Robot Patrolling for Stochastic and Adversarial Events

by

Ahmad Bilal Asghar

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2015

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In this thesis, we present and analyze two robot patrolling problems. The first problem discusses stochastic patrolling strategies in adversarial environments where intruders use the information about a patrolling path to increase chances of successful attacks on the environment. We use Markov chains to design the random patrolling paths on graphs. We present four different intruder models, each of which use the information about patrolling paths in a different manner. We characterize the expected rewards for those intruder models as a function of the Markov chain that is being used for patrolling. We show that minimizing the reward functions is a non convex constrained optimization problem in general. We then discuss the application of different numerical optimization methods to minimize the expected reward for any given type of intruder and propose a pattern search algorithm to determine a locally optimal patrolling strategy. We also show that for a certain type of intruder, a deterministic patrolling policy given by the orienteering tour of the graph is the optimal patrolling strategy.

The second problem that we define and analyze is the Event Detection and Confirmation Problem in which the events arrive randomly on the vertices of a graph and stay active for a random amount of time. The events that stay longer than a certain amount of time are defined to be true events. The monitoring robot can traverse the graph to detect newly arrived events and can revisit these events in order to classify them as true events. The goal is to maximize the number of true events that are correctly classified by the robot. We show that the off-line version of the problem is NP-hard. We then consider a simple patrolling policy based on the TSP tour of the graph and characterize the probability of correctly classifying a true event. We investigate the problem when multiple robots follow the same path, and show that the optimal spacing between the robots in that case can be non uniform.

Acknowledgements

I would like to thank my advisor Professor Stephen Smith for his constant guidance and support throughout the program. I would like to express my sincere gratitude for his patience, motivation and highly informative discussions during our meetings.

I would also like to thank the readers of this thesis — Professor Andrew Heunis and Professor Steven Waslander — for their valuable time.

I should also thank Frank Imeson and Armin Sadeghi for their help and insightful discussions.

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

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

We discuss two robot patrolling problems in this thesis. The problem of finding a path in an environment to optimize some objective is a wide area of research ranging from Traveling Salesman Problem where the objective function is to minimize the length of the path to persistent monitoring of stochastic events where the target is to minimize the delay between observations of events. We look at two monitoring problems where the events of interest are stochastic and/or intelligent.

1.1 Patrolling in Adversarial Settings

Consider a marketplace where a security agent is patrolling at night to avoid thefts. Each of the shops has a certain value, e.g. a jewelery shop has more incentive for the thief than a grocery shop. If the patrolling path of the agent is predictable, a potential thief can study the path, and time the theft to avoid being detected. As a simple example, assume that the patrolling agent takes 45 minutes to complete a round of the market and hence visits each shop every 45 minutes. If the thief requires half an hour to steal, he can commit the crime without being detected. Moreover, if the patrolling agent visits the grocery shop as often as the jewelery shop, the thief would make an attempt at the more valuable item. So, in such a case, randomizing the patrolling path can increase the surveillance quality of the area. Another relevant example is in border patrol [3, 1] where an intruder can time its crossing of the border in order to infiltrate unnoticed.

We model the environment to be monitored as a graph. The robot visits the regions of the environment(vertices of the graph) on its surveillance path and detects any malicious

activities. Some regions of the environment might be more important or more prone to attack than others. Meanwhile, an intruder is trying to launch attacks on the region with a specified length of attack. If the intruder is intelligent enough, it can observe the patrolling path and use the information to increase its chances of successfully attacking the environment. We study the effects of randomizing the patrolling path in the presence of such intruders and attempt to design the paths with a better surveillance quality for the environment.

The robotic surveillance problem is very well studied and there is a substantial amount of work related to design of patrolling policies for surveillance of environments. For deterministic paths, there is a breadth of work considering both single and multiple robots [39, 34, 42, 16]. In [5] the authors discuss deterministic monitoring strategies to minimize the weighted latency of the path(i.e. the maximum time between visits to a vertex weighted by the importance of that vertex). The path visits the vertices with more weight more often. Our work looks at a similar problem, but considers intelligent intruders and thus we look at non deterministic paths.

Srivastava *et al.* [40] advocate random patrolling paths by showing that it is hard to find deterministic strategies that satisfy the surveillance criterion of visiting vertices proportional to their importance. In [24] the authors consider stochastic surveillance paths so that the intruders cannot easily exploit the predictability of a deterministic path. We extend this notion and look at ways in which an intelligent intruder can learn a randomized path, and utilize this information to launch attacks.

In [3] the authors look at intelligent intruders in a perimeter patrolling problem. Our problem is different in the sense that we deal with environments represented as graphs, for which border patrol is a special case. Burda *et al.* [14] find maximum entropy random walks such that all the paths of length t with the same endpoints have equal probability of being traversed.

