Deployment Algorithms for Multi-Agent Exploration and Patrolling

by

Mikhail Volkov
B.A., B.A.I., University of Dublin, Trinity College (2010)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2013

Author .....
Department of Electrical Engineering and Computer Science
January 28, 2013

Certified by .....
Daniela Rus
Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students

Deployment Algorithms for Multi-Agent Exploration and Patrolling

by

Mikhail Volkov

B.A., B.A.I., University of Dublin, Trinity College (2010)

Submitted to the Department of Electrical Engineering and Computer Science
on January 28, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Exploration and patrolling are central themes in distributed robotics. These deployment scenarios have deep fundamental importance in robotics, beyond the most obvious direct applications, as they can be used to model a wider range of seemingly unrelated deployment objectives.

Deploying a group of robots, or any type of agent in general, to explore or patrol in dynamic or unknown environments presents us with some fundamental conceptual steps. Regardless of the problem domain or application, we are required to (a) understand the environment that the agents are being deployed in; (b) encode the task as a set of constraints and guarantees; and (c) derive an effective deployment strategy for the operation of the agents. This thesis presents a coherent treatment of these steps at the theoretical and practical level.

First, we address the problem of obtaining a concise description of a physical environment for robotic exploration. Specifically, we aim to determine the number of robots required to be deployed to clear an environment using non-recontaminating exploration. We introduce the medial axis as a configuration space and derive a mathematical representation of a continuous environment that captures its underlying topology and geometry. We show that this representation provides a concise description of arbitrary environments, and that reasoning about points in this representation is equivalent to reasoning about robots in physical space. We leverage this to derive a lower bound on the number of required pursuers. We provide a transformation from this continuous representation into a symbolic representation.

We then present a Markov-based model that captures a pickup and delivery (PDP) problem on a general graph. We present a mechanism by which a group of robots can be deployed to patrol the graph in order to fulfill specific service tasks. In particular, we examine the problem in the context of urban transportation, and establish a model that captures the operation of a fleet of taxis in response to incident customer arrivals throughout the city. We consider three different evaluation criteria: minimizing the

number of transportation resources for urban planning; minimizing fuel consumption for the drivers; and minimizing customer waiting time to increase the overall quality of service.

Finally, we present two deployment algorithms for multi-robot exploration and patrolling. The first is a generalized pursuit-evasion algorithm. Given an environment we can compute how many pursuers we need, and generate an optimal pursuit strategy that will guarantee the evaders are detected with the minimum number of pursuers. We then present a practical patrolling policy for a general graph. We evaluate our policy using real-world data, by comparing against the actual observed redistribution of taxi drivers in Singapore. Through large-scale simulations we show that our proposed deployment strategy is stable and improves substantially upon the default unmanaged redistribution of taxi drivers in Singapore.

Thesis Supervisor: Daniela Rus
Title: Professor

Acknowledgments

Deus é fiel

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivations and Goals

Exploration and patrolling are central themes in distributed robotics. The simple task of deploying a robot to autonomously explore an unknown environment needs little motivation. With the Curiosity Rover having just celebrated its first birthday at the time of writing of this thesis, it would not be an overstatement to say that exploration is one of the most enduring pursuits in robotics.

The goal of exploration is to inspect and obtain a representation of some region under consideration. The goal of patrolling is to continuously inspect some region under consideration, while performing certain actions or fulfilling certain problem conditions. These deployment scenarios have obvious direct applications in surveillance, disaster response, and military operations. But beyond such direct and practical applications, exploration and patrolling have even deeper fundamental importance in robotics as they can be used to model an ever wider superset of seemingly unrelated deployment objectives.

To elaborate, exploration is fundamentally the *static* acquisition of information from an environment. In the most general sense, an agent or group of agents traverse an environment and record some kind of sensory data. The objectives of an exploration task can be different and will inevitably depend on the application in question. We consider the canonical example, wherein a robot is tasked to traverse

an unknown environment and record a physical map using sensor data. The task is accomplished when data has been gathered from every part of the environment, and the environment is fully mapped. This is the literal interpretation of exploration. But this condition is loose and is malleable to interpretation.

In another example, a robot could be deployed on the battlefield in search of land mines. In this case, the immediate physical environment may or may not be known; the robot may need to clear a specific area, or may need to find a safe path through a certain distance. Any combination of these and other parameters would result in different problems, but all are easily encoded as exploration problems by adjusting the conditions to express the requirements of the problem.

As a final example, we consider a special case of exploration considered in this thesis, called pursuit-evasion. In this scenario, a group of robots are required to sweep an unexplored environment and detect any intruders that are present. Pursuit-evasion is an example of non-recontaminating exploration, whereby an initially unexplored and contaminated region is cleared while ensuring that the cleared region does not become contaminated again. There are interesting real-life applications of pursuit-evasion. Aside from the literal interpretation (pursuing an evader), there are other scenarios that motivate non-recontaminating exploration. For example, in a disaster response scenario following an earthquake, a group of robots is required to locate all survivors in inaccessible areas, while the survivors are possibly moving in the environment. In the aftermath of the Tohoku Earthquake and Tsunami of 2011, the KOHGA3 and Quince search and rescue robots were deployed under these kind of circumstances (Figure 1-1).

Patrolling is an even more general abstraction of such deployment scenarios. Patrolling can be interpreted as the *dynamic* acquisition of data from an environment. In the most general sense, an agent or group of agents traverse an environment and *continuously* record some kind of sensory data. It is not hard to see that patrolling is a generalization of exploration. To motivate this, consider a special case of patrolling where the information being gathered from the environment is constant. If the goal of the task is to continuously acquire this information with time, then a single ac-

(a) KOHGA3 robot **(b)** Quince robot

Figure 1-1: KOHGA3 and Quince search and rescue robots deployed in the aftermath of the Tohoku Earthquake and Tsunami of 2011.

quisition is sufficient at all point, as we then have full knowledge of this information thereafter. Thus we can see that this simply reduces to an exploration problem, and exploration in the sense of static data acquisition is a special case of patrolling. If the data source changes with time, acquisition needs to occur continuously, so patrolling can be interpreted as "persistent exploration".

Once again, the goals of a patrolling task depend on the problem being modeled. In the canonical example, a surveillance robot patrols for an intruder in a closed environment. This is the most familiar interpretation of patrolling; in this scenario perhaps the robots are required to patrol regions where intrusion activity is more likely, and avoid safer regions. The task is never "accomplished" as such, since the patrolling is continuous, rather the objective now becomes to provide certain guarantees, for example that no intruders enter the patrolled area over a period of time.

To consider another example, a group of autonomous robots may be deployed to cater to an oil spill in the ocean. Following the Deepwater Horizon oil spill of 2010, the Seaswarm oil absorbing robot was deployed for exactly this purpose (Figure 1-2). Such robots may be required to patrol more frequently in regions where there is oil, and avoid clean regions. We may have an estimate of the rate of expansion of surfaced oil based on the hydraulic pressure at the source of the leak. However, we do not know where in the region the oil will surface. The robots patrolling the

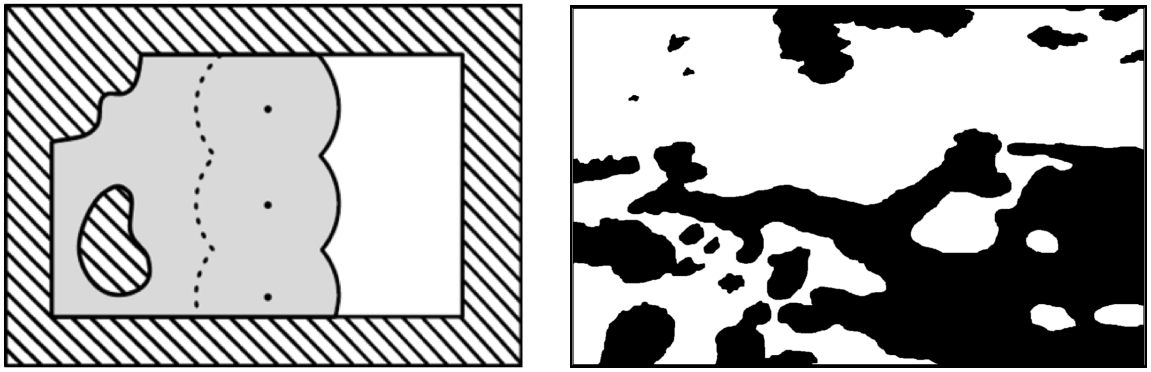
(a) Horizon oil spill **(b)** Seaswarm oil absorbing robot

Figure 1-2: Aerial view of the Deepwater Horizon oil spill, 2010 (left) and the oil absorbing robot deployed in the cleanup efforts (right).

region should cover the entire area at some minimum rate, in order to guarantee that an oil spill will not be greater than a certain radius. Clearly this is a different problem than the previous example, and the objective and criteria for success depend on many different parameters. Nonetheless, these two examples and the previous three exploration examples, can all be encoded as patrolling problems.

Deploying a group of robots, or any type of agent in general, to explore or patrol in dynamic or unknown environments presents us with some fundamental conceptual steps. Regardless of the problem domain or application, we can broadly categorize these as follows. We are required to (a) understand the environment that the agents are being deployed in; (b) encode the task as a set of constraints and guarantees; and (c) derive an effective deployment strategy for the operation of the agents.

There are notable technical challenges involved in each step. In the first step (a), we need to understand what kind of properties of the environment we are interested in: for example, intuition tells is that width is important; and how they can affect the deployment scenario: the width of the environment may affect the number of agents required to explore it. This is difficult because even in trivial cases it is already unclear how to interpret the simplest geometric parameters (for example in Figure 1-3a it is already unclear how to define the "width" of the environment). In more realistic environments (e.g. Figure 1-3b) our intuition fails and we must be very precise about what kind of metrics we need to obtain and how they relate to the specific scenario



(a) Trivial environment **(b)** Non-trivial environment

Figure 1-3: A trivial example environment (left) and a more realistic non-trivial environment (right). The free region is shown in white and the obstacle region is shown in black.

that is under consideration.

In the second step (b), the difficulty lies in identifying the right level of abstraction for the model. On the one hand, a model that is too simple will not be general enough to express the nuances that fully capture the problem requirements. The problem constraints should be fully captured by the model, and the necessary guarantees should be verifiable from the solution format. On the other hand, if the model is too complicated, it may be difficult to develop a tractable solution and/or to encode the solution as a deployment algorithm. Different levels of abstraction yield different results: some are more general, some more efficient, some more elegant. Which level of abstraction is most suitable depends on the particular requirements of the problem.

In the third step (c), the challenge is, put simply, to find a solution that works, and to prove that it works in theory and in practice. The solution can be imperative – an algorithm that directly dictates the deployment of the agents, or it can be passive – a policy that defines a set of rules according to which the agents make decisions as the deployment scenario unfolds. Generally speaking, a sound algorithm or policy should at least guarantee progress and termination. Informally, we understand progress to mean that the algorithm or policy evolves in some progressive manner towards its objective, and we understand termination to mean that the algorithm will eventually complete. Finally, proving that the solution is sound is the primary challenge on the theory front. But equally important is that the solution actually works in a real

application.

This thesis presents a coherent treatment of these steps at the theoretical and practical level. In consideration of the first step (a), we present a novel methodology for understanding an arbitrary continuous environment. We introduce tools for analyzing the environment's underlying topology and geometry. We present an algorithm for obtaining a concise configuration space representation of a physical environment. We derive reasoning about this representation, and show how it can be transformed into a discrete acyclic graph representation of the environment.

In consideration of the second step (b), first we consider a special case of exploration called pursuit-evasion. We show how the problem constraints of this exploration scenario can be encoded as rules within a discrete environment representation. We establish a mechanism for deploying agents in the environment by casting these rules as a game. We then consider the more general problem of patrolling. We show how a patrolling problem can be set up on a general graph. We encode the problem constraints in a discrete Markov model that faithfully captures all the important properties in problem domain.

In consideration of the final step (c) we present two deployment algorithms. First, we present an exploration algorithm for pursuit-evasion. We show how our environment characterization can be used to derive an algorithm that will guarantee the evaders are detected with the minimum number of pursuers. Second, we present a policy for patrolling on a general graph, and show how this policy is used to solve an important real-life problem in urban transportation. We extend our Markov model to capture the operation of a fleet of service agents (taxis) patrolling the city in response to incident requests (arriving customers) throughout the day. Our goal is to compute a solution in the form of the required number of taxis in the system and their patrolling policy. We encode the solution as a scalable optimization problem and present a practical patrolling policy.

The thesis presents a coherent treatment of all the steps involved in developing a deployment algorithm for robotic exploration and patrolling: (1) understanding the problem specification, (2) characterizing and representing an arbitrary continuous

environment, (3) transforming the problem into a discrete representation, (4) developing an effective algorithm to solve the problem, and (5) applying the algorithm to a real-life application.

1.2 Contributions to Robotics

This thesis makes the following contributions:

- **Environment characterization for non-recontaminating exploration.** The first part of this thesis addresses the problem of obtaining a concise description of a physical environment for robotic exploration. We aim to determine the number of robots required to clear an environment using non-recontaminating exploration. We introduce the medial axis as a configuration space and derive a mathematical representation of a continuous environment that captures its underlying topology and geometry. We show that this representation provides a concise description of arbitrary environments, and that reasoning about points in this representation is equivalent to reasoning about robots in physical space.

- **Deployment strategy for non-recontaminating exploration.** We leverage our continuous environment representation to derive a lower bound on the number of required pursuers required to explore the environment using non-recontaminating exploration. We provide a transformation from this continuous representation into a symbolic representation. Finally, we present a generalized pursuit-evasion algorithm. Given an environment we can compute how many pursuers we need, and generate an optimal pursuit strategy that will guarantee the evaders are detected with the minimum number of pursuers.

- **Markov-Based Model for Urban Mobility Networks.** In the second part of this thesis we present a Markov-based urban transportation model that captures the operation of a fleet of taxis in response to incident customer arrivals throughout the city. We leverage data from a fleet of 16,000 taxis in Singapore to create a realistic model of taxi fleet operation in Singapore. We show

that a standard undirected clique model is highly restrictive in terms of the kind of transportation network that it can describe. We present an improved model and discuss the steps taken to ensure that the model is realistic while still complying with the Markov framework. We then present a mechanism by which an optimization problem can be set up to handle a sparse network while maintaining a computational complexity that is independent of the degree of precision of the model.

- **Deployment strategy for patrolling a Markov-based graph model, with application to urban mobility systems.** Using the above urban transportation model, we show how we can learn and interpret the current default behavior of taxi drivers within our framework, and prove that the current behavior is sub-optimal with respect to several evaluation criteria. We show how to compute a solution in the form of the required number of vehicles in the system and their redistribution policy. We then consider the solution with respect to three seemingly different optimization criteria. The first criterion considers the customers, whose end goal is to minimize the time spent waiting for a taxi. The second criterion considers the urban planning authority whose goal is to minimize the number of vehicles in the road network. The third criterion considers the cost and environmental implications of fuel consumption. We encode the solution as a scalable optimization problem and present a practical redistribution policy.

- **Experiments.** We evaluate our policy by comparing it against the actual observed redistribution of taxi drivers in Singapore. We present experimental results via implementation in large-scale traffic simulations and consider the extent to which optimization at different levels of abstraction can work together as part of a complete urban mobility system. We show that our proposed policy is stable and improves substantially upon the default unmanaged redistribution of taxi drivers in Singapore with respect to the three optimization criteria.

1.3 Relation to Previous Work

An important aim of our work is to bridge the different levels of abstraction that work to date has been grounded in. This section elaborates on this claim, and highlights the motivations for improvement over the existing state of the art. Chapter 2 provides a comprehensive survey of related work.

Environment characterization for pursuit-evasion has been considered for polygonal spaces. We highlight that these studies considered robots equipped with infinite range visibility sensors deployed in polygonal spaces. Consequently the scope of environment characterization was limited. By contrast we consider limited visibility sensors, and we do not require the environment to be polygonal.

The medial axis has previously been studied in the context of robotic navigation. In this work, we use the medial axis to capture the underlying topological and geometric properties of our environment, and use it to transform the problem into a graph formulation. In our work, we consider the medial axis in a completely novel way – by treating it as a configuration space to derive a robust representation of a continuous environment that captures its underlying properties. We then use the medial axis we then leverage existing exploration models to build a concise representation of arbitrary environments in continuous two-dimensional space.

Visibility-based pursuit-evasion has been considered for continuous two-dimensional spaces. Generally speaking, visibility-based algorithms do the correct thing locally, but do not rely on, or make any guarantees for, a global description of an environment. As such, they may not be guaranteed to terminate.

Pursuit-evasion on graphs that are representations of some environment date back more than four decades. Although discrete graph-based models offer termination and correctness guarantees, they assume the world is suitably characterized and make no reference to the underlying physical geometry of the environment that is being represented.

In our work, we make use of the medial axis to establish a transformation from a continuous representation of an environment into the discrete domain. First, this

allows us to calculate bounds on the number of robots required to clear an environment, and to provide termination guarantees for existing pursuit-evasion algorithms. Second, this establishes an application platform for existing graph-based algorithms making them applicable to continuous environment descriptions.

