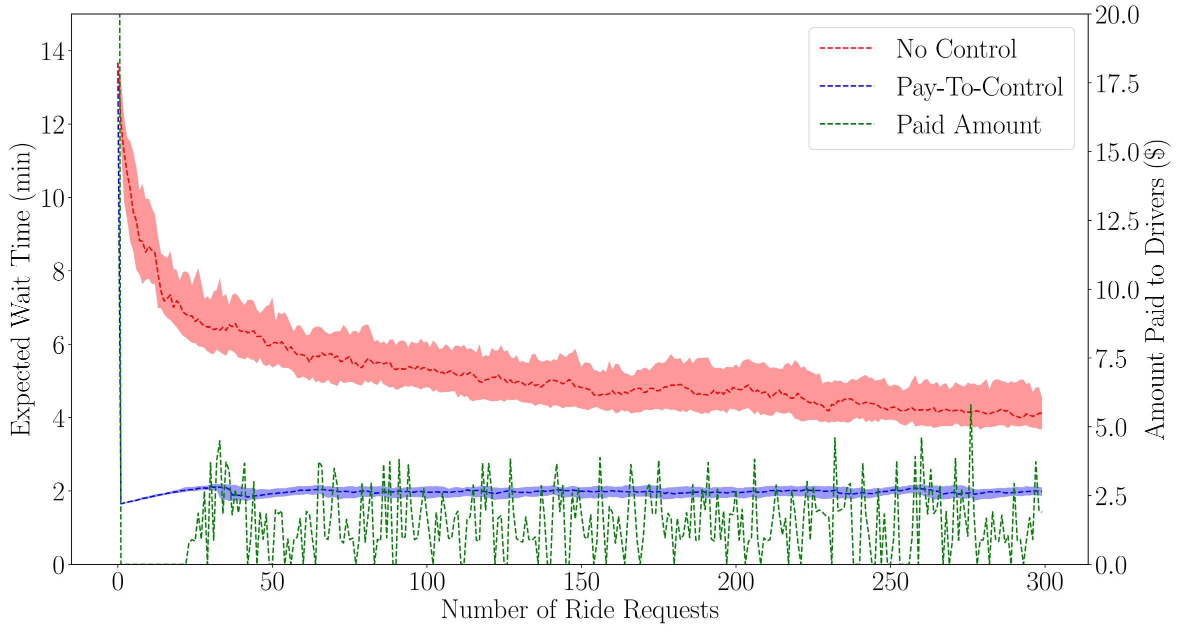


Figure 6.13: The expected wait time and the paid amount to a system of 80 drivers executing 300 ride requests arriving over time under the pay-to-control method

is 1.95$ per ride request. Figure 6.14 shows the same experiment with the objective of minimizing the maximum wait-time. The pay-to-control is applied to the system if the maximum wait time is greater than 7 minutes. The average pay to maintain the low expected wait-time is 1.58$ per ride.

In summary, the experiments on the real-world ride-sharing data show that the proposed algorithms significantly improve the expected or the maximum wait-time when the drivers are concentrated in a region. In particular, given a jammed configuration of the drivers, the proposed algorithms improve the service quality significantly with a small number of control inputs and maintain the same quality over-time. The improvements are more evident for the social objective of minimizing the maximum wait-time, since the natural dynamics of the system tend to steer drivers away from the low-demand areas. Therefore, the control from the proposed algorithm is required to maintain equal service quality across different regions.