Markov chains are often used to model random paths on a graph [40, 24, 2]. For example, in [2] the authors represent the environment as a graph and design Markov chains that generate randomized paths with desired behaviors. They use semidefinite programming to minimize the mean first passage time from one state of the Markov chain to any other state, which is constant for a given chain and is called Kemeny constant [26]. The problem can be formulated as a semidefinite program for reversible Markov chains only. They also use the Markov chains on weighted graphs, where the probability of traversing an edge is independent of the weight of that edge in the graph. This formulation is more useful in the case of surveillance since the environment is usually represented as a weighted graph and the Markov chain is then defined on that graph. This *doubly weighted graph*

formulation is not studied much in the literature to the best of our knowledge. We study the problem for both the weighted and unweighted graphs.

Boyd *et al.* [12] design a Markov chain with fastest mixing time which approaches the steady state distribution as quickly as possible. They also formulate the problem as a semidefinite problem. In [4] the authors use a finite set of Markov chains to formulate the problem as a *Bayesian Stackelberg Game* where the patroller first picks a mixed optimal strategy and the adversary then picks the region to attack to maximize its payoff.

In [37] the authors empirically show that randomizing the paths can decrease the probability of successful intrusions.They use different ways to randomize the path instead of using Markov chains, like randomly visiting some vertices within a traveling salesman problem(TSP) tour or randomly interchanging the order of vertices in each cycle of the TSP tour. They consider three different types of intruders based on the attacking strategy. We also present different attack models for the intruders and analyze their performance. The approach of presenting attack intruder models for attacks on the environment is similar to that used in the computer security literature (for example, [21] on encryption or [9] on wireless sensor networks).

1.2 Robot Monitoring for Detection and Confirmation of Stochastic Events

Consider the following motivating example. An autonomous robotic vehicle traverses a parking lot, issuing tickets to vehicles that have overstayed the allowed amount of parking time T. The robot goes from spot to spot, recording the time and license plate numbers of the parked vehicles. If a vehicle has been present for more than T time units after it was first spotted by the robot, then it gets a ticket. However, there is a possibility that some vehicles overstay their allowed parking time but leave before the robot has a chance to ticket them. Our goal is to define a monitoring policy for the robot which minimizes the number of un-ticketed, overstaying vehicles.

Formally, we define the *event detection and confirmation problem* as follows. We are given a weighted graph. Events arrive randomly at the vertices of the graph, and a robot or group of robots can patrol the graph by traversing its edges. Once an event arrives at a particular vertex it remains active for a randomly distributed amount of time. If the event remains active for more than a given threshold time $T > 0$, then we say it is a *true event*, otherwise it is a *false event*. For a robot to verify that an event is true, it must first detect the event by visiting the vertex, and then must revisit the vertex at least T time

units later to confirm the event. Thus, the goal for the robots is to maximize the expected number of true events that are successfully confirmed. This is a classification problem in which false positives are not permitted: Each event is initially classified as false, and it can be classified as true only if it is confirmed.

Several closely related problems to the proposed event detection and confirmation problem have been studied in the literature. In the patrolling problems discussed in [16, 3, 8], the goal is to monitor an environment or boundary using one or more robots/sensors. The performance criteria is to minimize the maximum time between visits to any region in the environment. In [16], the problem is considered for multiple robots, and it is shown that good patrolling performance can be achieved by computing a single traveling salesman tour (TSP) [28], and then equally distributing the robots along this tour. We show that for the event detection and confirmation problem, equally distributing two robots along a tour might not be the best choice.

In [32], the authors look at cooperative patrolling problems and give approximation algorithms for certain classes of discrete environments. In [5] the patrolling is extended to environments in which each region has a different importance level, and the goal is to minimize the time between visits to a region, weighted by that regions importance. The authors give approximation algorithms to the hard problem. The work in this thesis can be thought of as a natural extension of patrolling in which an action must be taken if an event is detected during the patrol (that action being confirmation).

TSP with time windows [38, 29] is also a related problem where the input is a graph along with a time window assigned to each vertex. The goal is to find the shortest tour that visits each vertex exactly once, and within its time window. We show that the event confirmation aspect of our problem is closely related to TSP with time windows, since each event must be confirmed at least T time units after its detection, but before the event expires.

Another closely related problem is the pickup and delivery problem [35], where one seeks to pick up a set of customers at their origin locations and drop them off at their desired destination locations, all within their specified time windows. Our problem can be thought of as a variation in which the pickup time (i.e., the event arrival time) is unknown to the robot, the pickup and destination locations coincide, and the dropoff time window depends on the time that the pickup occurred (i.e., the event was detected).