More generally, we establish techniques which can in principle be used to transform a low-level continuous representation into a high-level graph-based discrete representation, and thus offer a mechanism for transforming any low-level exploration or patrolling problem into high-level problem.

Many different application scenarios of the exploration and patrolling model have been studied, as motivated by the examples in the preceding discussion and in Figures 1-1 and 1-2. As an end point for our work we consider an application of some practical significance – urban mobility has been an active area of research since the turn of the century. In the US, the annual congestion cost is projected to grow to $133 billion by 2015 [22]. Not surprisingly, social and municipal trends are changing in favor of a modernized system of public transportation, and the recent volume of research in the subject reflects this.

Dynamic Traffic Assignment problems generally aim to optimized traffic flow while accounting for congestion effects. DTA models commonly differ widely in the representation of the supply and demand processes. Mobility-on-Demand (MOD) is a newer paradigm for handling traffic congestion. The Pickup and Delivery problem (PDP) is a paradigm for handling traffic congestion whereby passengers arriving into a network are transported to a delivery site by vehicles. Load balancing in the Mobility-on-Demand systems is a similar a problem. As well as system-level traffic flow considerations, socially motivated objectives have also been considered, where the collective optimization criteria of the entire system are incentivized in favor of individual optima.

DTA and PDP problems are characterized by inherent mathematical intractability and challenging complexities. One practical consequence of this is that research has favored the development of heuristic solutions that emphasize effectiveness, robustness, and deployment efficiency over claims of uniqueness or global optimality

that may not be essential or particularly meaningful in the practical applications. As such, solutions tend to be less general, and are useful within narrow margins of the problem context.

Our work extends to encompass a broad scope of optimization criteria. We consider the interplay between global optimization criteria typical of related studies of DTA and MOD systems, as well as social optimization criteria as motivated by recent studies of congestion-aware traffic systems.

We believe that urban mobility is an exciting application scenario for robotics algorithms, in particular for the following reason. Infrastructure and technology developments are resulting in MOD systems becoming "smart" to varying degrees. Vehicles can now drive themselves and human drivers often rely on automated navigation systems. The lines between what is a manned system and what is an autonomous system are becoming blurred. Developments in distributed robotics are becoming ever more pertinent to vehicle operation, navigation systems and traffic networks. The application of our algorithms traditionally developed for autonomous robots to human taxi drivers offers us new insights into transportation systems and sets precedent for the urban mobility systems of the future.

1.4 Thesis Organization

This thesis is organized into eight chapters. Chapter 2 provides a comprehensive survey of related work. Chapter 3 presents a mechanism for obtaining a concise representation of a physical environment for robotic exploration. Chapter 4 presents an optimal deployment algorithm for a group of robots engaged in distributed non-recontaminating exploration. Chapter 5 presents a Markov-based urban mobility model and a mechanism by which a group of robots can patrol the graph to service persistent requests. Chapter 6 presents a practical deployment policy for patrolling the urban mobility model and shows how our algorithm offers several improvements over current ground truth behavior. We conclude the thesis in Chapter 7, reflect on the contributions of the work and consider lessons learned for future work in this area.

Chapter 2

Related Work

This thesis builds on ground-breaking prior research in robotic algorithms for exploration, pursuit-evasion, and task allocation. The research also builds on prior research on modeling and lower bounds for decentralized algorithms. In this chapter we detail some of the key results.

2.1 Exploration, Pursuit-Evasion and Patrolling

Visibility-based pursuit-evasion in continuous two-dimensional space was first introduced in [43]. Frontier-based exploration was introduced in [49] and extended to multiple robots in [50]. In [10] the authors consider limited visibility frontier-based pursuit-evasion in non-polygonal environments, making use of the fact that not allowing recontamination means we do not need to store a map of the environment; here a distributed algorithm is presented which works by locally updating the frontier formed by the sensor footprints of the robots. Another distributed model was more recently considered in [6]. Limited visibility was also considered in [41] which presents an algorithm for clearing an unknown environment without localization. In [13] the problem was considered for a single searcher in a known environment.

Environment characterization for pursuit-evasion has been considered for polygonal spaces. In [15],[17] pursuit-evasion in connected polygonal spaces is investigated, and tight bounds derived on the number of pursuers necessary based on the number of

polygon edges and holes. In [29] basic environment characterization is established, by means of a general decomposition concept based on a finite complex of conservative cells. In [48] similar ideas were explored, and several metrics proposed for primitive characterization of polygonal spaces.

The medial axis has previously been studied in the context of robotic navigation. For example, in [16],[20],[47] the medial axis was used as a heuristic for probabilistic roadmap planning. One of the key features that makes the medial axis attractive for such applications is that it captures the connectivity of the free space.

Pursuit-evasion on graphs that are representations of some environment goes back as early as [33],[35]. Randomized pursuit strategies on a graph are considered in [1]. Roadmap-based pursuit-evasion is considered in [24] and [39] where the pursuer and evader share a map and act according to different rules. In [39] a graph-based representation of the environment is used to derive heuristic policies in various scenarios. More recently, [27] presents a graph-based approach to the pursuit-evasion problem whereby robots use blocking or sweeping actions to detect all intruders in the environment. In [25] and [26] the more general graph variant of the problem was reduced to a tree by blocking edges.

Many other approaches and starting assumptions have been explored. Probabilistic evader detection, where multiple searchers are tasked with efficiently locating evaders is studied in [21], while [19] presents a greedy probabilistic policy for multiple pursuers. Roadmap-based pursuit-evasion, where the pursuer and evader share a map and act according to different rules is studied in [24] and [39]. Randomized pursuit strategies under different conditions are investigated in [23]. An adaptive planning strategy for pursuit-evasion in unknown environments is presented in [2].

2.2 Applications to Urban Mobility

The Dynamic Traffic Assignment problem (DTA) dates back as early as [31] and [12]. Generally speaking, the objective of DTA problems is to optimize traffic flow while accounting for congestion effects. A thorough survey can be found in [38]. For

example, the problem is grounded in continuous time control theory in [12], while [11] presents a variational inequality formulation. A mathematical programming approach in used in [31], [51]. Another recent work [51] models the problem as a linear program, and a simulation-based approach in [3] presents an offline model for estimation of supply and demand.

Mobility-on-Demand (MOD) is an emerging paradigm for handling traffic congestion. In MOD systems, the goal is to provide users with on-demand rental facilities of convenient and efficient modes of transportation [32]. Load balancing in MOD systems is similar to the Pickup and Delivery problem (PDP), whereby passengers arriving into a network are transported to a delivery site by vehicles. For a review of the state of the art see [4], [34] and the references therein. Autonomous load rebalancing in MOD systems has recently been studied in [36] and [37], where a fluid model was used to represent supply and demand.

As well as system-level traffic flow optimizations, socially motivated criteria have also been considered. Recent work on traffic planning explored optimizing a drivers's route subject to congestion [30]. Social optimum planning models for computing vehicle paths are presented in [40, 46].

Chapter 3

Environment Characterization for Non-Recontaminating Exploration

This chapter[1] addresses the problem of obtaining a concise description of a physical environment for robotic exploration. We aim to determine the number of robots required to clear an environment using non-recontaminating exploration. We introduce the medial axis as a configuration space and derive a mathematical representation of a continuous environment that captures its underlying topology and geometry. We show that this representation provides a concise description of arbitrary environments, and that reasoning about points in this representation is equivalent to reasoning about robots in physical space. Finally, we leverage this representation to derive a lower bound on the number of required pursuers.

This chapter of the thesis is organized as follows. In Section 3.1 we provide a formal model for non-recontaminating exploration and state the problem that we are addressing. In Section 3.2 we introduce the medial axis as a configuration space and show that reasoning about points in this space is equivalent to reasoning about robots in the physical world. We formalize the notion of width, corridors and junctions and derive bounds on the number of robots required to traverse a junction.

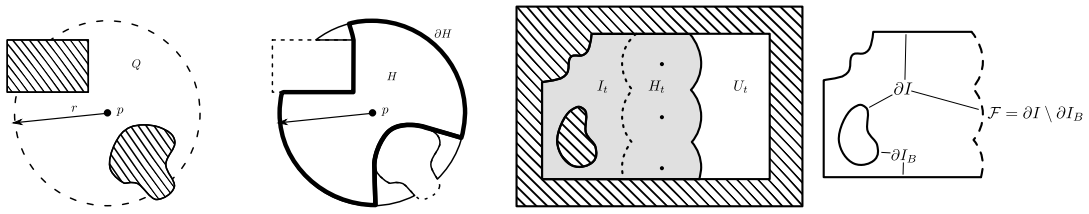
[1] The majority of this chapter was published in [45].

3.1 Problem Formulation

We now present a formal model of the problem we are addressing. Our model builds on the notation and terminology introduced in [10]. We have a team of n exploration robots deployed in the Euclidean plane $\mathbb{R}^2$. Each robot is equipped with a holonomic (uniform in all orientations) sensor that records a line of sight perception of the environment within a maximum sensing radius r. We assume that two robots can reliably communicate if they are within line of sight of each other and if the distance between their positions is less than or equal to $2r$.

The position of a robot is constrained to be within some *free region* Q, which is a closed compact subset of $\mathbb{R}^2$. The *obstacle region* B makes up the rest of the world, and is defined as the complement of Q. In this work we require both Q and B to be connected spaces, which means there are no holes in the environment. We define the *obstacle boundary* ∂B as the oriented boundary of the obstacle region (which by definition is the oriented boundary of the free region).

We assume a continuous time model, i.e. time $t \in \mathbb{R}_{\geq 0}$. Let H_t^i be the holonomic *sensor footprint* of robot i at time t, which is defined as the subset of Q that is within direct line of sight of robot i and within distance r of robot i. Formally, if $p \in \mathbb{R}^2$ is the position of robot i at time t, then $H_t^i = \{x \in Q \mid d(p,x) \leq r \wedge \forall y \in \big[px\big], y \in Q\}$ where $d(x,y)$ is the Euclidean distance between x and y (see Fig. 3-1a). Let H_t be the union of the sensor footprints of all robots at some time t, given by $H_t = \bigcup_{i=0}^{n} H_t^i$. This corresponds to the region being sensed by the robots at time t. We define the *inspected region* $I_t \subseteq Q$ as the union at time t of all previously recorded sensor footprints, given by $I_t = \{p \in \mathbb{R}^2 \mid \exists t_0 \in [0,t] \text{ such that } p \in H_{t_0}\}$. The *contaminated region* (or unexplored region) U_t is defined as the free space that has not been inspected by time t, given by $U_t = Q \setminus I_t$ (see Fig. 3-1b). Note that at time $t = 0$ the contaminated region is given by $Q \setminus H_0$. We define the *cleared region* $C_t \subseteq I_t$ as the inspected region that is not currently being sensed, given by $C_t = I_t \setminus H_t$. We say that *recontamination* occurs at time t if the cleared region C_t comes in contact with the contaminated region U_t (we understand two regions to be in contact if the intersection of their closure is



(a) Sensor Footprint **(b)** Inspected region **(c)** Boundaries

Figure 3-1: Fig. 3-1a shows a robot located at some position p within the free region Q. The sensor footprint H and its oriented boundary ∂H are shown on the right. Fig. 3-1b shows the inspected region I_t and the contaminated region U_t at some time t. Fig. 3-1c shows the inspected region boundary ∂I, the inspected obstacle boundary ∂I_B and the frontier $\mathcal{F}$.

non-empty, i.e. $\mathrm{cl}(C_t) \cap \mathrm{cl}(U_t) \neq \emptyset$).

When time is clear from context, we understand I and U to mean the current inspected region and the current contaminated region, respectively. We define the *inspected region boundary* ∂I as the oriented boundary of the inspected region I. We define the *inspected obstacle boundary* ∂I_B as the intersection of the inspected region boundary and the obstacle boundary, given by $\partial I_B = \partial I \cap \partial B$. We define the *frontier* $\mathcal{F}$ as the free (non-obstacle) boundary of the inspected region, given by $\mathcal{F} = \partial I \setminus \partial I_B$ (see Fig. 3-1c). Observe that by definition the frontier $\mathcal{F}$ separates the free region Q into the inspected region I and the contaminated region U. Observe also that the frontier need not be connected, and is in general the union of one or more disjoint maximally connected arcs. We understand the frontier of a group of robots $\mathcal{F}' \subseteq \mathcal{F}$ to mean a single maximally connected arc of the total frontier formed by the exterior boundary of the sensor footprints of that group of robots.

The goal of exploration algorithms is to inspect the entire free region. For non-recontaminating exploration the goal is to inspect the entire fee region without admitting recontamination. In both cases we say that an environment has been successfully explored if $I_t = Q$ at some time t. In this thesis we deal specifically with non-recontaminating exploration and present an algorithm that is guaranteed to explore a space without admitting recontamination.

An algorithm for exploration relies on robots to "expand" the frontier boundary until the entire free region becomes inspected. However, in a non-recontaminating exploration algorithm the goal is not only to expand, but also to "guard" the fron-

tier, ensuring that the inspected region does not become contaminated again. This difference makes non-recontaminating exploration more restrictive than conventional exploration. For example, observe that regardless of the size of the sensing radius of the robots (as long as $r > 0$), or the properties of the world (as long as Q is a connected space), a single robot can always explore the world. However, in non-recontaminating exploration this is not true in general. Informally speaking, if the "width" of the corridors in the free region is larger than the sensing radius of a robot, then it should appear obvious that a single robot cannot simultaneously expand and guard the frontier to inspect the entire free region.

Consider the simple rectangular free region Q shown in Fig. 3-1b. We can reason that if the width of Q is less than the sum of the sensor diameters of the n robots, then the environment can be explored without admitting recontamination. However, even in this simple example it is not completely clear what is meant by width. Notice that if we consider width to be the distance from the left to the right border then this reasoning fails — in this case width would specifically mean the smaller of the two dimensions. So it is already non-trivial how to characterize a very simple environment, and things become much more complicated in non-rectangular environments.

In this thesis we study the relationship between an environment Q, the sensor radius r, and the number of robots n required for non-recontaminating exploration of Q. Intuition tells us that corridor width and junctions are important features. We formalize the notion of corridors and junctions and present a general method for computing a configuration space representation of the environment that captures this intuition. We show that this representation provides a concise description of arbitrary environments.

A canonical example of non-recontaminating exploration is pursuit-evasion. In this scenario there is a group of robot pursuers and a group of robot evaders deployed in the free region Q. The evaders are assumed to be arbitrarily small and fast. The goal of the pursuers is to catch the evaders (by detecting their presence within the sensor footprint), and the goal of the evaders is to avoid getting caught. Whenever part of the frontier is not being guarded by the pursuers, the evaders can move

undetected from the contaminated region to the previously inspected region, thereby recontaminating it.

3.2 Environment Analysis

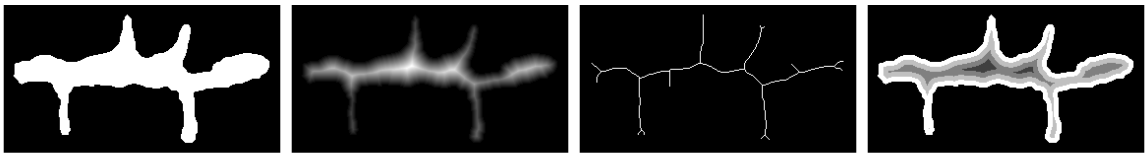
In this section we present the medial axis as a configuration space and show that reasoning about points in this configuration space is equivalent to reasoning about robots in physical space. First, we establish the necessary geometric framework, accompanied by a series of definitions and claims. Second, we introduce an exploration model in this configuration space and justify that it allows us to reason about the physical movement of the robots in the environment.

3.2.1 Environment Geometry

The *distance transform* is a mapping $\mathcal{D} : \mathbb{R}^2 \to \mathbb{R}$ where $\mathcal{D}(x) = \min_{y \in B} \{d(x, y)\}$ and $d(x, y)$ is the Euclidean distance between x and y (extending definition in [8] to the continuous domain) (see Fig. 3-2b). Observe that by definition if $x \notin Q$ then $\mathcal{D}(x) = 0$. The distance transform of a point $x \in Q$ captures the notion of "undirected width" of a region around a point x in free space, that is we get a measure of how wide or narrow a region is without being explicit about orientation.

The *medial axis* or *skeleton* S of a free space is defined as the locus of the centers of all maximal inscribed circles in the free space [7] (see Fig. 3-2c). Equivalently, the skeleton can be defined as the locus of quench points of a fire that has been set to a grass meadow at all points along its boundary [5], [42]. The skeleton captures the topology of the free space, and aids us in determining which parts of an environment should be considered "corridors" and which parts should be considered "junctions" of multiple corridors.

The *degree* of a point $x \in S$ is given by the function $\theta : S \to \mathbb{N}_{>0}$ which maps every point on the skeleton to a natural number k. Specifically, we define a point $x \in S$ to have degree $\theta(x) = k$ if there exists an $a \in \mathbb{R}_{>0}$ such that $\forall \varepsilon \in (0, a]$ a circle centered at x of radius ε intersects the skeleton S at exactly k points.