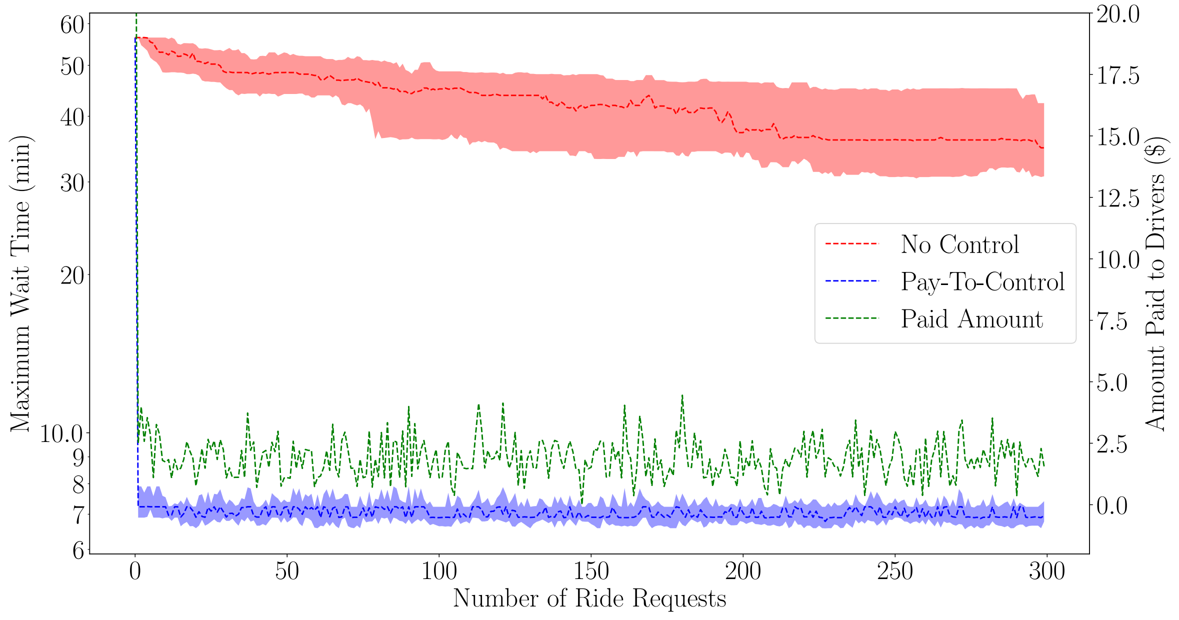


Figure 6.14: The maximum wait time and the paid amount to a system of 80 drivers executing 300 ride requests arriving over time under the pay-to-control method

Chapter 7

Conclusions and Future Work

In this thesis, we studied four problems on the deployment of robots. The first problem considered the multi-robot coverage problem in convex and non-convex environments. A connection is established between the solution quality of the continuous coverage problem and the solution to the coverage problem on a discrete representation of the environment. We also proposed the first distributed approximation algorithm for the coverage problem in discrete and continuous environments and provide bound on the quality of the solution. Finally, we characterized the run-time and communication complexity of the proposed algorithm. The simulations on convex and non-convex environments showed a significant improvement in the sensing cost compared to the state-of-the-art algorithms.

For the second problem, we considered the problem of coverage control of multiple robots with heterogeneous sensing capabilities in continuous and discrete environments. A new formulation is introduced for measuring the coverage of multiple event types with different event distributions, and a distributed control law is presented for maximizing the coverage of the robots in the environment, which only requires communication between neighboring robots. Then we extended the distributed control law for discrete environments with different event types. Then we evaluated the performance of the proposed algorithm in a continuous convex environment in which the results show significant improvement in the sensing cost of the events compared to the state-of-the-art method. Finally, we employed the proposed control law in discrete environments for ride-sharing applications and presented the results on improving the expected wait-times using the proposed algorithm in a ride-sharing network with different ride sizes.

Next, for the third problem, we considered the problem of deploying a set of robots to efficiently service tasks that arrive sequentially in an environment over time. The presented policies provide a close to the optimal solution and show significant improvement in the run-time compared to dynamic programming approaches. Also, various experiments show the robust behavior of the policies in practice. In addition, the results for the experiment with unknown distribution and high arrival rate provides a direction for analyzing the performance of the policies in these scenarios.