The stochastic aspect of the problem bears a close resemblance to dynamic vehicle routing (DVR) [13], where spatially distributed customers arrive stochastically over time, and the goal is to minimize the expected time between a customers arrival and the time it is visited by a vehicle. The most closely related work in this area is [33], in which the

customers are impatient and hence exit the system if they are not visited within a time window. However, DVR differs from the proposed work in three regards: i) the environment is a Euclidean space rather than a graph, ii) the customer is known to the vehicles upon arrival, and iii) a second confirmation visit is not required.

The event detection and confirmation problem is also related to [42] where a patroller is visiting the vertices of a graph to detect stochastic events. The arrival rate of the events at each station is different and known to the patroller. The authors discuss two objective functions. First is to maximize the minimum expected number of observed events at a location and the second is to minimize the maximum delay between consecutive observations at a location. They propose an optimal policy which optimizes both of these objective functions for a given cyclic tour of the graph. Our approach is similar to theirs in the respect that we also analyze our problem based on a periodic tour of the graph.

1.3 Contributions

We analyze two patrolling problems in this thesis. For the patrolling in adversarial settings problem, we formally define the intruder attack models for four different types of intruders. Each of the intruder models require different capabilities and knowledge from the intruder. We characterize the intelligent intruders that can learn the patrolling path and use this information to launch attacks on the environment. The expected reward for each of these intruders is derived given a stochastic patrolling policy. The expected reward for a given intruder serves as the objective function to find a patrolling policy in the presence of that intruder. We then show that for an intelligent intruder that has to attack pre-assigned locations of the environment, a deterministic patrolling policy minimizes the intruder's expected reward. Finding this deterministic patrolling path is equivalent to solving the ORIENTEERING PROBLEM which is a hard problem. The expected reward of the intruder is a non convex and non smooth function of the patrolling policy, and hence we discuss some of the numerical optimization techniques from the existing literature that can be applied to our problem. We show that the gradient projection method can be applied to find a local minimum of the of the objective function, but it is very inefficient. We propose the pattern search method to find the local minimum of the expected reward and present the experimental results.

For the event detection and confirmation problem, we characterize the complexity of the problem by relating the off-line version of the problem to the TSP with time windows. We then propose a simple periodic visit strategy based on the TSP and evaluate the probability of confirming a true event for that policy. We show that for more than one robot, it is not

always optimal to equally space the robots if they are on the same path, unlike traditional patrolling [16],

1.4 Organization

The organization of this thesis is as follows. The mathematical preliminaries and background are given in chapter 2. We also introduce some notation in this chapter that we will be using throughout the thesis. Chapter 3 discusses patrolling in adversarial settings in detail. The event detection and confirmation problem is presented and studied in chapter 4. Finally, the conclusions and future work directions are given in chapter 5.

Chapter 2

Preliminaries

In this chapter we provide some mathematical background and notation for the thesis. We present some concepts related to probability and random processes in Section 2.1. In Section 2.2 we present Markov chains. Some basic definition from graph theory are given in Section 2.3. The definitions related to probability and random processes are from [25]. The background on Markov chains is from [26, 25] and the definitions for graph theory are from [11].

2.1 Probability and Random Processes

We cannot predict the outcome of a random experiment with certainty before the completion of the experiment. Probability is a way to model the uncertainty in the outcomes of random experiments. We present some basic definitions from probability theory and then mention some results which will be useful in this thesis.

Definition 2.1.1 (Sample Space)**.** *The set of all possible outcomes of an experiment is called the sample space and is denoted by* Ω

A collection of outcomes of an experiment is called an *event*. The empty set ϕ is called the impossible event and the set Ω is called a certain event since the experiment's outcome will certainly be an element of Ω by the definition of sample space.

Definition 2.1.2 (σ-field)**.** *A collection* $\mathcal{F}$ *of subsets of* Ω *is called a* σ*-field if it satisfies the following conditions.*

1. $\varnothing \in \mathcal{F}$

2. *If* $A_1, A_2, \ldots \in \mathcal{F}$, *then* $\bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$

3. *If* $A \in \mathcal{F}$, *then* $A^c \in \mathcal{F}$

where $A^c = \Omega \backslash A$ *is the complement of set* A.

It follows from the definition that the smallest σ-field is always $\{\phi, \Omega\}$ and the power set of Ω written as 2^{Ω} and containing all the subsets of Ω is also a σ-field.

Definition 2.1.3 (Probability Measure)**.** *A function* $\mathbb{P} : \mathcal{F} \to [0, 1]$ *satisfying*

1. $\mathbb{P}(\phi) = 0$.

2. $\mathbb{P}(\Omega) = 1$.

3. *If* $A_1, A_2, \ldots$ *are disjoint members of* $\mathcal{F}$, *then*

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mathbb{P}(A_i)$$

is called a probability measure.