(a) Environment image **(b)** Distance transform **(c)** Skeleton **(d)** Relief map

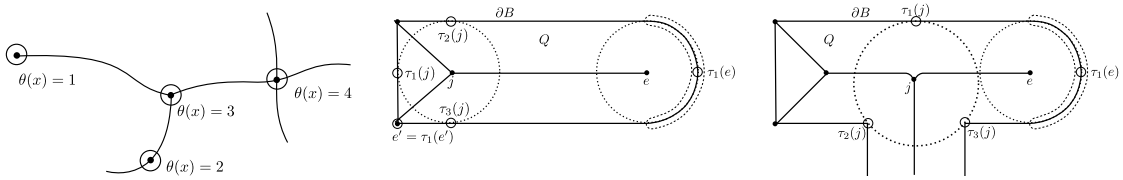
Figure 3-2: The environment is represented by a binary image in Fig. 3-2a. Fig. 3-2b shows the distance transform $\mathcal{D}$ of the environment. Fig. 3-2c shows the skeleton S. Fig. 3-2d shows the relief map quantization. The relief contours indicate multiples of the sensing radius r.

Borrowing notation from [9], we use the degree of a point $x \in S$ to distinguish between three types of points on the skeleton: corridor points, end points and junction points. Specifically, for a point $x \in S$, we say x is an *end point* if $\theta(x) = 1$, x is a *corridor point* if $\theta(x) = 2$, and x is a *junction point* if $\theta(x) > 2$ (see Fig. 3-3a). We refer to a continuous arc of corridor points on the skeleton simply as a *corridor*.

An alternative definition for $\theta(\cdot)$ can be stated as follows. For a point $x \in S$ let C be the maximal inscribed circle centered at x, and let G be the intersection of this circle with the obstacle boundary, given by $G = C \cap \partial B$. (Observe that by definition C has radius $\mathcal{D}(x) \neq 0$, and since C is maximal, G is non-empty.) Then $\theta(x)$ is defined as the number of maximally connected arcs in G. This definition for for $\theta(\cdot)$ is equivalent to the previous one [7], [14], [28].

Let $G_1, G_2, \ldots, G_{\theta(x)}$ be the set of maximally connected arcs of G. Note that in most cases these arcs are in fact just single points, which corresponds to the intuitive notion of the circle being tangent to the boundary at these points. A cursory glance reveals that this is the case for most corridor points and junction points. For end points that lie on the obstacle boundary, the tangent point coincides with the end point itself. However, the generality is necessary in a few special cases, such as end points of regions that taper off in a sector. In these cases the maximal inscribed circle C will be tangent to the obstacle boundary at a continuous arc segment of points. In order to simplify the discussion we define the tangent points $\tau_1(x), \tau_2(x), \ldots, \tau_{\theta(x)}(x)$ of a point $x \in S$ as the midpoints of the tangent arcs $G_1, G_2, \ldots, G_{\theta(x)}$ (see Fig. 3-3b).

We define a *boundary wall* as a maximally connected arc segment of the obstacle boundary ∂B that does not contain a tangent point of any end point. Formally



(a) Skeleton points **(b)** Dead end tangent point **(c)** Junction tangent points

Figure 3-3: Fig. 3-3a shows the three types of points on the skeleton: end points, corridor points, and junction points. A corridor is a continuous arc of corridor points. Fig. 3-3b shows three points on the skeleton e, e', j and their respective tangent points. For end point e the tangent point $\tau_1(e)$ is the midpoint of the tangent arc segment shown (dotted outline). For end point e' the tangent point $\tau_1(e')$ coincides with the end point itself. For junction point j there are $\theta(j) = 3$ tangent points τ_1, τ_2, τ_3. Fig. 3-3c shows a junction point j of degree $\theta(j) = 3$. All 3 tangent points τ_1, τ_2, τ_3 are located on distinct boundary walls.

$\partial B_0 \subset \partial B$ is a boundary wall if it is a maximally connected arc segment such that $\forall e \in S \mid \theta(e) = 1, \tau(e) \notin \partial B_0$.

Lemma 1. *For a junction point $j \in S$, the $\theta(j)$ tangent points of j are located on $\theta(j)$ distinct boundary walls.*

Proof. A circle C of radius $\mathcal{D}(j)$ centered at a junction point j will be tangent to the obstacle boundary at the $\theta(j)$ tangent points of j. Each pair of adjacent tangent points $\tau_i, \tau_{i+1} \in C$ (in the sense of counter-clockwise orientation along C) will be on opposite sides of one corridor. Consider the obstacle boundary arc segment $[\tau_i\ \tau_{i+1}]$ (in the sense of counter-clockwise orientation along ∂B). If e is the end point of the corridor that straddles the interior of $[\tau_i\ \tau_{i+1}]$, then $\tau(e)$ will lie on $[\tau_i\ \tau_{i+1}]$. Thus the boundary wall containing τ_i is disjoint from the boundary wall containing τ_{i+1} since neither contains $\tau(e)$. Since this applies for each pair of adjacent tangent points, we conclude that all $\theta(j)$ tangent points $\tau_1, \tau_2, \ldots, \tau_{\theta(j)}$ will be located on $\theta(j)$ distinct boundary walls (see Fig. 3-3c). □ □

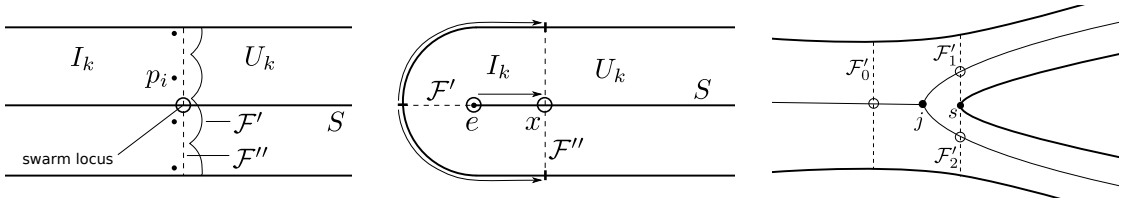
3.2.2 Exploration Model

We now show how we can use the preceding definitions and geometric claims to form a model for frontier-based exploration and establish an equivalence between the the medial axis configuration space and the physical environment.

We claim that reasoning about a single point moving along the skeleton allows us to reason about a group of robots that form a frontier with their end to end sensor footprints moving through physical space. We call this point the *swarm locus*. By definition a corridor point $x \in S$ has exactly two tangent points. For a swarm locus stationed at x we call these two points the frontier *anchor points*. The frontier of a group of robots is represented by a corresponding frontier formed by two line segments joining the swarm locus to its anchor points. A group of robots engaged in non-recontaminating exploration will form a frontier arc subtended between two obstacle boundary walls in physical space; the frontier arc separates the inspected region on one side from the contaminated region on the other side. Our abstraction allows us to reason in similar terms: a swarm locus stationed at a point x on the corridor of the skeleton will similarly form a frontier arc consisting of two line segments subtended between two obstacle boundary walls; the frontier arc transposed onto Q likewise separates the inspected region from the contaminated region (see Fig. 3-4a). We understand the frontier of a swarm locus to mean the frontier $\mathcal{F}' \subseteq \mathcal{F}$ of a group of robots represented by a swarm locus stationed at a point $x \in S$.

Note that we are making a simplifying abstraction in representing the frontier $\mathcal{F}' \subseteq \mathcal{F}$ of a group of n_0 robots by two end to end line segments subtended between two obstacle boundary walls. Observe that as n grows, the abstraction becomes more accurate as the periodic protrusion of the frontier due to the curvature of sensor footprints becomes finer-grained and less prominent with respect to its length. In general, this abstraction is justified as we are usually interested in characterizing environments where $n \gg 0$.

For the purposes of introducing the exploration model we assume that the swarm locus always begins at an end point. (Note that this assumption only serves to simplify the discussion, and can be removed easily by introducing several special cases.) From the definition of the degree of a point on the skeleton, a maximal inscribed circle C centered at an end point $e \in S$ will be tangent to the obstacle boundary at a single point $\tau(e)$. Thus both anchor points are the same point $\tau(e)$ and the frontier $\mathcal{F}' \subseteq \mathcal{F}$ of a swarm locus stationed at e is formed by two identical line segments $[e\ \tau(e)]$. In



(a) Swarm locus **(b)** Initial frontier **(c)** Split frontier

Figure 3-4: Fig. 3-4a shows a group of robots at positions p_i forming a frontier $\mathcal{F}'$ with the exterior boundary of their sensor footprints. Superimposed is the corresponding swarm locus stationed at a point $x \in S$ forming a frontier $\mathcal{F}''$ with two line segments subtended between two obstacle boundary walls. Fig. 3-4b shows the initial frontier $\mathcal{F}'$ of a swarm locus stationed at an end point e, separating the environment into $I_0 = \emptyset$ and $U_0 = Q$. As the swarm locus moves along the skeleton to reach a point $x \in S$, it forms a frontier $\mathcal{F}''$. The swarm locus has swept across the environment and cleared the region to the left of $\mathcal{F}''$. Fig. 3-4c shows the configuration of a split frontier. A swarm locus is traversing a junction point j with $\theta(j) = 3$. The ingoing frontier $\mathcal{F}'_0$ splits, producing 1 split point s and 2 outgoing frontiers $\mathcal{F}'_1, \mathcal{F}'_2$.

this configuration, $\mathcal{F}'$ separates the environment Q into the inspected region $I_0 = \emptyset$ and the contaminated region $U_0 = Q$, corresponding to the fact that the swarm locus has not yet explored any of the environment. As the swarm locus starts moving along the skeleton, the anchor points will move along ∂B on either side of the corridor and the frontier will "sweep" across the environment. The frontier now separates Q into two disjoint nonempty regions. The inspected region begins growing, while the contaminated region begins shrinking, corresponding to the fact that the robots have begun clearing the environment (see Fig. 3-4b).

Moving Through Corridors

We define the *relief map* $\mathcal{R} : \mathbb{R}^2 \to \mathbb{N}$ as the quantization of the distance transform using the sensing radius r, given by $\mathcal{R}(x) = \lceil \mathcal{D}(x)/r \rceil$ (see Fig. 3-2d). The relief map uses the distance transform to similarly capture the notion of width, expressing the same information in terms of the number of robots required at a point x to reach the closest point on the obstacle boundary.

Lemma 2. *A group of n_0 robots represented by a swarm locus that reaches a corridor point $x \in S$ prevents recontamination if and only if $n_0 \geq \mathcal{R}(x)$.*

Proof. In order to prevent recontamination, the group of robots must subtend a frontier between two obstacle boundary walls in physical space. By definition, the

distance between $x \in S$ and the closest point to the obstacle boundary on either side is exactly $\mathcal{D}(x) \leq \mathcal{R}(x)$. Thus we can always produce two lines l_1, l_2 from x to the two closest points on the obstacle boundary of combined length $L \leq 2\mathcal{R}(x)$. If $n_0 \geq \mathcal{R}(x)$ then the robots can always align themselves in physical space such that their frontier passes through x in the same arrangement as l_1, l_2 of combined length $L \geq 2\mathcal{R}(x)$. Therefore a group of $n_0 \geq \mathcal{R}(x)$ robots can always form a frontier that subtends between two obstacle boundary walls, and recontamination can always be prevented. If $n_0 < \mathcal{R}(x)$ then any arrangement of the robots in physical space such that their frontier passes through x will always result in a frontier of length $L < 2\mathcal{R}(x)$. Therefore a group of $n_0 < \mathcal{R}(x)$ robots can never form a frontier that subtends between two obstacle boundary walls, and recontamination will always occur. □ □

Corollary 1. *A group of n_0 robots represented by a swarm locus moving along a corridor $G = [a\ b] \subset S$ prevents recontamination at all points $x \in G$ if and only if $n_0 \geq \max_{x \in G} \{\mathcal{R}(x)\}$.*

When a swarm locus reaches an end point, the situation is the reverse of that at the beginning. From the definition of the degree of a point on the skeleton, a maximal inscribed circle C centered at an end point $e \in S$ will be tangent to the obstacle boundary at a single point $\tau(e)$. Thus both anchor points are the same point $\tau(e)$ and the frontier $\mathcal{F}' \subseteq \mathcal{F}$ of a swarm locus stationed at e is formed by two identical line segments $[e\ \tau(e)]$. In this configuration, $\mathcal{F}'$ separates the environment Q into the inspected region I_t and some part of the contaminated region $\emptyset$, corresponding to the fact that the swarm locus has cleared a particular corridor. In the case where there is only one swarm locus, $\mathcal{F}'$ separates the environment Q into the inspected region $I_t = Q$ and the entire contaminated region $U_t = \emptyset$, corresponding to the fact that the entire environment has been cleared.

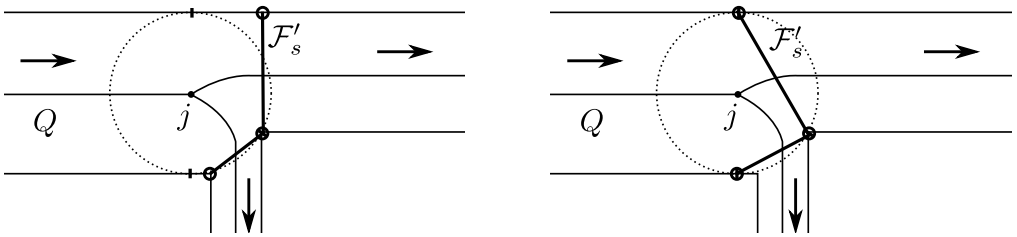
Traversing Junctions

When a group of robots reaches a junction in physical space, they should split and explore the outgoing junction corridors separately. If the robots do not split then recontamination will occur due to any of the unattended outgoing corridors coming in contact with the inspected region. Upon reaching a physical junction with some number of outgoing corridors, a single group of robots forms more than one group of robots with that number of disjoint maximally connected arcs making up the exterior boundary of their sensor footprints. Correspondingly, we define the frontier $\mathcal{F}'_0 \subseteq \mathcal{F}$ of a group of robots to *split* when the exterior boundary of the sensor footprints of the robots is no longer connected and becomes the union of two or more disjoint maximally connected arcs. Thus a group of robots splits when their frontier splits. For a junction point j with $\theta(j) - 1$ outgoing corridors, the junction is considered *traversed* when the frontier $\mathcal{F}'_0 \subseteq \mathcal{F}$ of the group of robots splits to form $\theta(j) - 1$ outgoing frontiers $\mathcal{F}'_1, \mathcal{F}'_2, \ldots, \mathcal{F}'_{\theta(j)-1} \subset \mathcal{F}$.

Observe that since the total frontier is at all times given by $\mathcal{F} = \partial I \setminus \partial I_B$, if one or more new disjoint maximally connected frontier arcs form, then by necessity one or more new disjoint maximally connected inspected obstacle boundary arcs also form. At the time t_0 that a frontier $\mathcal{F}'_0 \subseteq \mathcal{F}$ of a group of robots splits to form $\theta(j) - 1$ frontiers, it will have $\theta(j) - 2$ points of contact with the obstacle boundary. We call a frontier $\mathcal{F}'_0$ in such a configuration a *split frontier* and we call these points *split points* (see Fig. 3-4c). For $t > t_0$ the split points on the obstacle boundary grow into the $\theta(j) - 2$ new disjoint maximally connected inspected obstacle boundary arcs.

Split Frontier Bounds

We establish the lower bound as follows. To traverse a junction we require a split frontier $\mathcal{F}'_s$ to be formed with $\theta(j) - 2$ split points. The ingoing frontier $\mathcal{F}'_0$ subtends between two boundary walls $\partial B_1, \partial B_{\theta(j)}$, intersecting them at the frontier anchor points c_1, c_2. Therefore the split points $s_1, s_2, \ldots, s_{\theta(j)-2}$ are located on each of the other $\theta(j) - 2$ boundary walls $\partial B_2, \partial B_3, \ldots, \partial B_{\theta(j)-1}$. Without a split point on each



(a) Lower bound split frontier **(b)** Upper bound split frontier

Figure 3-5: Frontier configurations for lower and upper bound derivations.

of these boundary walls, the split frontier cannot be formed and the junction cannot be traversed. Hence, a lower bound on the number of robots required to traverse a junction is given by minimizing the length of the split frontier over all possible points on the respective boundary walls. Thus we can never form a split frontier of total length less than

$$\begin{aligned} \left\|\mathcal{F}'_{s,min}\right\| &= \min\left\{ \|c_1\ s_1\| + \|s_1\ s_2\| + \ldots + \left\|s_{\theta(j)-3}\ s_{\theta(j)-2}\right\| + \left\|s_{\theta(j)-2}\ c_2\right\| \right\}, \\ &\qquad \text{for } c_1 \in \partial B_1, s_1 \in \partial B_2, \ldots, s_{\theta(j)-2} \in \partial B_{\theta(j)-1}, c_2 \in \partial B_{\theta(j)}\ . \end{aligned} \tag{3.1}$$

If the number of robots is insufficient to form a split frontier of the smallest possible length, then it will certainly be insufficient to form a split frontier of any other length. But a split frontier must be formed in order to traverse a junction, irrespective of the exploration model. Thus no exploration model can dictate a configuration of robots that is able to traverse a given junction with fewer than $\|\mathcal{F}'_{s,min}\|$ robots. This establishes the lower bound (see Fig. 3-5a).