The last problem we studied the problem of controlling self-interested drivers in ride-sharing applications. Two indirect control methods were proposed and for each, a near-optimal algorithm was presented. For the objective of minimizing the expected wait-time with the first control method, we proposed an LP-rounding algorithm and for minimizing the maximum wait-time with the same method we propose a 3 approximation algorithm. For the objectives of minimizing the maximum and expected wait-time with the second control method, we proposed two 3-approximation algorithms. The extensive results show significant improvement in the expected wait-time and the maximum wait-time on real-world ride-sharing data.

7.1 Future Work

In this section, we will discuss the future work directions for the problems studied in this thesis.

7.1.1 Distributed Coverage Control of Multiple Robots

In Chapter 3, we discussed the problem of multi-robot coverage with homogeneous robots in convex and non-convex environments. In the coverage problem, we assumed that the sensors have unlimited range and an event is sensed by the closest robot. However, in real applications, the sensors are only capable of sensing events within a certain range. A future direction for the multi-robot coverage problem with homogeneous robots is to provide a distributed algorithm with guarantees on the solution quality when the sensors have a limited range. Another natural extension to the problem studied in Chapter 3 is the coverage with heterogeneous robots where the robots have different sensor foot-prints.

In Chapter 4, we provided a new formulation for the multi-robot coverage with multiple event types. The approach proposed in this chapter is a descent algorithm so that the robots converge to a locally optimal position. However, unlike the results in Chapter 3 for the multi-robot coverage with homogeneous robots, we did not provide guarantees on the solution quality of the proposed algorithm. A possible future direction for the multi-robot coverage problem with multiple event types is to extend the algorithm in Chapter 3 to capture multiple event types while preserving the approximation factor properties.

7.1.2 Re-balancing Self-interested Drivers in Transportation Network

In Chapter 6, we discussed the problem of re-balancing self-interested drivers in transportation networks to improve customer wait-time. The problem studied in this section captured a microscopic aspect of ride-sharing where we focused on the movement of individual drivers. Although this approach provides the opportunity to devise algorithms with guarantees on their solution quality, they might suffer in capturing environments with large number of drivers. A possible direction future direction is to provide a flow formulation to approximate the average number of drivers relocating and adapt the proposed two control methods in Chapter 6 to control the relocation of the drivers. Such a formulation is given as follows: Consider a graph $G = (\mathcal{V}, \mathcal{E}, c)$ where $\mathcal{V}$ represents the stations in the network, $\mathcal{E}$ represents the road network and c is a function assigning travel times to the roads in the network. Let m_i be the number of drivers at station i, let $\mathcal{M}(j, \lambda_j, m_j)$ be the expected profit of driver for station j with arrival rate λ_j, given the information that there are m_j drivers at station j. Similarly, let $\mathcal{M}(j, \lambda_j, \emptyset)$ be the expected profit of driver for station j without providing information on the number of drivers at station j. Then the minimum payment required to convince a driver at i to relocate to station j is

$$\begin{aligned} p_{ij} &= c(i,j) - \max\{\mathcal{M}(j, \lambda_j, m_j), \mathcal{M}(j, \lambda_j, \emptyset)\} \\ &, \max_k -c(i,k) + \min\{\mathcal{M}(k, \lambda_k, m_k), \mathcal{M}(k, \lambda_k, \emptyset)\}. \end{aligned} \tag{7.1}$$

Let J be some measure of service quality and let x^r_{ij} be the number of vehicles relocating from i to j with payments by the service provider. Therefore, the problem of re-balancing drivers with minimum payment such that there is no shortage of drivers in the stations can be formulated as follows:

$$\underset{x^r_{ij}}{\text{maximize}} \quad J - \sum_{i,j \in \mathcal{V}} p_{ij} x^r_{ij} \tag{7.2a}$$

$$\text{subject to} \quad \sum_{j \in \mathcal{V}} x^r_{ij} - \sum_{j \in \mathcal{V}} x^r_{ji} = -\sum_{j \in \mathcal{V}} \lambda_{ij} + \sum_{j \in \mathcal{V}} \lambda_{ji} \ \forall i \in \mathcal{V}, \tag{7.2b}$$

$$x_{ij} \in \{0,1\} \ \forall i, j \in V \tag{7.2c}$$

The Constraint 7.2b ensures that the supply of the drivers and the demand at each station is balanced. Unlike the proposed algorithm in Chapter 6, the flow formulation can handle ride-sharing networks systems with a large number of drivers.