We establish the upper bound as follows. Assume we have some number of robots at a junction of degree $\theta(j)$. The ingoing frontier $\mathcal{F}'_0$ subtends between two boundary walls $\partial B_1, \partial B_2$, intersecting them at the frontier anchor points c_1, c_2. The ingoing frontier can always be aligned, without admitting recontamination, such that its anchor points correspond to the tangent points $\tau_1, \tau_{\theta(j)}$ on the two boundary walls that it subtends. By Lemma 1, a junction of degree $\theta(j)$ will have tangent points on $\theta(j)$ distinct boundary walls. Thereafter the ingoing frontier can always be extruded

toward each of the other $\theta(j) - 2$ tangent points in turn, likewise without admitting recontamination. Thus for a junction j with tangent points $\tau_1, \tau_2, \ldots, \tau_{\theta(j)}$ we can always form a split frontier of total length

$$\begin{aligned} \left\|\mathcal{F}'_{s,max}\right\| &= \|c_1\ s_1\| + \|s_1\ s_2\| + \ldots + \left\|s_{\theta(j)-3}\ s_{\theta(j)-2}\right\| + \left\|s_{\theta(j)-2}\ c_2\right\| \\ &= \|\tau_1\ \tau_2\| + \|\tau_2\ \tau_3\| + \ldots + \left\|\tau_{\theta(j)-1}\ \tau_{\theta(j)}\right\| \ . \end{aligned} \tag{3.2}$$

We can traverse any junction in this way, thus no junction will ever require more than $\|\mathcal{F}'_{s,max}\|$ robots to traverse. This establishes the upper bound (see Fig. 3-5b). Observe that since all the tangent points are on the boundary of a maximal inscribed circle centered at j, the ingoing frontier $\mathcal{F}'_0$, aligned such that its anchor points correspond to $\tau_1, \tau_{\theta(j)}$, is at most $2\mathcal{D}(j)$ in length, i.e. the diameter of the circle. For each of the $\theta(j) - 2$ split points, the frontier gains an additional line segment, likewise of at most $2\mathcal{D}(j)$ in length. Thus we get a numeric upper bound on the maximum length of the split frontier, given by

$$\left\|\mathcal{F}'_{s,max}\right\| \ \leq \ 2\big(\theta(j) - 1\big)\mathcal{D}(j) \ . \tag{3.3}$$

We have now established an equivalence between the medial axis configuration space and the physical movement and frontier expansion of robots in physical space. We introduced an exploration model whereby a swarm locus moving along the skeleton allows us to reason about a group of robots moving through physical space. We defined what it means for a frontier to split and for a group of robots to traverse a junction, and derived lower and upper bounds on the number of robots required to traverse a junction.

3.3 Implementation

From photograph to graph

The environment characterization algorithm was implemented in MATLAB. The procedure takes as an input a binary image W representing the environment under consideration. The binary image distinguishes the free region Q (encoded in white) from the obstacle region B (encoded in black).

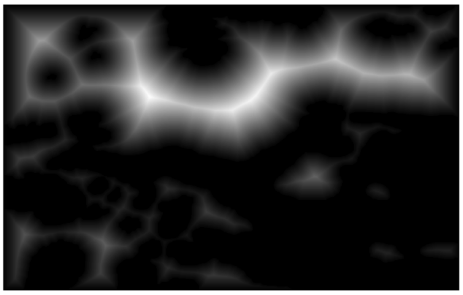
For illustration we will present results continuing from the real-world example presented in Figure 1-2, which depicts an aerial view of the Deepwater Horizon oil spill. In order to provide this environment as input to our environment characterization algorithm, it was necessary pre-process the aerial photograph into a binary image representation. (Note that for the purposes of this example as motivate by the Seaswarm oil absorbing robot, the free region is considered to be the body of water, while the land mass makes up the obstacle region.) The discretization of the binary image W is arbitrary, but for ease of illustration we assume it to be determined pixel-wise by the resolution of the aerial photograph.

The preprocessing step consists of a standard cascade of image processing techniques. First, colors were adjusted to accentuate the difference between greens and blues by remapping them onto a gray scale color map. Next we applied Canny edge detection and thresholding operations. This provides us with a single pixel outline of the coastline. Finally, judicious application of morphological opening and smoothing was carried out to remove noise. The exact technical details of the pre-processing step depend on the image being used, and are not directly part of the algorithm, but it worth noting that this step can also be automated and incorporated into the environment characterization algorithm, provided that assumptions are specified beforehand and satisfied by the input image. The preprocessing step was implemented in MATLAB using image processing toolbox.

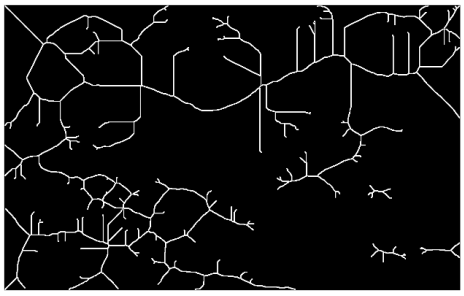
The resulting binary image W is shown in in Figure 3-6a. Note that although the pursuit-evasion algorithm described in this chapter requires both Q and B to be connected spaces, there is no such requirement for the environment characterization

(a) Aerial photograph of a typical real-world environment

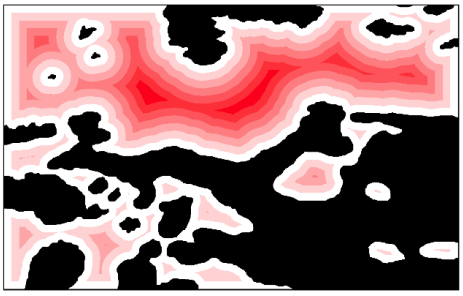
(b) Binary image representation of the environment in Figure 3-6a



(c) Distance transform of the environment in Figure 3-6b



(d) Medial axis of the environment in Figure 3-6b



(e) Relief map of the environment in Figure 3-6b

Figure 3-6: Example of implementation results for a typical real-world environment.

procedure. The configuration space representation generalizes to arbitrary environments, which means that the free region can have holes and can itself be disjoint. The environment encoded by W illustrates both of these cases.

We now compute the two main configuration space components, the relief map R and the medial axis or skeleton S. The relief map was computed as follows. First, the distance transform of W is computed, as described in Section 3.2.1. This was done using the MATLAB method `bwdist`. Note that distance is understood to be Euclidean distance from the boundary, and not the pixel-wise (Manhattan) distance. This is an important point of clarification when it comes to implementation, as several morphological and image processing techniques have counterparts defined as pixel set operations. The distance transform of W is shown in Figure 3-6c.

The relief map was then computed by labeling each point in the input binary image with the distance transform value at that point divided by the sensing radius r. The relief map R is shown in Figure 3-6e. To compute the skeleton, we make use of the MATLAB method `bwmorph` to apply successive morphological thinning to W. This is repeated until the next iteration does not change from the previous one; the resulting structure is the skeleton of the binary image. The skeleton S is shown in Figure 3-6b.

Finally, we need to convert the continuous configuration space representation into the symbolic graph representation. This was computed as follows. First, the endpoints and branch points of the skeleton were computed in MATLAB using the `bwmorph` function. The endpoints identify ends of line segments on the skeleton, and brach points identify junctions of two or more line segments (using the same definitions as described in Section 3.2.1. Next, we disconnect all line segments by removing all branch points from the skeleton, and assign a labeling $b_1, b_2, \ldots$ to each branch point and a labeling $l_1, l_2, \ldots$ to each unique line segment. Next, for each branch point b_i removed in this manner, we replace it on the skeleton, and then recompute the modified set of unique line segments $l'_1, l'_2, \ldots$. The general graph G representing the environment is then obtained as follows. The subset of line segments $l_u, l_v, \ldots$ that becomes a single line segment l'_w due to the replacement of a branch point b_i consti-

tutes the set of edges $e_u, e_v, \ldots$ connected to a vertex v_i on G. The resulting graph G is shown in Figure 3-7.

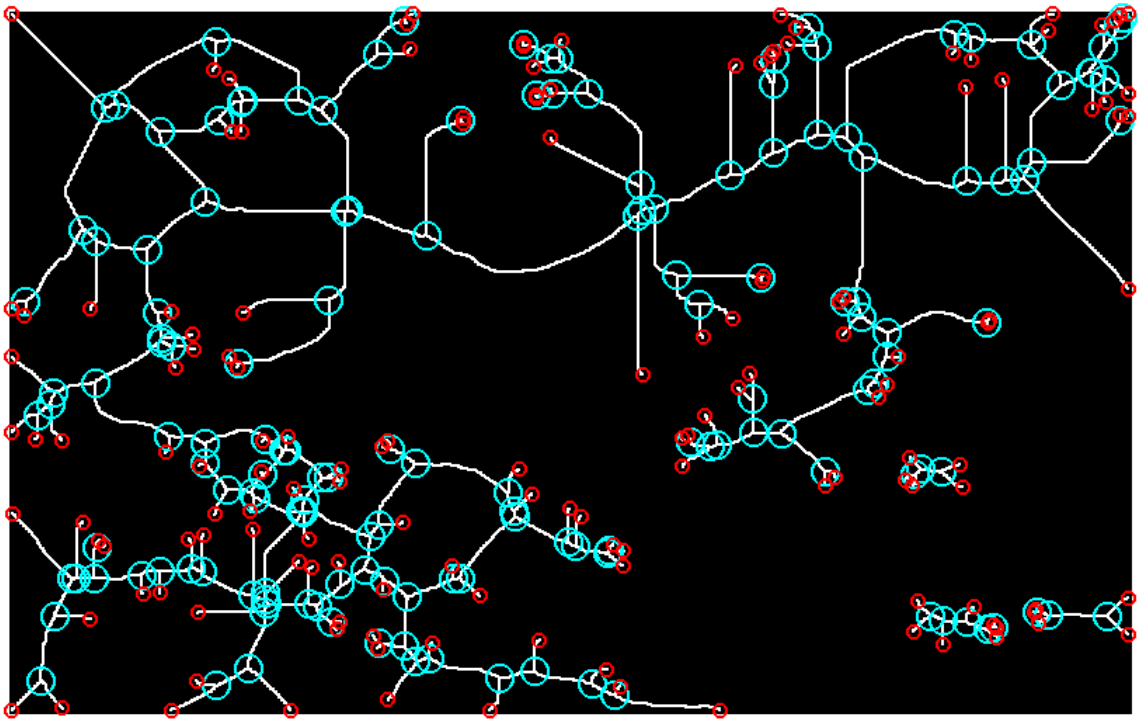


Figure 3-7: Image depicting the topology graph of the environment represented by the binary image in Figure 1-3b. Brach point are colored in cyan and endpoints are colored in red.

3.4 Summary

In this chapter the problem of obtaining a concise characterization of a physical environment in the context of frontier-based non-recontaminating exploration was considered. We introduced the medial axis as a configuration space and showed that reasoning about points in this configuration space is equivalent to reasoning about robots in physical space. We formalized the notion of width, corridors and junctions. We introduced an exploration model whereby a swarm locus moving along the skeleton allows us to reason about a group of robots moving through physical space. We defined what it means for a frontier to split and for a group of robots to traverse a junction, and derived lower and upper bounds on the number of robots required to traverse a junction.

Chapter 4

Deployment Algorithm for Non-Recontaminating Exploration

In this chapter[1] we present a series of steps that will transform our continuous configuration space into a symbolic representation of the environment in the discrete domain. We establish a set of rules for navigating the environment in this discrete representation, that allow us to develop an algorithmic pursuit strategy. Using the junction lower bound from Result (3.1) we derive a lower bound on the total number of pursuers necessary to clear the environment, showing that no fewer than this number can possibly clear the environment regardless of the exploration model or pursuit strategy. Using the junction upper bound from Result (3.2) we develop an upper bound on the total number of pursuers that will always be sufficient to clear the environment, for any pursuit strategy. Finally, we derive an optimal pursuit strategy and prove that it guarantees we can clear the environment with the minimum number of pursuers for a given exploration model.

In Section 4.1 we present a transformation from this continuous configuration space into a symbolic representation in the discrete domain. In Section 4.1.1 we establish a set of rules for navigating the environment. In Section 4.1.2 we derive bounds in the number of pursuers required to clear the environment. Finally, in

[1] The majority of this chapter was published in [45].

Section 4.1.3 we present a pursuit-evasion algorithm, and prove that the algorithm is optimal.

4.1 Topology Tree

The most natural representation of the skeleton of an environment where both Q and B are connected is a tree. Since we are also given a starting point on the skeleton, we consider a directed rooted tree (rooted at the start node). We refer to this as the environment *topology tree*, denoted by $T = (V, E)$, where V is the set of vertices (nodes) and E is the set of edges of T. Let $s \in V$ be the root node of T. Nodes correspond to end points and junction points, and edges correspond to corridors connecting these points on the skeleton.

Let $\gamma : V \to \mathbb{N}$ denote the out-degree of a node. There are four types of nodes on the topology tree. The swarm locus starts at an end point on the skeleton which corresponds to the root node s of out-degree $\gamma(s) = 1$. Every other end point corresponds to a leaf node w of out-degree $\gamma(w) = 0$. Each junction point j is represented by a unique junction *entry node* u of out-degree $\gamma(u) = \theta(j) - 1$, connected to an associated set of distinct junction *exit nodes* $\{v_1, v_2, \ldots, v_{\gamma(u)}\}$ of out-degree $\gamma(v) = 1$. (Observe that since T is a directed rooted tree, every node has in-degree 1, except for the root node s which has in-degree 0.)

The root node is connected to some junction entry node, while junction exit nodes are connected to leaf nodes and other junction entry nodes, as determined by the corridors connecting these points on the skeleton. Every edge $e \in E$ on the topology tree is assigned a weight $\alpha : E \to \mathbb{N}$. There are two fundamentally different types of edges: edges that represent a corridor on the skeleton and edges that represent the split frontier at a junction point.

Motivated by Corollary 1, for an edge e connecting node u of out-degree $\gamma(u) = 1$ to a node v, the edge weight $\alpha(e)$ is determined by the number of robots necessary and sufficient to advance from u to v, given by the maximum relief value $\mathcal{R}(x)$ at some point $x \in S$ amongst the points along that corridor.

Motivated by the reasoning in Section 3.2.2, for junction nodes the representation is as follows. For each junction entry node $u \in V$ let $E_u \subset E$ be the set of edges $\{e_1, e_2, \ldots, e_{\gamma(u)}\}$ connecting u to its associated set of junction exit nodes $\{v_1, v_2, \ldots, v_{\gamma(u)}\}$. For each junction, we define a traversal function $\delta : E_u \to \mathbb{N}$, where $\sum_i \delta(e_i)$ is the number of robots required to traverse the junction. This corresponds to the minimum number of robots required to form a split frontier at the junction, given by its ceiling length $\lceil \|\mathcal{F}'_s\| \rceil$. Since we do not have tight bounds on the length of the split frontier, the traversal function depends on the the context of the analysis. Namely, if the goal is to derive a lower bound on the number of robots required to clear an environment, then we consider the length of the split frontier $\lceil \|\mathcal{F}'_{s,min}\| \rceil$ given by Result (3.1). If the goal is to derive an upper bound then we consider the length of the split frontier $\lceil \|\mathcal{F}'_{s,max}\| \rceil$ given by Result (3.2). Each edge e_i is assigned weight $\alpha(e_i) = \delta(e_i)$ which corresponds to the number of robots n_i that are are required to form a frontier $\mathcal{F}'_i$ at each outgoing corridor, given by the ceiling length of each outgoing frontier $\lceil \|\mathcal{F}'_i\| \rceil$.

4.1.1 Exploration Rules

We consider exploration of the topology tree to be a game. We start the game with a single group of n_0 robots stationed at the root node s. Every node $v \in V$ on the topology tree is marked with a label λ, which can have one of three values: CONTAMINATED, EXPLORED and CLEARED. Initially, the root node is marked EXPLORED and all other nodes are marked CONTAMINATED.

We play the game in rounds, each round moving some number of robots from one node to another. If a group of robots is unable to move from one node to another on some round, then the robots are "stuck" at that node. This corresponds to the fact that if there are insufficient robots to clear a corridor, they will remain stuck guarding the corridor, unable to retreat without allowing recontamination. Let $\lambda_k(v)$ denote the labeling of a node $v \in V$ on round k. We win the game if the tree is cleared on some round k_0, that is if $\exists k_0 \in \mathbb{N} \mid \forall k > k_0 \; \forall v \in V, \lambda_k(v) = \text{CLEARED}$. We lose the game if all robots are stuck at some node but the tree has not been cleared by some

round k_0, that is if $\exists k_0 \in \mathbb{N} \mid \forall k > k_0 \; \exists v \in V, \lambda_k(v) \neq \textsf{CLEARED}$.

Robots can split into smaller groups and join to form larger groups. In general, we are free to choose how we move the robots on the topology tree, provided that we obey the following transition rules.

1. If a group of n_0 robots reaches a node u where $\gamma(u) > 0$, then the group splits into some permutation of $\gamma(u)$ groups of n_i robots advancing to each of the children nodes $\{v_1, v_2, \ldots, v_{\gamma(u)}\}$. We are free to choose this permutation, subject to the following restrictions:

   (a) If $\lambda(v_i) = \textsf{CONTAMINATED}$, then $n_i \geq \alpha(e_i(u))$.

   (b) If $\lambda(v_i) = \textsf{EXPLORED}$, then $n_i \geq 0$.

   (c) If $\lambda(v_i) = \textsf{CLEARED}$, then $n_i = 0$.

   If no such permutation exists, then the group remains stuck at node u.
2. If a group of robots is stationed at a node u where $\lambda(u) = \textsf{CLEARED}$, then the group backtracks to the parent node.
3. If a group of robots reaches a node u that is marked CONTAMINATED, then u is marked EXPLORED.
4. If a group of robots reaches a leaf node u or a node u where all children of u are marked CLEARED, then u is also marked CLEARED.
5. If two or more groups of $n_1, n_2, \ldots, n_k$ groups of robots are stationed at the same node, then they form a single group of $n_1 + n_2 + \ldots + n_k$ robots.

The reasoning behind these rules follows from the problem formulation and results in Section 3.2. Rule 1(a) enforces that our exploration is non-recontaminating. Rule 1(b) allows robots to move to explored nodes and join other robots. Rules 3 and 4 define the progression of the game, and Rules 1(c) and 2 ensure that exploration is always progressive (the latter ensures a group of robots leaves a region once it has been cleared, while the former ensures that no group of robots re-enters that region unnecessarily). Rule 5 ensures that robots always act in a single group when stationed at a node.

We understand a state of the topology tree to mean the labeling of each node and the number of robots stationed at each node on a given round. We call a sequence of transitions between states of the topology tree a *pursuit strategy*. (We omit a formal definition for brevity.) A pursuit strategy is like a written record of a game of chess that allows the game to be replayed by carrying out the recorded sequence of transitions. Observe that the only degree of freedom in choosing a pursuit strategy is what to do at a given junction entry node u. We can always choose what junction exit node to send a group of robots to as long as it is not marked CLEARED. If the junction exit nodes are marked CONTAMINATED then the group traverses the junction if and only if $n_0 \geq \sum_i \alpha(e_i(u))$. If the junction is traversed, then the group splits into some permutation of $\gamma(u)$ groups of n_i robots advancing to each of the associated junction exit nodes $\{v_1, v_2, \ldots, v_{\gamma(u)}\}$. We are free to choose this permutation, provided that $\forall i, n_i \geq \alpha(e_i(u))$. The choice we make in selecting this permutation may affect the outcome of the game. (Note also that a node u is only marked CLEARED once the entire subtree $T(u)$ is marked cleared. Thus robots are forced to clear subtrees recursively, and can only backtrack once a given subtree is cleared.)

We also note that because n_0 and $|V|$ are finite, there are a finite number of possible pursuit strategies for a given topology tree. Intuition tells us that if n_0 is too low, every pursuit strategy will be a losing strategy, whereas if n_0 is sufficiently high then any pursuit strategy will be a winning strategy. We now formalize this intuition, and derive lower and upper bounds on the total number of robots that can clear the topology tree.

4.1.2 Environment Bounds

Consider the topology tree T with root node s. Let $T(q)$ be the tree obtained by considering node q as the root node and removing nodes that are not descendants of q. Let $n(T(q))$ be the number of robots required to clear $T(q)$. Let $P(v)$ be the set of nodes $\{p_1, p_2, \ldots, p_{\gamma(v)}\}$ that are children of $v \in V$, enumerated in order of ascending $n(T(p_i))$.

We motivate the lower bound as follows. At each node s we consider whether

more robots are required to advance to the child node p_1 than are required to clear the rest of the subtree $T(p_1)$, and apply this recursively for the entire tree. At each junction entry node we consider the maximum number of robots required to clear a given subtree $T(p_i)$ while guarding the remaining junction exit nodes that have not been cleared. Formally, let $n_{min}(T(s))$ be the total number of robots necessary to clear the environment with topology tree T, given by

$$n_{min}(T(s)) =$$

$$\begin{cases} 0 & \text{if } \gamma(s) = 0 \\ \max\left\{ \sum_{i=1}^{\gamma(s)} \alpha\big(e_i(s)\big),\ \max_{i=1,\ldots,\gamma(s)} \left\{ n_{min}\big(T(p_i)\big) + \sum_{j=i+1}^{\gamma(s)} \alpha\big(e_j(s)\big) \right\} \right\} & \text{otherwise .} \end{cases} \tag{4.1}$$

Lemma 3. *$n_{min}(T(s))$ robots are necessary to clear an environment with topology tree T, regardless of exploration model or pursuit strategy.*

Proof. We begin at the root node s, which has out-degree $\gamma(s) = 1$. If $\alpha(e_1(s)) > n(T(p_1))$, i.e. if more robots are required to advance to the child node $p_1 \in P(s)$ than to clear the rest of the tree $T(p_1)$, then $n_{min}(T(s)) = \alpha(e_1(s))$. (The second term in the outer max expression is not evaluated if $\gamma(s) = 1$.) This applies recursively for any node of out-degree $\gamma(s) = 1$. Leaf nodes provide the recursion base case, where $n(T(w)) = 0$ for $w \in V$ if $\gamma(w) = 0$.

For junctions the logic is as follows. For a junction entry node $u \in V$, each associated junction exit node $p_i \in P(u)$ is the root node of a subtree $T(p_i)$. By Rules 1 and 2, a group of robots that reaches a leaf node w will backtrack until it reaches a node with previously unexplored children. By induction a group of robots that clears $T(p_i)$ will backtrack until it returns to v and advance to a different junction exit node $p_{j\neq i}$. By Rule 5, this group of robots will join with any other group of robots stationed at p_j. When a group of robots reaches a junction, at least $n_0 > \sum_i \alpha(e_i(u))$

robots are required to traverse it thus stationing $n_i = \alpha(e_i(u))$ robots at the $\gamma(u)$ junction exit nodes p_i. Thereafter n_0 must be at least enough to clear the subtree with smallest $n(T(p_i))$ while holding station at the remaining $\gamma(u) - 1$ exit nodes. This subtree is $T(p_1)$ since $p_i \in P(u)$ are enumerated in order of ascending $n(T(p_i))$. If n_0 robots is not enough to clear $T(p_1)$, then it will certainly not be enough to clear any other subtree, and thus the junction cannot be traversed. If n_0 robots is enough to clear $T(p_1)$ then $n(T(p_1))$ robots are now able to join one of the groups guarding the remaining $\gamma(u) - 1$ junction exit nodes. We apply this iteratively for all subtrees $T(p_i)$, each time gaining the services of the group that cleared the previous subtree. The final subtree simply requires $n(T(p_{\gamma(u)}))$ robots, since all other subtrees have been cleared and do not require any robots to guard the junction exit nodes. (Note that all n_i groups of robots will actually clear their subtrees simultaneously, in which case two given groups may not join at the junction exit nodes, but elsewhere along a given subtree. However, considering each group to clear its subtree in stages with the other groups guarding the junction exit nodes simplifies the abstraction.) We now consider the maximum of the number of robots required to traverse the junction (the first term in the outer max expression) and the maximum number of robots required to clear a given subtree $T(p_i)$ and guard the remaining $\gamma(u) - i$ junction exit nodes (the second term in the outer max expression, itself a maximum over $\gamma(u)$ stages). The maximum of these two outer terms is the minimum number of robots required to clear the subtree $T(u)$. We apply this logic recursively for all junctions.

Thus no fewer than $n_{min}(T(s))$ robots can clear an environment with topology tree T, for a given exploration model. Using the junction lower bound from Result (3.1) to obtain the traversal function δ_{min} for each junction, we know that no fewer than $\sum_i \delta_{min}(e_i(u))$ robots can traverse the junction point j corresponding to the junction entry node u. Thus, using $\alpha(e_i(u)) = \delta_{min}(e_i(u))$ for each junction entry point u, $n_{min}(T(s))$ gives a lower bound on the number of robots that is necessary to clear an environment with topology tree T, regardless of exploration model or pursuit strategy. □ □

We motivate the upper bound as follows. We imagine an adversary that dictates

the pursuit strategy of a number of robots, with the goal of placing the maximum number of them on the topology tree T, while preventing T from being cleared. Then we argue that given any such adversarial configuration of $n^\star(T(s))$ robots, $n^\star(T(s))+1$ robots will always be able to clear T, regardless of the pursuit strategy chosen by the adversary. Formally, let $n_{max}(T(s))$ be the total number of robots sufficient to clear the environment with topology tree T, given by

$$n_{max}(T(s)) = n^\star(T(s)) + 1 \ ,$$

$$n^\star(T(s)) = \begin{cases} 0 & \text{if } \gamma(s) = 0 \\ \max\left\{ \left(\sum_{i=1}^{\gamma(s)} \alpha\big(e_i(s)\big) \right) - 1 \ , \ \sum_{i=1}^{\gamma(s)} n^\star\big(T(p_i)\big) \right\} & \text{otherwise .} \end{cases} \tag{4.2}$$

Lemma 4. *$n_{max}(T(s))$ robots are sufficient to clear an environment with topology tree T, for a given exploration model, regardless of pursuit strategy.*

Proof. The adversary begins at the root node s, which has out-degree $\gamma(s) = 1$. If $\alpha(e_1(s)) > n(T(p_1))$, i.e. if more robots are required to advance to the child node $p_1 \in P(s)$ than to clear the rest of the tree $T(p_1)$, then $n^\star(T(s))$ simply equals $\alpha(e_1(s)) - 1$ since adding one more robot will result in the entire tree being cleared. This applies recursively for any node of out-degree $\gamma(s) = 1$. Leaf nodes provide the recursion base case, where $n(T(w)) = 0$ for $w \in V$ if $\gamma(w) = 0$.

For junctions the logic is as follows. For a junction entry node $u \in V$, each associated junction exit node $p_i \in P(u)$ is the root node of a subtree $T(p_i)$. The adversary can choose one of two options: either place $\left(\sum_{i=1}^{\gamma(s)} \alpha(e_i(s)) \right) - 1$ robots at u such that the junction cannot be traversed, or traverse the current junction with $\sum_{i=1}^{\gamma(s)} n^\star\big(T(p_i)\big)$ robots knowing that they will not be able to clear the rest of the tree. The adversary chooses the maximum of these two values since she is trying to maximize the number of robots on the tree. We apply this logic recursively for all junctions.

Consider any such configuration of $n^\star(T(s))$ robots. We now introduce one addi-

tional robot at the root node s. The robot must navigate the tree according to the exploration rules, but the adversary is still free to choose its pursuit strategy. By Rule 5, whenever the robot reaches any node u with $n^\star(T(u))$ robots stationed at the node, a single group of $n^\star(T(u)) + 1$ robots forms at u which is sufficient to clear the subtree $T(u)$. By Rule 2, any group of robots that clears a subtree will backtrack along the tree until it reaches a previously unexplored part of the tree. This group of robots will join other groups of $n^\star(T(v))$ robots similarly stationed at other nodes $v \in V$. This will continue recursively until the entire tree is cleared. Thus placing an additional robot at s causes a "chain reaction" that results in T being cleared regardless of the pursuit strategy that the adversary chooses for any of the $n^\star(T(s))+1$ robots.

Thus no more than $n_{max}(T(s)) = n^\star(T(s)) + 1$ robots will ever be required to clear an environment with topology tree T, regardless of pursuit strategy. Using the junction upper bound from Result (3.2) to obtain the traversal function δ_{max} for each junction, we know that no more than $\delta_{max}(e_i(u))$ robots are required to traverse the junction point j corresponding to the junction entry node u for the given exploration model. Thus, using $\alpha(e_i(u)) = \delta_{max}(e_i(u))$ for each junction entry point u, $n_{max}(T(s))$ gives an upper bound on the number of robots that is sufficient to clear an environment with topology tree T, for a given exploration model, regardless of pursuit strategy. □ □

4.1.3 Optimal Pursuit Strategy

We now present an optimal pursuit strategy that guarantees that the environment is cleared with the minimum number of robots for a given exploration model. Consider the topology tree T with root node s. We know that $n_{min}(T(s))$ robots are necessary to clear T, given by Result (4.1). The following algorithm guarantees that T will be cleared with $n_{min}(T(s))$ robots. (We use the same notation as in Section 4.1.2.)

Lemma 5. *$n_{min}(T(s))$ robots are necessary and sufficient to clear an environment*

Algorithm 1 `Clear` Algorithm

Given n_0 robots located at root node s of topology tree $T = (V, E)$:

1. If s is a leaf node or if all children of s are marked CLEARED, then:
   (a) mark $s \leftarrow$ CLEARED.
   (b) Backtrack to parent node. If parent node does not exist, terminate.
2. If $\lambda(s) =$ CONTAMINATED, mark $s \leftarrow$ EXPLORED.
3. If $\gamma(s) = 1$, `Clear`$\big(\ T(p_1),\ n_0\ \big)$.
4. If $\gamma(s) > 1$ and if all children of s are marked CONTAMINATED, then:
   (a) for $i = 2, \ldots, \gamma(s)$: `Clear`$\big(\ T(p_i),\ \alpha(e_i(s))\ \big)$.
   (b) `Clear`$\big(\ T(p_1),\ n0 - \sum_{i=2}^{\gamma(s)} \alpha(e_i(s))\ \big)$.
5. If $\gamma(s) > 1$ and if all children of s are not marked CONTAMINATED, then:
   (a) let $i = \min_{2,\ldots,\gamma(s)} \big\{ i \mid \lambda(p_i) \neq$ CLEARED$\big\}$.
   (b) `Clear`$\big(\ T(p_i),\ n_0\ \big)$.

with topology tree T, for a given exploration model.

Proof. Given $n_{min}(T(s))$ robots we prove that we can use the `Clear` algorithm to clear the environment. Let $n_0 = n_{min}(T(s))$. We begin at the root node s, which has out-degree $\gamma(s) = 1$. A group of robots can always advance to the child node p_1 since we have $n_0 = n_{min}(T(s))$ robots. This applies for all nodes of out-degree $\gamma(s) = 1$. When a group of robots reaches a leaf node it is marked CLEARED. A group of robots stationed at a node that is marked CLEARED will backtrack until it reaches a node with children that are not marked CLEARED.

For junctions, the logic is as follows. The first time a group of robots reaches a given junction entry node u, all children of u are marked CONTAMINATED. We send $n_i = \alpha\big(e_i(s)\big)$ robots to each of the $\gamma(s) - 1$ junction exit nodes p_i, and use the remaining $n_1 = n0 - \sum_{i=2+1}^{\gamma(s)} \alpha\big(e_i(s)\big)$ robots to clear the subtree $T(p_1)$. By proof to Lemma 3, we know that the entire subtree $T(p_1)$ can be cleared with n_1 robots, and therefore n_1 robots will eventually backtrack to u. Thereafter, all children of u are either EXPLORED or CLEARED. Each group of robots reaching u in this way is sent to clear $T(p_i)$ where i is the smallest number such that $\lambda(i) \neq$ CLEARED. We do this iteratively for each subtree, each time gaining the services of the group that

cleared the previous subtree. We can always clear each subtree $T(p_i)$ in this manner because $\{p_1, p_2, \ldots, p_{\gamma(v)}\}$ are enumerated in order of ascending $n(T(p_i))$, and by proof to Lemma 3 we know that $n(T(u))$ robots is enough to clear every subtree in this way. For each subtree $T(p_i)$ cleared in this way, p_i is marked CLEARED. When all subtrees have been cleared, u will be marked CLEARED, and the entire group of robots will backtrack to the parent node. This applies recursively recursively for all junctions. □ □

Using the junction upper bound from Result (3.2) to obtain the traversal function δ_{max} for each junction, we know that no more than $\delta_{max}(e_i(u))$ robots are required to traverse the junction point j corresponding to the junction entry node u for the given exploration model. Thus, using $\alpha(e_i(u)) = \delta_{max}(e_i(u))$ for each junction entry point u, the `Clear` algorithm gives an optimal pursuit strategy for clearing an environment with topology tree T, for a given exploration model.

4.2 Implementation

Numerical simulations were carried out in MATLAB to verify the correctness and guarantees of the optimal pursuit strategy. In this section we aim to elucidate the operation of the algorithm by presenting a example case pursuit evasion deployment scenario.