References

[1] S.-k. Yun and D. Rus, “Distributed coverage with mobile robots on a graph: locational optimization and equal-mass partitioning,” *Robotica*, vol. 32, no. 2, pp. 257–277, 2014.

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

[3] A. B. Asghar and S. L. Smith, “Stochastic patrolling in adversarial settings,” in *IEEE American Control Conference, 2016*, 2016, pp. 6435–6440.

[4] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, “Exposure in wireless ad-hoc sensor networks,” in *Proceedings of the 7th annual international conference on Mobile computing and networking.* ACM, 2001, pp. 139–150.

[5] T. B. Curtin, J. G. Bellingham, J. Catipovic, and D. Webb, “Autonomous oceanographic sampling networks,” *Oceanography*, vol. 6, no. 3, pp. 86–94, 1993.

[6] K. Niechwiadowicz and Z. Khan, “Robot based logistics system for hospitals-survey,” in *IDT Workshop on interesting results in computer science and engineering*, 2008.

[7] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, “On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment,” *Proceedings of the National Academy of Sciences*, p. 201611675, 2017.

[8] J. Miller and J. P. How, “Demand estimation and chance-constrained fleet management for ride hailing,” *arXiv preprint arXiv:1703.02130*, 2017.

[9] J. T. Isaacs, D. J. Klein, and J. P. Hespanha, “Algorithms for the traveling salesman problem with neighborhoods involving a dubins vehicle,” in *IEEE American Control Conference*, 2011, pp. 1704–1709.

[10] D. Bhadauria and V. Isler, "Data gathering tours for mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.* IEEE, 2009, pp. 3868–3873.

[11] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus, "Robotic load balancing for mobility-on-demand systems," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, 2012.

[12] ——, "Load balancing for mobility-on-demand systems," in *Robotics: Science and Systems.* Los Angeles, CA, 2011.

[13] M. L. Balinski, "On finding integer solutions to linear programs," Mathematica Princeton NJ, Tech. Rep., 1964.

[14] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482–1504, 2011.

[15] A. Renzaglia, L. Doitsidis, A. Martinelli, and E. B. Kosmatopoulos, "Adaptive-based distributed cooperative multi-robot coverage," in *Proceedings of the 2011 American Control Conference.* IEEE, 2011, pp. 468–473.

[16] M. M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal, "Cobots: Robust symbiotic autonomous mobile service robots." in *International Joint Conferences on Artificial Intelligence*, 2015, p. 4423.

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

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

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

[20] Y. Kantaros, M. Thanou, and A. Tzes, "Distributed coverage control for concave areas by a heterogeneous robot–swarm with visibility sensing constraints," *Automatica*, vol. 53, pp. 195–207, 2015.

[21] O. Arslan and D. E. Koditschek, “Voronoi-based coverage control of heterogeneous disk-shaped robots,” in *IEEE International Conference on Robotics and Automation*, 2016, pp. 4259–4266.

[22] A. A. Kuehn and M. J. Hamburger, “A heuristic program for locating warehouses,” *Management science*, vol. 9, no. 4, pp. 643–666, 1963.

[23] D. B. Shmoys, É. Tardos, and K. Aardal, “Approximation algorithms for facility location problems,” in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of computing*. ACM, 1997, pp. 265–274.

[24] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, “Analysis of a local search heuristic for facility location problems,” *Journal of algorithms*, vol. 37, no. 1, pp. 146–188, 2000.

[25] K. Jain and V. V. Vazirani, “Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation,” *Journal of the ACM (JACM)*, vol. 48, no. 2, pp. 274–296, 2001.

[26] S. Guha and S. Khuller, “Greedy strikes back: Improved facility location algorithms,” *Journal of Algorithms*, vol. 31, no. 1, pp. 228–248, 1999.