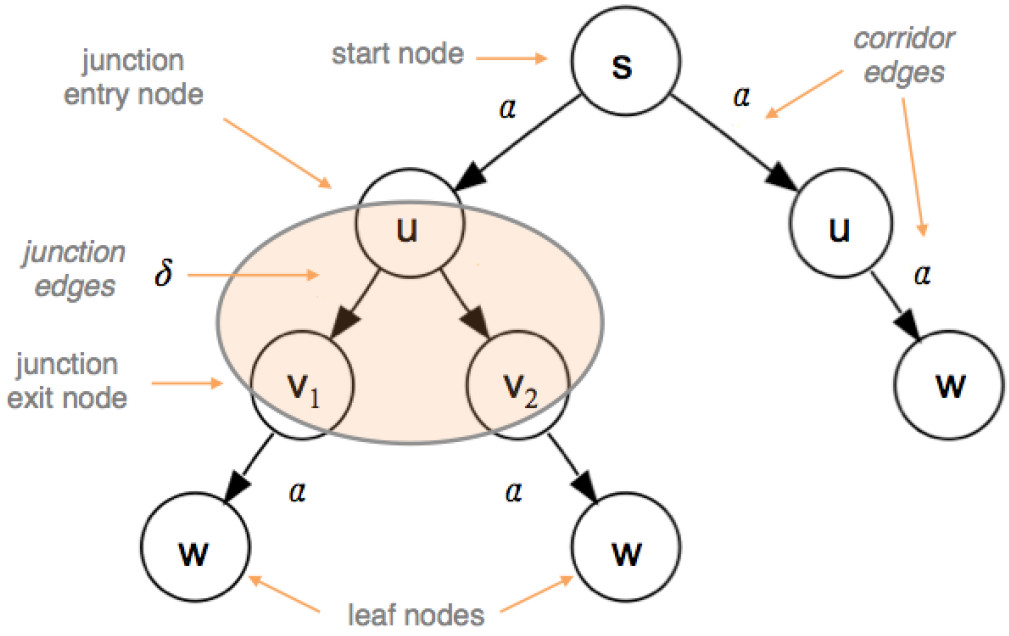
It is clear from Figure 3-7 that environments of modest complexity may have a lot of endpoints and branch points. In-fact, it can be shown that the degree to which the skeleton fractalizes is a function of how "smooth" the wall contours are. Having a lot of branching in a topology tree does not necessarily imply that many more robots will be required to explore the corresponding environment, since a lot of the branching be reveled to be redundant when considered in the context of the sensing radius of the robots. Furthermore, the optimal pursuit strategy presented in Algorithm 1 is highly recursive by nature. For these reasons it is neither essential nor helpful to consider such an expansive tree for our example case scenario. We will consider a minimal subtree that is part of a larger topology tree, and is illustrative of the

algorithm as a whole. We show how we can numerically evaluate a pursuit evasion deployment scenario on this subtree, and convince the reader that the correctness extends recursively to our deployment strategy on any scale.

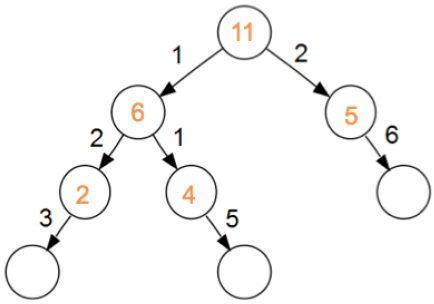
The minimal subtree is shown in Figure 4-1a. Note that there is not much structural variation possible in a topology tree, and this example is enough to cover the different types of nodes and edges. The topology tree is further characterized by the edge weights *alpha* indicating the number of robots required to traverse the junction to reach the junction exit nodes. Figures 4-1b and 4-1c show an example weighting.

We first consider the environment upper bound policy as described in Section 4.1.2, i.e. a suboptimal policy. The recursive logic is to compute the maximum of either one less than the number of robots required to reach a given junction node or the recursive sum of the number of robots required to clear the subtree of each child node. Observing the edge weights leading to the leaf nodes, we assign one less than that number to the parent nodes of the leaves, in accordance with the upper bound proof. This yields 6 robots at the left junction entry node as the sum of v_1 and v_2 and 5 robots at the right junction entry node. Recursively this yields 11 robots at s, and hence 12 robots will clear the environment by Lemma 4. The action of the robots is irrelevant, as the lemma guarantees that they will clear the environment.

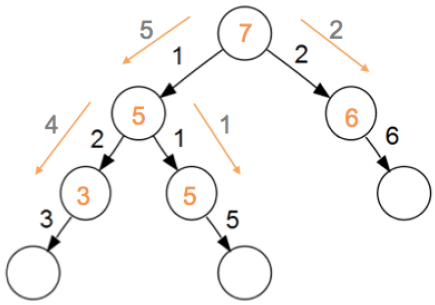
Let us now consider our optimal deployment strategy under this example case scenario as described in Section 4.1.3. By Lemma 5 we compute that 7 robots is necessary and sufficient to clear the environment. The robots begin at the start node and proceed as follows: 2 robots advance to the right junction entry node and the other 5 robots advance to the left junction entry node, in accordance with the fact that the left subtree requires less total robots to clear than the right subtree. At the left junction entry node, the 5 robots split according to the same recursive logic – 1 robot advances to v_2 and the remaining 4 robots clear the subtree with root node v_1. Once the recursive steps are complete, the algorithm unwinds. 4 robots return from v_1 and join with the 1 robot at v_2, clearing the subtree with root node v_1; and then all 5 robots return to s and join with the remaining 2 robots to clear the rest of the tree.



(a) Subtree used in our deployment case scenario. The diagram shows the possible types of nodes and edges. The junction is highlighted by the oval.



(b) Example weight assignment for the subtree, showing the associated α. The numbers in orange show the upper bound on the number of pursuers necessary to clear the subtree.



(c) Optimal pursuit strategy that clears the environment with the minimum number of robots. The numbers in orange show the recursive solution for optimal pursuit strategy.

Figure 4-1: Minimal subtree that illustrates the mechanics of the optimal pursuit strategy. Such a subtree would typically be a small part of a large tree that represents a real-world environment.

In this way we can recursively compute and simulate the deployment of robots in any large-scale environment. A numerical simulator was developed in MATLAB which computes the suboptimal upper bound strategy as well as the optimal pursuit

strategy. This allows us to compute an optimal deployment strategy for any topology tree representing an arbitrary continuous environment.

4.3 Summary

In this chapter we presented a transformation from a continuous configuration space representation of an environment into a symbolic representation in the discrete domain. We cast the exploration problem as a game, established rules for playing this game, and derived bounds on the number of robots necessary and sufficient to clear the environment. Finally we presented an optimal pursuit strategy that guarantees that we can clear the environment with the minimum number of robots. Finally, we discussed implementation and presented a numerical simulation of an example case deployment scenario.

Chapter 5

Markov-based Model for Future Urban Mobility Networks

In this chapter[1] we extend on the idea of modeling exploration deployment on a graph to the more general case of patrolling. For our application scenario, we consider optimization of an urban transportation network. Understanding how to optimize transportation is critical for urban planning.

We leverage data from a fleet of 16,000 taxis in Singapore to create a Markov-based urban transportation model that realistically describes the operation of a fleet of service agents (taxis) in response to incident requests (arriving customers) in the city. We establish a theoretical Markov-based framework that describes an urban transportation network. We assume that we have a road network with discrete pickup and drop-off locations corresponding to designated points in the city. The arrival rates of customers at each location are known. A vehicle with a person on board will drive to the customer's goal destination. If a customer is waiting at this location the vehicle picks up the customer; if there are no waiting customers, the vehicle goes to a different location according to a redistribution policy.

Our goal is to compute a solution in the form of the required number of vehicles in the system and their redistribution policy. We encode the solution as a scalable

[1] The majority of this chapter was published in [44].

optimization problem. We develop a mechanism for this model by which a group of agents can patrol the graph to persistently service these requests.

This chapter is organized as follows. Section 5.1 states the problem and presents the model formally. Section5.2 outlines the format of the solution and sets up the optimization problem. Section 5.3 describes the challenges in creating a realistic urban mobility model and presents the detailed steps to do so.

5.1 Problem Statement

We consider a pickup and delivery problem (PDP) on an undirected graph. There are m nodes in the network, subject to incident request arrivals. The graph is patrolled by n mobile agents (taxis) that traverse it along its edges and service requests (customers) as they arrive. Requests arrive according to a Poisson process with an arrival rate of λ requests per time unit τ and are distributed among nodes according to an arrival distribution $\alpha = [\alpha_1\ \alpha_2\ \ldots\ \alpha_m]$. Thus a request will arrive at each node with an arrival rate of $\lambda\alpha_i$. The destination of incident requests is determined by a *request transition matrix* D, where $d_{i,j}$ is the probability that a request arriving at node i is destined for node j. Since each row of D is a probability distribution over a node, we require that the rows sum to 1, i.e. D is a Markov chain.

When a vehicle arrives at a node and encounters a request it must service that request and when a vehicle delivers a request to its destination node it is immediately available to service new requests. A vehicle that does not encounter a request transitions according to a *redistribution policy* transition matrix P. Each row of P is a probability distribution over a node, i.e. P is also a Markov chain.

We consider the system to evolve according to a single *system transition matrix* S so that for $X \sim h_i$

$$s_{i,j} = \Pr(X_{k+1} = j \mid X_k = i). \tag{5.1}$$

Thus the system evolves according to D when a request is being serviced and according to P when a request is not being serviced. Denoting by β_i the probability that a

vehicle leaving node i is servicing a request, we express (5.1) as

$$s_{i,j} = \beta_i \Pr(X_{k+1} = j | X_k = i) + (1 - \beta_i) \Pr(X_{k+1} = j | X_k = i). \tag{5.2}$$

We introduce the $m \times m$ matrix $B = diag([\beta_1 \ \beta_2 \ \dots \ \beta_m])$ so that (5.2) can now be expressed in matrix form as

$$S = BD + (I_m - B) P \tag{5.3}$$

where I_m is the identity matrix of size m.

The stationary distribution of a Markov chain P is a vector q such that $qP = q$. For convenience we define the function

$$\pi : P \longmapsto q \mid qP = q. \tag{5.4}$$

In steady state the system will exhibit a stationary distribution $\phi = \pi(S)$ of agents among nodes. Given a number of vehicles n and a stationary distribution ϕ, in steady state we expect $n\phi_i$ vehicles to arrive at node i at a given time, and to find each vehicle located at node i with probability ϕ_i.

5.2 Optimization Setup

Informally, we want to ensure stability in steady state. We understand system stability to mean the condition whereby the steady state service rate at each node in the system exceeds the steady state arrival rate at that node. The solution space is the number of taxis n and the redistribution policy P. The objective of the problem is therefore to find a solution (i.e. determine the number taxis n and a policy P) such that the overall system is stable.

We formalize the problem as follows.

$$\begin{aligned} \text{Find} \quad & n, P \\ \text{s.t.} \quad & n\phi_i > \lambda\alpha_i, \ \forall i && (5.5) \\ & n \leq n_{max} && (5.6) \\ & 0 \leq p_{i,j} \leq 1, \ \forall i && (5.7) \\ & \sum\nolimits_j p_{i,j} = 1, \ \forall i. && (5.8) \end{aligned}$$

The first constraint (stability constraint) states that the service rate at each node must be greater than the arrival rate at each node. The second constraint states that the solution space is physically bounded by some maximum number of taxis. The last two constraints ensure that P is a valid Markov chain.

5.2.1 Hastings-Metropolis Algorithm

The solution space of the optimization problem is the number of taxis n and the redistribution policy P, while the stability constraint of the problem is specified as a function of the stationary distribution of the system transition matrix $\phi = \pi(S)$. This presents a computational challenge since the transformation from a Markov chain to its stationary distribution is non-linear. Thus it is infeasible to consider linear programming methods to find P directly.

Instead we propose a different approach, employing the Hastings-Metropolis algorithm [18]. The HM algorithm is a Markov chain Monte Carlo method that, given a stationary distribution, can be used to construct a Markov chain with that stationary distribution. Using this method we may simplify a potential optimization problem to that of finding a desired stationary distribution and generating the policy using HM. For convenience we denote the HM algorithm as

$$H : q \longmapsto P \mid qP = q. \tag{5.9}$$

However, using the HM algorithm poses another challenge: by solving for the

stationary distribution, we cannot enforce the zero constraints of the target redistribution policy matrix (in other words we cannot enforce sparsity), which implies an underlying undirected clique network. In the following section, we show that an undirected clique model is highly restrictive in terms of the kind of transportation network that it can describe. We present a model for a real urban mobility network, and discuss the steps taken to ensure that the model is realistic while still complying with the Markov framework. We then present a mechanism by which
the HM optimization can be set up to handle a sparse network while maintaining a computational complexity that is independent of the degree of precision of the model.

5.3 Modeling Urban Mobility

In this study we use transportation data from Singapore. The dataset is one month (August 2010) of taxi data from a fleet of 16,000 taxis. This data amounts to approximately 500 million data points at 42,000 GPS locations in Singapore. Each record contains the taxi and driver ID, time stamp, GPS coordinates, and status of operation. We partitioned the space of nodes from the Singapore taxi dataset with a k-means clustering into 27 regions (Fig. 5-1). The clustering was based on previous work, and the number of clusters chosen such that the clustering aligns well with the postal regions of Singapore. We also derived an extensive set of statistics for these regions from the Singapore taxi data.

5.3.1 Extended Network

A true Markov chain is a discrete probabilistic state machine, meaning that each transition occurs in one time step. This presents a challenge, as we can assign a real travel time to correspond to each transition, but in the current formulation there is no way to enforce varying travel times. We present an extended framework that captures this information. First, we derive a *base network* that considers each cluster as an individual *base node*. The base network is an undirected clique graph $G = (V, E)$ of size m. The Singapore dataset was used to calculate the average travel time $t_{i,j}$

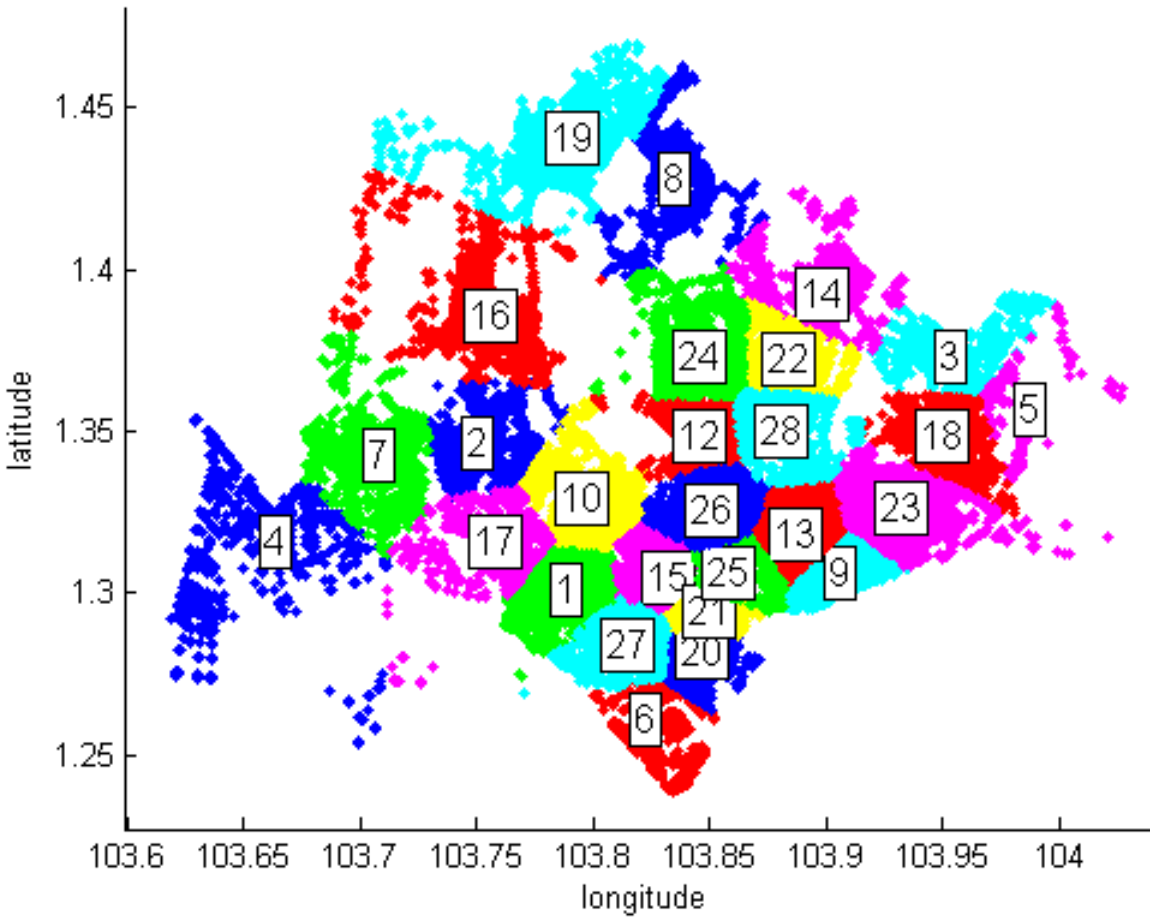


Figure 5-1: The 42,000 Singapore nodes used in this study, color-coded according to their k-means clustering. (The computed clusters align well with postal regions in Singapore.)

from cluster i to cluster j. Trips within clusters are also considered, so there is no requirement that $i \neq j$. Also note that in general $t_{i,j} \neq t_{j,i}$, which reflects traffic inhomogeneity caused by congestion throughout the day. A discretization parameter τ specifies the the shortest travel time between 2 nodes represented by a single transition in the extended model. A travel time matrix T encodes the discretized travel times, where $\tau_{i,j} = \max\{\text{round}(t_{i,j}/\tau), 1\}$.

We use this information to derive the *extended network* G'. Starting with the base network G, for each pair of nodes $i, j \in V$ we remove the edge connecting i and j by setting $G'_{i,j} = 0$ and create two *auxiliary connections* between i and j, one for each direction of travel. Each such auxiliary connection consists of $\ell = \tau_{i,j} - 1$ *auxiliary nodes* $x_1,\ x_2,\ \ldots\ x_\ell$. New edges are added by daisy-chaining node i to node j through the ℓ auxiliary nodes by setting $G'_{i,x_1} = G'_{x_1,x_2} = \ldots = G'_{x_\ell,j} = 1$. We

refer to the first auxiliary node x_1 as the *proxy node* of i to j, denoted by $\theta(i,j)$. If $\tau_{i,j} = 1$, node i is simply connected to node j. Finally, for $i = j$, a single auxiliary connection is created in this way, representing the average travel time within cluster i. The resulting extended network is a sparse network $G' = (V', E')$ of size m'.

We make the assumption that customers arrive only at and travel to base nodes and are not picked up on the side of the road. Although in reality taxi drivers may divert from any prescribed policy, this is a reasonable model corresponding to the intuition of base nodes representing taxi stations for example. As a result, the extended arrival distribution is simply given by $\alpha' = [\alpha_1\ \alpha_2\ \ldots\ \alpha_m\ 0\ \ldots\ 0]$. This greatly simplifies the optimization problem since we do not need to satisfy stability constraints at auxiliary nodes.

5.3.2 Extended Redistribution Policy

Given an $m \times m$ redistribution policy P for the base network, we extend this policy to an $m' \times m'$ policy matrix P' that incorporates the transitional logic of P while still maintaining the mathematical properties of a Markov chain. We derive P' as follows. For each $p_{i,j}$ corresponding to a connection $g_{i,j} = 1$ in the base network, there is a corresponding auxiliary connection consisting of a set of $\ell = \tau_{i,j} - 1$ auxiliary nodes $G'_{i,x_1} = G'_{x_1,x_2} = \ldots = G'_{x_\ell,j} = 1$ in the extended network. We set $p'_{i,\theta(i,j)} = p_{i,j}$, i.e. the proxy node of i to j serves to ensure that the probability of transition from i eventually leading to j remains the same. We set $p'_{x_1,x_2} = \ldots = p'_{x_\ell,j} = 1$, i.e. once the taxi is en route from node i to j (corresponding to a single discrete transition in the base network), it will arrive at node j with probability 1 in exactly $\tau_{i,j}$ transitions. For convenience we denote this transformation from a base transition matrix to an extended transition matrix by $P' = extend(P)$. (The same transformation also applies to the customer transition matrix D.) The following example illustrates extending a

simple base network with $m = 2$ to 3 nodes.

$$T = \begin{bmatrix} 1 & 1 \\ \mathbf{2} & 1 \end{bmatrix},\ P = \begin{bmatrix} 0.5 & 0.5 \\ \mathbf{0.2} & 0.8 \end{bmatrix},\ P' = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.8 & \mathbf{0.2} \\ \mathbf{1} & 0 & 0 \end{bmatrix}.$$

In this example the discretized travel time from node $i = 2$ to node $j = 1$ is $\tau_{2,1} = 2$. A single auxiliary node is created corresponding to $\tau_{2,1} - 1 = 1$. In extending a policy P to P' we maintain the transition probabilities from every node i to the corresponding eventual destination node j in the base network as transitions to the proxy node $\theta(i, j)$ of i to j. So, since $\tau_{2,1} > 1$, we set $p'_{2,1} = 0$ and set $p'_{2,3} = p_{2,1}$ instead. Once a taxi has made the transition to node 3, it transitions to node 1 with probability 1.

5.3.3 Extended Stability Condition

Extending the network by incorporating travel times yields an extended system transition matrix S'. However the associated stationary distribution $\phi' = \pi(S')$ does not relate linearly to ϕ, and depends on the varying ingoing travel times from each node $j \neq i$ to i. Thus a stable policy for the base network may not be stable for the extended network. To ensure that the stability condition is correctly translated, we introduce a scaling vector ζ that accounts for the travel times between nodes in the extended network by considering transitions through auxiliary nodes en route to node i as counting towards the fraction of time that the vehicle spends at node i in steady state. We denote by $Z \in V'$ the set of auxiliary nodes that lead to base node i. Then ζ is given by:

$$\zeta_i = \frac{\phi'_i}{\phi'_i + \sum_k \phi'_{Z_k,i}}. \tag{5.10}$$

Then the stationary distribution of the base network ϕ is given by $\phi = [\phi'_1/\zeta_1\ \phi'_2/\zeta_2\ \ldots\ \phi'_m/\zeta_m]$, and the stability constraint (5) becomes

$$n\zeta_i\phi_i > \lambda\alpha_i,\ \forall i \tag{5.11}$$

5.4 Summary

In this chapter we presented a Markov-based model for patrolling on a general graph. We presented the model formally and develop a mechanism for this model by which a group of agents can patrol the graph to persistently service these requests. We established our solution objective in the form of the required number of agents in the system and their patrolling policy. We derived a mechanism for encoding a patrolling policy in the network as a solution to a scalable optimization problem.

As an application scenario we modeled an urban transportation network that captures the operation of a fleet of taxis in a city. We leveraged data from a fleet of 16,000 taxis in Singapore to construct our model, discussed the challenges and described the steps taken to ensure that the model is realistic while still complying with a true Markov framework.

Chapter 6

Deployment Algorithm for Patrolling an Urban Mobility Network

In this chapter[1] we present a deployment algorithm for patrolling a Markov-based graph model, and demonstrate its application to the urban mobility scenario presented in Chapter 5. We consider the solution with respect to three seemingly different optimization criteria. The first criterion considers the customers, whose end goal is to minimize the time spent waiting for a taxi. The second criterion considers the urban planning authority whose goal is to minimize the number of vehicles in the road network. The third criterion considers the cost and environmental implications of fuel consumption. We leverage data from a fleet of 16,000 taxis in Singapore to show how we can learn and interpret the current default behavior of taxi drivers within our framework, and prove that the current behavior is sub-optimal with respect to the evaluation metrics.

We evaluate our practical patrolling policy through simulation and show that our proposed policy is stable and improves substantially upon the default unmanaged redistribution of taxi drivers in Singapore with respect to the three evaluation criteria.

[1] The majority of this chapter was published in [44].

This chapter is organized as follows. Section 6.1 formulates the optimization problem and provides an algorithm for a practical redistribution policy. We address stability of the policy and discuss the trade-offs between complexity and accuracy. Section 6.2 discusses the optimization criteria and provides metrics for evaluating policies accordingly. Finally, Section 6.3 describes experiments using our patrolling policy and discusses the results.

6.1 Practical HM Policy

The solution space of the optimization problem is the number of taxis n and the redistribution policy P. To simplify the solution, we fix n constant. This eliminates variable product terms in the stability constraint and allows us to formulate the problem as a linear program. Since n is discreet we solve successive linear programs for increasing values of n until a feasible solution is found and the system is stable. Stability is determined experimentally by running a series of bootstrap simulations to check if the request queues remain bounded. We denote by n_{min} the minimum number of taxis that admits a feasible solution that is stable. We set up the linear program as follows:

$$\begin{aligned}
\text{Minimize} \quad & 0 \text{ (i.e. find } q\text{)} \\
\text{s.t.} \quad & n\zeta_i q_i > \lambda\alpha_i, \ \forall i && (6.1)\\
& 0 \leq q_i \leq 1, \ \forall i && (6.2)\\
& \sum\nolimits_i q_i = 1 && (6.3)\\
& n \geq n_{min}. && (6.4)
\end{aligned}$$

Algorithm 2 describes the procedure for calculating the practical HM redistribution policy P_{HM}. There is no cost function to minimize n, i.e. the optimization problem reduces to a search problem to find a stationary distribution q that satisfies the stability condition. The stability condition is evaluated by running a number of bootstrap simulations with redistribution policy P_{HM} and checking the stability of the sys-

Algorithm 2 Practical HM Policy Optimization Algorithm

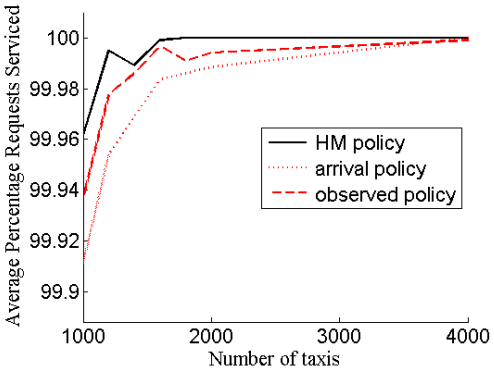
Data: n: proposed number of taxis
λ: customer arrival rate
α: customer arrival distribution
D: customer transition matrix
ζ: stationary distribution scaling vector
n_{min}: minimum required number of taxis (optional)
Result: P_{HM}: redistribution policy

find q s.t. $n\zeta_i q_i > \lambda\alpha_i,\ \forall i$
$P := H(q)$
$P' := extend(P)$
$\phi' := \pi(S')$ {simulation}
$\phi := [\phi'_1/\zeta_1\ \phi'_2/\zeta_2\ \ldots\ \phi'_m/\zeta_m]$
if $\exists\ n_{min},\ n \geq n_{min}$ **then**
  $P_{\mathsf{HM}} := P$
else
  if $n\zeta_i\phi_i > \lambda\alpha_i,\ \forall i$ **then**
    $n_{min} := n$
  else
    $n := n + 1$
  end if
  restart 1
end if
return P_{HM}

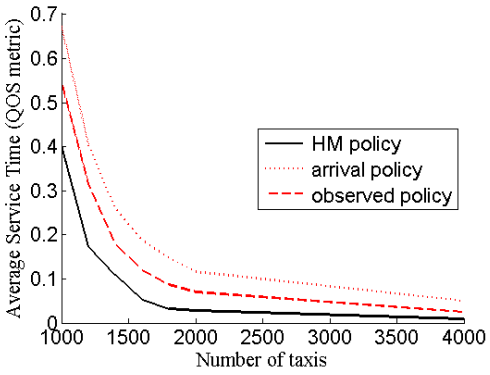
tem transition matrix S' at the end of the simulation, as dictated by $n\zeta_i\phi_i > \lambda\alpha_i,\ \forall i$. n_{min} is then given by the smallest n that yields experimental stability for S'. The practical HM policy is obtained by applying the Hastings-Metropolis transformation to q giving $P_{\mathsf{HM}} = H(q)$.

6.1.1 Accuracy and Complexity

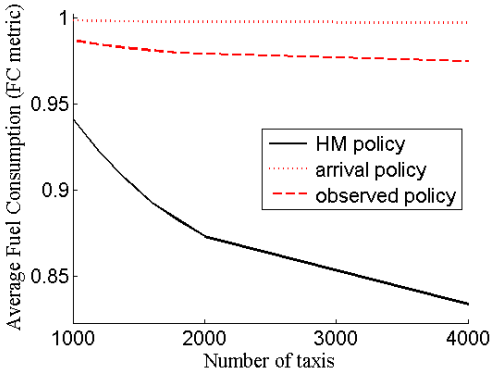
The extended network is parametrized by the travel time discretization τ. This presents a trade-off between two degrees of accuracy in our urban mobility model. For a base network of size m, this determines the number of auxiliary nodes that will be added to the extended network. Assuming an average travel time of t_0 between two clusters, each base node inherits $2(m-1)$ auxiliary connections with $\tau_0 = \max\{\text{round}(t_0/\tau), 1\}$ auxiliary nodes connecting it to other base nodes and one



(a) Experimental Stability



(b) Quality of Service



(c) Fuel Consumption

Figure 6-1: Simulation Results. Fig. 6-1a shows the percentage of requests serviced for each policy with increasing n. Fig. 6-1b shows the decrease in service time for each policy with increasing n (QOS metric). Fig. 6-1c shows the improvement in fuel consumption for each policy with increasing n (FC metric).

---

self-loop with τ_0 auxiliary nodes. This yields a total of $m\left[2\left(m-1\right)+1\right]\tau_0 = O(m^2)\tau_0$ auxiliary nodes. Thus a smaller discretization τ means a larger ratio τ_0, which increases the size of the extended network as $O(m^2)$.

A smaller τ means a more accurate representation of the road network. However, the increase in accuracy comes at the cost of generality. Since customers arrive only at base nodes, a greater increase in the number of nodes in the extended network means a coarser granularity of customer origins and destinations. Conversely, for a given extended network size m', a larger base network size m and smaller travel time discretization ratio means a larger fraction of the m' nodes are considered as origins and destinations.

Finally, as discussed in Section 5.3.1, restricting arrivals to base nodes allows us to formulate the search problem for the base network and transform the result to the extended network. Thus the complexity of the linear program depends only on m and there is no penalty incurred with finer travel time discretization. This means that for a given base network size, we can achieve arbitrary travel time granularity for a fixed computational cost.

6.2 Policy Criteria

We are interested in examining feasible solutions from three independent objectives: urban planning (UP), quality of service (QOS), and fuel consumption (FC). In order to meaningfully evaluate a proposed solution, we establish metrics that correspond to the three criteria under consideration.

Urban Planning

We assume that the main goal of the municipal authority is to reduce congestion in the city by minimizing the number of vehicles on the streets. From the UP perspective, the redistribution policy is irrelevant: the objective is simply to employ the minimum number of taxis that yields a valid solution. We define the UP metric as

$$L_{\mathsf{UP}} = n/n_{min} \tag{6.5}$$

i.e. the metric expresses the degree by which more taxis are employed in a given solution than is strictly necessary for stability. Note that $L_{\mathsf{UP}} \in [1, \infty)$, and $L_{\mathsf{UP}} = 1$ when $n = n_{min}$, i.e. the ideal UP policy is any solution that uses n_{min} taxis.

Quality of Service

Since the stability condition is satisfied for feasible solutions, and all customers in the system are being serviced, we reason that the net revenue from all customers is constant, regardless of the redistribution policy P. Thus we assume that the main

incentive of the taxi company is to provide the best possible service quality. QOS is measured as the average service time (the queueing time plus the time waiting for a taxi) for a customer in the system. We define the QOS metric as

$$L_{\mathsf{QOS}} = \sum_i \alpha_i R_i \tag{6.6}$$

where R_i is the average service time for a customer waiting at node i. Note that $L_{\mathsf{QOS}} \in [0, \infty)$, and $L_{\mathsf{QOS}} = 0$ when $R_i = 0,\ \forall i$, i.e. the ideal QOS policy is a solution where the service time at all nodes is zero.

Fuel Consumption

We assume that it is in the interests of the taxi driver to minimize the fuel costs associated with the operation of their vehicle, and that the fuel consumption of a vehicle is reasonably characterized by its total daily mileage. We define the FC metric as

$$L_{\mathsf{FC}} = \sum_i \phi_i (1 - \beta_i) \sum_{j \neq i} p_{i,j}. \tag{6.7}$$

i.e. fuel consumption is minimized by maximizing the amount of time that the redistribution policy dictates the vehicle to remain on standby and wait for a customer at its current location. Note that $L_{\mathsf{FC}} \in [0, 1]$, and $L_{\mathsf{FC}} = 0$ when $S = I_m$, i.e. the ideal FC policy is the solution where all vehicles remain at their current locations and do not redistribute to other nodes.

6.3 Experiments

A simulation framework was implemented in MATLAB. A base cluster network of 27 clusters was created from the Singapore taxi dataset according to the methodology described in Section 5.3. With a discretization of $\tau = 60s$, this yields an extended network of size $m' = 8615$. A one hour epoch was chosen for parameter measurements. The customer arrival rate λ was learned by recording the number of trips made by a

taxi with a customer on board; this was calculated to be approximately 48 arrivals per minute. The customer transition matrix D was learned by recording the distribution of destination nodes of taxis leaving each node with a customer on board. In order to evaluate our proposed policy within the context of the actual taxi system in Singapore, simulations were conduction in comparison with the following two test policies.

Observed Policy

P_{obs} is the actual redistribution policy derived from the Singapore taxi data. This was learned by analyzing the distribution of taxis leaving each node without a customer on board. This is the "ground truth" policy and represents the actual redistribution behavior of an unmanaged taxi fleet. This is the policy that is of most interest to us because it provides an insight into the effectiveness of unmanaged redistribution. Further, the number of taxis n_{obs} that was actually observed to be in operation on Singapore roads was estimated by recording the number of individual taxi IDs that registered journeys within the given epoch.

Arrival Policy

P_{arr} is a "smart" but naive policy that provides a reasonable model for individual taxi driver behavior. The arrival policy is defined by $p_{i,j} = \alpha_j$, i.e. the taxi driver will choose his next location based on the chances that a customer will arrive there.

Simulations were carried out for 8 simulation hours each (at $\tau = 60s$). Each simulation was carried out 5 times for each policy and for each value of n, and the results aggregated. First, n_{min} was determined by means of bootstrap simulations as described in Section 6.1. Then the main test simulations were carried out (105 in total) employing P_{obs}, P_{arr} and P_{HM} policies with n set to increasing multiples of n_{min}. Results were evaluated using the metrics described in Section 6.2.