[27] F. A. Chudak and D. B. Shmoys, “Improved approximation algorithms for the uncapacitated facility location problem,” *SIAM Journal on Computing*, vol. 33, no. 1, pp. 1–25, 2003.

[28] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, “Local search heuristics for k-median and facility location problems,” *SIAM Journal on computing*, vol. 33, no. 3, pp. 544–562, 2004.

[29] E. D. Demaine, M. Hajiaghayi, H. Mahini, A. S. Sayedi-Roshkhar, S. Oveisgharan, and M. Zadimoghaddam, “Minimizing movement,” *ACM Transactions on Algorithms (TALG)*, vol. 5, no. 3, p. 30, 2009.

[30] Z. Friggstad and M. R. Salavatipour, “Minimizing movement in mobile facility location problems,” *ACM Transactions on Algorithms (TALG)*, vol. 7, no. 3, p. 28, 2011.

[31] S. Ahmadian, Z. Friggstad, and C. Swamy, “Local-search based approximation algorithms for mobile facility location problems,” in *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2013, pp. 1607–1621.

[32] R. Halper, S. Raghavan, and M. Sahin, "Local search heuristics for the mobile facility location problem," *Computers & Operations Research*, vol. 62, pp. 210–223, 2015.

[33] M. Tsao, R. Iglesias, and M. Pavone, "Stochastic model predictive control for autonomous mobility on demand," *arXiv preprint arXiv:1804.11074*, 2018.

[34] G. C. Calafiore, C. Novara, F. Portigliotti, and A. Rizzo, "A flow optimization approach for the rebalancing of mobility on demand systems," in *IEEE International Conference on Decision and Control*, 2017, pp. 5684–5689.

[35] G. P. Cachon, K. M. Daniels, and R. Lobel, "The role of surge pricing on a service platform with self-scheduling capacity," *Manufacturing and Service Operations Management*, vol. 19, no. 3, pp. 368–384, 2017.

[36] Q. Wei, J. A. Rodriguez, R. Pedarsani, and S. Coogan, "Ride-sharing networks with mixed autonomy," in *American Control Conference*, 2019, pp. 3303–3308.

[37] S. Banerjee, R. Johari, and C. Riquelme, "Pricing in ride-sharing platforms: A queueing-theoretic approach," in *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, 2015, pp. 639–639.

[38] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms.* MIT press, 2009.

[39] V. V. Vazirani, *Approximation algorithms.* Springer Science & Business Media, 2013.

[40] R. Diestel, "Graph theory. 2005," *Grad. Texts in Math*, vol. 101, 2005.

[41] R. Jonker and T. Volgenant, "Improving the hungarian assignment algorithm," *Operations Research Letters*, vol. 5, no. 4, pp. 171–175, 1986.

[42] S. Li, "A 1.488 approximation algorithm for the uncapacitated facility location problem," *Information and Computation*, vol. 222, pp. 45–58, 2013.

[43] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete partitioning and coverage control for gossiping robots," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 364–378, 2012.

[44] M. Santos, Y. Diaz-Mercado, and M. Egerstedt, "Coverage control for multirobot teams with heterogeneous sensing capabilities," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 919–925, 2018.

[45] A. Sadeghi and S. L. Smith, “Coverage control for multiple event types with heterogeneous robots,” in *IEEE International Conference on Robotics and Automation*, 2019, pp. 3377–3383.

[46] ——, “On re-balancing self-interested agents in ride-sourcing transportation networks,” in *IEEE Conference on Decision and Control.* 2018, 2019, pp. 5119–5125.

[47] A. Sadeghi, A. B. Asghar, and S. L. Smith, “Approximation algorithms for distributed multi-robot coverage in non-convex environments,” *arXiv preprint arXiv:2005.02471*, 2020.

[48] C. H. Caicedo-Nunez and M. Zefran, “Performing coverage on nonconvex domains,” in *2008 IEEE International Conference on Control Applications.* IEEE, 2008, pp. 1019–1024.

[49] C. H. Caicedo-Núnez and M. Zefran, “A coverage algorithm for a class of non-convex regions,” in *2008 47th IEEE Conference on Decision and Control.* IEEE, 2008, pp. 4244–4249.

[50] M. Thanou, Y. Stergiopoulos, and A. Tzes, “Distributed coverage using geodesic metric for non-convex environments,” in *2013 IEEE international conference on robotics and automation.* IEEE, 2013, pp. 933–938.

[51] Y. Kantaros, M. Thanou, and A. Tzes, “Visibility-oriented coverage control of mobile robotic networks on non-convex regions,” in *IEEE International Conference on Robotics and Automation*, 2014, pp. 1126–1131.

[52] H. Mahboubi, F. Sharifi, A. G. Aghdam, and Y. Zhang, “Distributed coordination of multi-agent systems for coverage problem in presence of obstacles,” in *IEEE American Control Conference*, 2012, pp. 5252–5257.

[53] R. J. Alitappeh, G. A. Pereira, A. R. Araújo, and L. C. Pimenta, “Multi-robot deployment using topological maps,” *Journal of Intelligent & Robotic Systems*, vol. 86, no. 3-4, pp. 641–661, 2017.

[54] S. Bhattacharya, N. Michael, and V. Kumar, “Distributed coverage and exploration in unknown non-convex environments,” in *Distributed autonomous robotic systems.* Springer, 2013, pp. 61–75.

[55] D. B. Shmoys, “Approximation algorithms for facility location problems,” in *International Workshop on Approximation Algorithms for Combinatorial Optimization.* Springer, 2000, pp. 27–32.

[56] S. Li and O. Svensson, "Approximating k-median via pseudo-approximation," *SIAM Journal on Computing*, vol. 45, no. 2, pp. 530–547, 2016.

[57] A. Sadeghi and S. L. Smith, "Re-deployment algorithms for multiple service robots to optimize task response," in *IEEE International Conference on Robotics and Automation*, 2018.

[58] M.-F. F. Balcan, S. Ehrlich, and Y. Liang, "Distributed k-means and k-median clustering on general topologies," in *Advances in Neural Information Processing Systems*, 2013, pp. 1995–2003.

[59] S. M. LaValle, *Planning algorithms.* Cambridge university press, 2006.

[60] D. J. Bertsimas and G. Van Ryzin, "A stochastic and dynamic vehicle routing problem in the euclidean plane," *Operations Research*, vol. 39, no. 4, pp. 601–615, 1991.

[61] T. Lemaire, R. Alami, and S. Lacroix, "A distributed tasks allocation scheme in multi-uav context," in *IEEE International Conference on Robotics and Automation*, vol. 4, 2004, pp. 3622–3627.

[62] J. Miller and J. P. How, "Predictive positioning and quality of service ridesharing for campus mobility on demand systems," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1402–1408.

[63] A. Sadeghi and S. L. Smith, "Heterogeneous task allocation and sequencing via decentralized large neighborhood search," *Unmanned Systems*, vol. 5, no. 02, pp. 79–95, 2017.

[64] X. Wang, S. Han, Y. Wu, and X. Wang, "Coverage and energy consumption control in mobile heterogeneous wireless sensor networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 975–988, 2013.

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

[66] J. Le Ny and G. J. Pappas, "Adaptive deployment of mobile robotic networks," *IEEE Transactions on automatic control*, vol. 58, no. 3, pp. 654–666, 2013.

[67] P. Kundu and L. Cohen, "Fluid mechanics, 638 pp," *Academic, Calif*, 1990.

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

[69] D. Pickem, M. Lee, and M. Egerstedt, "The Gritsbot in its natural habitat – a multi-robot testbed," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 4062–4067.

[70] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1699–1706.

[71] M. de Roo, P. Frasca, and R. Carloni, "Optimal event handling by multiple unmanned aerial vehicles," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 1230–1236.

[72] C. Nam and D. A. Shell, "When to do your own thing: Analysis of cost uncertainties in multi-robot task allocation at run-time," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 1249–1254.