Policy	n	L_{UP}	L_{QOS}	L_{FC}	% serviced
P_{HM}	n_{min}	1	**0.40**	**0.94**	99.96
P_{obs}		1	0.67	0.99	99.91
P_{arr}		1	0.55	0.99	99.94
P_{HM}	$n_{min} \times 1.2$	1.2	**0.17**	**0.92**	99.99
P_{obs}		1.2	0.40	0.99	99.95
P_{arr}		1.2	0.32	0.98	99.98
P_{HM}	$n_{min} \times 1.4$	1.4	**0.11**	**0.91**	99.99
P_{obs}		1.4	0.26	0.99	99.97
P_{arr}		1.4	0.18	0.99	99.99
P_{HM}	$n_{min} \times 1.6$	1.6	**0.05**	**0.89**	99.99
P_{obs}		1.6	0.19	0.99	99.98
P_{arr}		1.6	0.12	0.98	99.99
P_{HM}	$n_{min} \times 1.8$	1.8	**0.03**	**0.88**	99.99
P_{obs}		1.8	0.15	0.99	99.99
P_{arr}		1.8	0.09	0.98	99.99
P_{HM}	$n_{min} \times 2$	2	**0.03**	**0.87**	99.99
P_{obs}		2	0.12	0.99	99.99
P_{arr}		2	0.07	0.99	99.99
P_{HM}	$n_{min} \times 4$	4	**0.01**	**0.83**	99.99
P_{obs}		4	0.05	0.99	99.99
P_{arr}		4	0.02	0.97	99.99

Table 6.1: Simulation Results. A lower metric indicates better performance. The best policy is highlighted for each n.

6.3.1 Results

Table 6.1 summarizes the simulation results. The optimization algorithm yielded $n_{min} = 1000$ (rounded to the nearest 100 vehicles) for $\lambda = 48$. The number of taxis that was actually observed to be in operation on Singapore roads was recorded from the taxi data as $n_{obs} = 10,088$. This clearly suggests that there are many more taxis in operation in Singapore than strictly necessary to service all customers without a buildup of queues. Since simulations were carried out for fixed n, the UP metric is the same for all policies for a given n. Fig. 6-1a shows a plot of the average percentage of customers serviced for different values of n. Observe that for $n = n_{min}$ the policies marginally satisfy stability constraints, and for $n > n_{min}$ all policies achieve almost

full service.

The rightmost columns of Table 6.1 summarize the results for the QOS and FC metrics. Fig. 6-1b shows the calculated average waiting time of customers for increasing n. We see the HM policy improves substantially over the both test policies for the same number of taxis. Fig. 6-1c shows the average fuel consumption of taxis for increasing n. Again, the HM policy shows an improvement over both test policies.

These results confirm that the unmanaged redistribution behavior is indeed suboptimal. The fact that the HM policy shows a significant improvement for both QOS and FC metrics is intriguing as it indicates that the interests of the taxi driver and the customer may in fact be aligned, and suggests the potential for a natural incentive-based redistribution model for drivers that also improves quality of service.

6.4 Summary

In this chapter we develop a practical deployment policy for a Markov-based model of an urban transportation network. We considered three optimization criteria: urban planning, fuel consumption, and quality of service. We presented a solution that can be computed efficiently. We compared the computed policy to the activity of a fleet of 16,000 taxis in Singapore. Simulation experiments show that there we achieve a significant improvement with respect to all three optimization criteria, and that there is great potential to improve the efficiency of the current deployment strategy.

Chapter 7

Conclusions and Lessons Learned

In this thesis, we presented a complete analysis of the steps involved in developing a deployment algorithm for robotic exploration and patrolling. Specifically, we addressed the motivation behind exploration and patrolling algorithms and their importance within robots in general. We motivated the importance of a clear problem specification and how it can impact our understanding of the environment in an exploration or patrolling scenario. We addressed the problem of characterizing and representing an arbitrary continuous environment, and presented techniques for transforming the problem into a discrete representation. We introduce two novel deployment algorithms: an exploration algorithm for pursuit-evasion and more general policy for patrolling on a graph. Finally, we implemented our patrolling policy to an urban mobility network, and used real historical data to compare it against the actual observed redistribution of taxi drivers in Singapore. We conducted large-scale simulations and presented our results. In this chapter, we summarize the key contributions of each chapter and offer reflections, ideas for future work, and concluding remarks.

In Chapter 3 we considered the problem of obtaining a concise characterization of a physical environment in the context of frontier-based non-recontaminating exploration. We examined the relationship between an environment Q, the sensor radius r, and the number of robots n required for non-recontaminating exploration of Q. Guided by the intuition that corridor width and junctions are important features,

we formalized these notions and presented a general method for computing a configuration space representation of the environment that captures this intuition. We introduced the medial axis as a configuration space and showed that reasoning about points in this configuration space is equivalent to reasoning about robots in physical space. We introduced an exploration model whereby a swarm locus moving along the skeleton allows us to reason about a group of robots moving through physical space. We defined what it means for a frontier to split and for a group of robots to traverse a junction, and derived lower and upper bounds on the number of robots required to traverse a junction.

Reflecting on this part of our work, we express that environment characterization is a very rich and challenging problem. Fundamentally, we want to understand the world. But even within the specific problem domain of deployment algorithms for robotics, the understanding we seek to gain is intrinsically related to the problem that we are trying to solve. Thus it is difficult to develop a general characterization of an environment, just as it is difficult to talk about deployment algorithms in general. In our work, exploration provided us with a model that generalizes well to a lot of deployment scenarios, and yielded results that are general enough to be applied to a wide scope of problem domains.

We would consider the lessons learned from this part of our work to be twofold. First, considering different deployment models can yield different properties of the environment that can turn out to be useful in general. For example, the interpretation the "undirected width" of an environment was motivated by the very specific problem requirements of non-recontaminating exploration, but is indeed a very useful property for environment characterization in general – combined with the medial axis, we have a terse and complete representation of an environment. Second, techniques from seemingly unrelated disciplines may prove useful. For example, the distance transform is primarily used in computer vision, and the medial axis is a theoretical principle from mathematical morphology. When the intuition is already present, and one is looking for methods to capture it, thinking outside the box can be fruitful.

In Chapter 4 we presented a transformation from our continuous configuration space environment representation into a symbolic representation in the discrete domain. We cast the exploration problem as a game, established rules for playing this game, and derived bounds on the number of robots necessary and sufficient to clear the environment. Using the junction lower bound result we derived a lower bound on the total number of pursuers necessary to clear the environment, showing that no fewer than this number can possibly clear the environment regardless of the exploration model or pursuit strategy. Using the junction upper bound we derived an upper bound on the total number of pursuers that will always be sufficient to clear the environment, for any pursuit strategy. Finally, we derive an optimal pursuit strategy and prove that it guarantees we can clear the environment with the minimum number of pursuers for a given exploration model.

There are a number of interesting future lines of research for this work. First, the establishment of tight bounds on the number of robots required to traverse a junction – we suspect that the lower bound given by Result (3.1) in Section 3.2.2 is in-fact necessary and sufficient. A rigorous proof of this fact would have to generalize to accommodate a number of special cases.

Second, the extension of this work to environments with holes would be a significant contribution. The medial axis configuration space was chosen with this in mind, and the model presented in Section 3.2 soundly generalizes the characterization to arbitrary connected environments. A number of issues need to be addressed in transforming this representation into the discrete domain. One would need to consider an undirected graph and junctions would need to be represented accordingly. It is known that computing the number of searchers required to clear a general graph is NP-hard [35], so suitable heuristics or approximations would need to be employed. Alternatively, the graph could be converted into a tree such as in [25],[26]; however the non-isotropic nature of the junction transition function would demand a judicious approach to blocking cycles.

We believe that this work marks an important foundation for the characterization of environments in general. We presented several techniques that can help us

understand the capabilities of a group of robots deployed in an environment with complicated and seemingly arbitrary structural features. We treated a challenging robotic exploration scenario and showed how environment characterization can relate to the number of robots required to guarantee progress and termination. We believe the techniques presented in this work can be applied more generally, and integrated into online exploration algorithms.

In Chapter 5 we extended on the idea of modeling exploration deployment on a graph to the more general case of patrolling. We presented a Markov-based model for an urban transportation network. We showed the steps taken to ensure that the model is realistic while still complying with a Markov framework. We presented a mechanism by which a group of robots can be deployed to patrol the network and showed how such a deployment strategy can be encoded within a Markov model. We presented a scalable optimization framework for computing such a strategy.

In Chapter 6 we proposed a deployment strategy for patrolling the urban mobility network that can be computed efficiently. We considered three evaluation criteria: urban planning, fuel consumption, and quality of service. We conducted large-scale experiments that compared our computed algorithm to the activity of a fleet of 16,000 taxis in Singapore.

We believe that this work marked an important step toward understanding the basic trade-offs between the number of transportation vehicles in the system, vehicle fuel consumption, and the overall quality of service provided by the transportation system. We believe that a Markov model is a useful high-level abstraction of a supply-and-demand chain, offering a simple and robust mechanism for encoding optimization requirements, and a sound platform for more sophisticated solutions. Our practical patrolling policy demonstrates the effectiveness of a Markov model in encoding patrolling strategies that meet challenging optimization criteria. More importantly, we believe that our work shows the effectiveness of patrolling as a general approach to solving real urban mobility problems.

In reflection, a more general problem and deployment model does have some advantages but also some drawbacks. Being able to express an entire deployment policy

in a single Markov chain is an elegant solution to a complicated problem. However, proving the necessary guarantees about this problem was challenging. Ultimately the patrolling policy relied on experimental verification of its effectiveness.

The authors agree with the sentiments expressed in the thorough survey of previous work in [38] that theoretical completeness of proposed solutions is often not essential or even particularly meaningful in the real world. Some of the problems in the application domain of urban mobility are fundamentally intractable. We express that future work should seek to embrace the development of heuristic solutions that may not be optimal, but are effective, robust and efficient.

Bibliography

[1] M. Adler, H. Racke, N. Sivadasan, C. Sohler, and B. Vocking. Randomized pursuit-evasion in graphs. *Combinatorics, Probability and Computing*, 12(03):225–244, 2003.

[2] J. Annas and J. Xiao. Intelligent pursuit & evasion in an unknown environment. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4899–4906. IEEE, 2009.

[3] R. Balakrishna, M. Ben-Akiva, and H.N. Koutsopoulos. Offline calibration of dynamic traffic assignment: simultaneous demand-and-supply estimation. *Transportation Research Record: Journal of the Transportation Research Board*, 2003(-1):50–58, 2007.

[4] G. Berbeglia, J.F. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010.

[5] H. Blum. A transformation for extracting new descriptors of shape. *Models for the Perception of Speech and Visual Form*, 19:362–380, 1967.

[6] J.L. Burgos. Pursuit/evasion behaviors for multi-agent systems. 2010.

[7] H.I. Choi, S.W. Choi, and H.P. Moon. Mathematical theory of medial axis transform. *Pacific Journal of Mathematics*, 181(1):57–88, 1997.

[8] P.E. Danielsson. Euclidean distance mapping. *Computer Graphics and image processing*, 14(3):227–248, 1980.

[9] P. Dimitrov, C. Phillips, and K. Siddiqi. Robust and efficient skeletal graphs. In *cvpr*, page 1417. Published by the IEEE Computer Society, 2000.

[10] J.W. Durham, A. Franchi, and F. Bullo. Distributed pursuit-evasion with limited-visibility sensors via frontier-based exploration. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3562–3568. IEEE, 2010.

[11] T.L. Friesz, D. Bernstein, T.E. Smith, R.L. Tobin, and BW Wie. A variational inequality formulation of the dynamic network user equilibrium problem. *Operations Research*, pages 179–191, 1993.

[12] T.L. Friesz, J. Luque, R.L. Tobin, and B.W. Wie. Dynamic network traffic assignment considered as a continuous time optimal control problem. *Operations Research*, pages 893–901, 1989.

[13] B.P. Gerkey, S. Thrun, and G. Gordon. Visibility-based pursuit-evasion with limited field of view. *The International Journal of Robotics Research*, 25(4):299, 2006.

[14] Peter J. Giblin and Benjamin B. Kimia. Local forms and transitions of the medial axis. In Kaleem Siddiqi and Stephen M. Pizer, editors, *Medial Representations*, volume 37 of *Computational Imaging and Vision*, pages 37–68. Springer Netherlands, 2008.

[15] L. Guibas, J.C. Latombe, S. Lavalle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. *Algorithms and Data Structures*, 1272:17–30, 1997.

[16] L.J. Guibas, C. Holleman, and L.E. Kavraki. A probabilistic roadmap planner for flexible objects with a workspace medial-axis-based sampling approach. In *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 1, pages 254–259. IEEE, 1999.

[17] L.J. Guibas, J.C. Latombe, S.M. LaValle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry and Applications*, 9(4/5):471, 1999.

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

[19] J.P. Hespanha, H.J. Kim, and S. Sastry. Multiple-agent probabilistic pursuit-evasion games. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 3, pages 2432–2437. IEEE, 1999.

[20] C. Holleman and L.E. Kavraki. A framework for using the workspace medial axis in prm planners. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1408–1413. IEEE, 2000.

[21] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias. Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, 28(2):201, 2009.

[22] Texas Transportation Institute. Urban mobility report. Technical report, 2011.

[23] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *Robotics, IEEE Transactions on*, 21(5):875–884, 2005.

[24] V. Isler, D. Sun, and S. Sastry. Roadmap based pursuit-evasion and collision avoidance. In *Proc. Robotics, Systems, & Science*, 2005.

[25] A. Kolling and S. Carpin. The graph-clear problem: definition, theoretical properties and its connections to multirobot aided surveillance. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1003–1008. IEEE, 2007.

[26] A. Kolling and S. Carpin. Multi-robot surveillance: an improved algorithm for the graph-clear problem. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2360–2365. IEEE, 2008.

[27] A. Kolling and S. Carpin. Pursuit-evasion on trees by robot teams. *Robotics, IEEE Transactions on*, 26(1):32–47, 2010.

[28] A. Kuijper. Deriving the medial axis with geometrical arguments for planar shapes. *Pattern Recognition Letters*, 28(15):2011–2018, 2007.

[29] S.M. LaValle, D. Lin, L.J. Guibas, J.C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 1, pages 737–742. IEEE, 1997.

[30] S. Lim, H. Balakrishnan, D. Gifford, S. Madden, and D. Rus. Stochastic motion planning and applications to traffic. *The International Journal of Robotics Research*, 30(6):699–712, 2011.

[31] D.K. Merchant and G.L. Nemhauser. Optimality conditions for a dynamic traffic assignment model. *Transportation Science*, 12(3):200–207, 1978.

[32] W.J. Mitchell, C.E. Borroni-Bird, and L.D. Burns. Reinventing the automobile. *Personal Urban Mobility for the 21st Century. Cambridge/l\/IA*, 2010.

[33] R. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235–239, 1983.

[34] S.N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008.

[35] T. Parsons. Pursuit-evasion in a graph. *Theory and applications of graphs*, 642:426–441, 1978.

[36] M. Pavone, S.L. Smith, E. Frazzoli, and D. Rus. Load balancing for mobility-on-demand systems. *Robotics: Science and Systems, Los Angeles, CA*, 2011.

[37] M. Pavone, S.L. Smith, E. Frazzoli, and D. Rus. Robotic load balancing for mobility-on-demand systems. 2011.

[38] S. Peeta and A.K. Ziliaskopoulos. Foundations of dynamic traffic assignment: The past, the present and the future. *Networks and Spatial Economics*, 1(3):233–265, 2001.

[39] S. Rodriguez, J. Denny, T. Zourntos, and N. Amato. Toward simulating realistic pursuit-evasion using a roadmap-based approach. *Motion in Games*, 6459:82–93, 2010.

[40] T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM (JACM)*, 49(2):236–259, 2002.

[41] S. Sachs, S.M. LaValle, and S. Rajko. Visibility-based pursuit-evasion in an unknown planar environment. *The International Journal of Robotics Research*, 23(1):3, 2004.

[42] J. Serra. *Image analysis and mathematical morphology*. London.: Academic Press.[Review by Fensen, EB in: J. Microsc. 131 (1983) 258.] Technique Staining Microscopy, Review article General article, Mathematics, Cell size (PMBD, 185707888), 1982.

[43] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on computing*, 21:863, 1992.

[44] M. Volkov, J. Aslam, and D. Rus. Markov-based redistribution policy model for future urban mobility networks. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 1906–1911. IEEE, 2012.

[45] M. Volkov, A. Cornejo, N. Lynch, and D. Rus. Environment characterization for non-recontaminating frontier-based robotic exploration. *Agents in Principle, Agents in Practice*, pages 19–35, 2011.

[46] J.G. Wardrop. Some theoretical aspects of road traffic research. In *Inst Civil Engineers Proc London/UK/*, number 0, 1900.

[47] S.A. Wilmarth, N.M. Amato, and P.F. Stiller. Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1024–1031. IEEE, 1999.

[48] M. Yamashita, H. Umemoto, I. Suzuki, and T. Kameda. Searching for mobile intruders in a polygonal region by a group of mobile searchers. *Algorithmica*, 31(2):208–236, 2001.

[49] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151. IEEE, 1997.

[50] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53. ACM, 1998.

[51] A.K. Ziliaskopoulos. A linear programming model for the single destination system optimum dynamic traffic assignment problem. *Transportation science*, 34(1):37–49, 2000.