[73] A. Meyerson, "Online facility location," in *IEEE Symposium on Foundations of Computer Science*. IEEE, 2001, pp. 426–431.

[74] D. Fotakis, "Incremental algorithms for facility location and k-median," *Theoretical Computer Science*, vol. 361, no. 2-3, pp. 275–313, 2006.

[75] A. Ghodselahi and F. Kuhn, "Serving online demands with movable centers," *arXiv preprint arXiv:1404.5510*, 2014.

[76] G. Divéki and C. Imreh, "Online facility location with facility movements," *Central European Journal of Operations Research*, vol. 19, no. 2, pp. 191–200, 2011.

[77] E. Koutsoupias, "The k-server problem," *Computer Science Review*, vol. 3, no. 2, pp. 105–118, 2009.

[78] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[79] L. Rayle, S. Shaheen, N. Chan, D. Dai, and R. Cervero, "App-based, on-demand ride services: Comparing taxi and ridesourcing trips and user characteristics in san francisco university of california transportation center (uctc)," *University of California, Berkeley*, 2014.

[80] N. Diakopoulos. How Uber surge pricing really works. [Online]. Available: https://www.washingtonpost.com/news/wonk/wp/2015/04/17/how-uber-surge-pricing-really-works/

[81] A. Rosenblat and L. Stark, "Algorithmic labor and information asymmetries: A case study of Uber's drivers," *International Journal Of Communication*, 2016.

[82] W. Zhang, S. Guhathakurta, J. Fang, and G. Zhang, "The performance and benefits of a shared autonomous vehicles based dynamic ridesharing system: An agent-based simulation approach," in *Transportation Research Board 94th Annual Meeting*, no. 15-2919, 2015.

[83] M. Hyland and H. S. Mahmassani, "Dynamic autonomous vehicle fleet operations: Optimization-based strategies to assign AVs to immediate traveler demand requests," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 278–297, 2018.

[84] M. Chang, D. S. Hochbaum, Q. Spaen, and M. Velednitsky, "DISPATCH: an optimal algorithm for online perfect bipartite matching with i.i.d. arrivals," *CoRR*, vol. abs/1805.02014, 2018.

[85] M. Maciejewski, J. Bischoff, and K. Nagel, "An assignment-based approach to efficient real-time city-scale taxi dispatching," *IEEE Intelligent Systems*, vol. 31, no. 1, pp. 68–77, 2016.

[86] Uber. Driving with Uber, wait less, earn more. [Online]. Available: www.uber.com/ca/en/price-estimate/

[87] S. L. Smith, M. Pavone, M. Schwager, E. Frazzoli, and D. Rus, "Rebalancing the rebalancers: Optimally routing vehicles and drivers in mobility-on-demand systems," in *IEEE American Control Conference*, 2013, pp. 2362–2367.

[88] M. Salazar, F. Rossi, M. Schiffer, C. H. Onder, and M. Pavone, "On the interaction between autonomous mobility-on-demand and public transportation systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2262–2269.

[89] S. Bandyapadhyay, A. Banik, S. Das, and H. Sarkar, "Voronoi game on graphs," *Theoretical Computer Science*, vol. 562, pp. 270–282, 2015.

[90] R. Salhab, J. Le Ny, and R. P. Malhamé, "A dynamic ride-sourcing game with many drivers," in *55th Annual Allerton Conference on Communication, Control, and Computing*, 2017, pp. 770–775.

[91] R. Schuler, “An algorithm for the satisfiability problem of formulas in conjunctive normal form,” *Journal of Algorithms*, vol. 54, no. 1, pp. 40–44, 2005.

[92] T. Basar and G. J. Olsder, *Dynamic noncooperative game theory.* Siam, 1999, vol. 23.

[93] (2016) TLC yellow taxi trip data set. [Online]. Available: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[94] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[95] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[96] The Canadian Automobile Association. Driving costs. [Online]. Available: www.carcosts.caa.ca/