Definition 2.1.4 (Probability space)**.** *The triple* $(\Omega, \mathcal{F}, \mathbb{P})$ *is called a probability space.*

Definition 2.1.5 (Conditional Probability)**.** *Given two events* A *and* B *such that* $\mathbb{P}(B) > 0$, *the conditional probability of* A *occurring given that* B *occurs is*

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

Definition 2.1.6 (Independence)**.** *Events* A *and* B *are called independent if*

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$$

2.1.1 Random Variables

Random variables can be thought of as consequences related to the outcomes of the experiments.

Definition 2.1.7 (Random Variable)**.** *A random variable is a function $X : \Omega \to \mathbb{R}$ such that $\{\omega \in \Omega | X(\omega) \leq x\} \in \mathcal{F}$ for each $x \in \mathbb{R}$.*

Definition 2.1.8 (Distribution Function)**.** *The distribution function of a random variable X is a function $F : \mathbb{R} \to [0, 1]$ given by*

$$F(x) = \mathbb{P}[X \leq x]$$

Definition 2.1.9 (Discrete Random Variable)**.** *The random variable X is called discrete if takes values in some countable set. It has a* ***probability mass function(PMF)*** *$f : \mathbb{R} \to [0, 1]$ given by*

$$f(x) = \mathbb{P}[X = x]$$

Definition 2.1.10 (Continuous Random Variable)**.** *The random variable X is called continuous if its distribution function $F(x)$ can be written as*

$$F(x) = \int_{-\infty}^{x} f(u)du$$

for some integrable function $f : \mathbb{R} \to [0, \infty]$ called the ***probability density function(PDF)*** *of X.*

Definition 2.1.11 (Expectation of Discrete Random Variable)**.** *The expected value or expectation of a discrete random variable X with PMF $f(x)$ is*

$$\mathbb{E}(X) = \sum_{x} xf(x)$$

Definition 2.1.12 (Expectation of Continuous Random Variable)**.** *The expected value or expectation of a continuous random variable X with PDF $f(x)$ is*

$$\mathbb{E}(X) = \int_{-\infty}^{\infty} xf(x)dx$$

For a function $g : \mathbb{R} \to \mathbb{R}$, the expectation of the function $g(X)$ of random variable X is given by $\mathbb{E}(g(X)) = \int_{-\infty}^{\infty} g(x)f(x)dx$.

Expectation is a linear operator. So

$$\mathbb{E}(aX + bY) = a\mathbb{E}(X) + b\mathbb{E}(Y), \quad a, b \in \mathbb{R}$$

Definition 2.1.13 (Variance)**.** *The variance of a random variable X is*

$$\begin{aligned}\texttt{Var}(X) =& \mathbb{E}([X - \mathbb{E}(X)]^2) \\ =& \mathbb{E}(X^2) - (\mathbb{E}(X))^2\end{aligned}$$

2.1.2 Some Distribution Functions

Now we describe some useful distribution functions that we will be using in the thesis.

Exponential Distribution

A positive random variable X is said to be exponentially distributed with parameter $\lambda > 0$ if

$$\mathbb{P}[X \leq x] = F(x) = 1 - e^{-\lambda x}$$

The probability density function of the random variable is given by

$$f(x) = \lambda e^{-\lambda x}$$

The parameter λ is usually called the rate parameter. This distribution is commonly used to model the time between unpredictable events. The mean of an exponentially distributed variable with parameter λ is $1/\lambda$.

An exponentially distributed variable obeys the memoryless property which is defined below.

Definition 2.1.14. *The probability distribution of a continuous random variable X is memoryless if*

$$\mathbb{P}[X > s + t | X > s] = \mathbb{P}[X > t], \quad \forall s, t > 0$$

Multinomial Distribution

Let $X \in \{1, 2, \ldots, K\}$ be a discrete random variable, and $\mathbb{P}[X = j] = \theta_j$, then

$$\mathbb{P}[X] = \prod_{j=1}^{K} \theta_j^{(X=j)}$$

is a multinomial distribution where $(X = j)$ is a truth statement that evaluates to 1 or 0. In Chapter 5, we represent each row of the transition matrix of Markov chain as a multinomial distribution. Then, learning of Markov chains is equivalent to estimating the parameters of the multinomial distributions. In that case the parameters θ of the multinomial distributions are not fixed and we can write

$$\mathbb{P}\left[X|\theta\right] = \prod_{j=1}^{K} \theta_j^{(X=j)}$$

Given n independent trials, let X_i be the discrete random variable representing the number of times the outcome of the trial is $i \in \{1, 2, \ldots, K\}$. Then the probability mass function of multinomial distribution is also defined as

$$f(x_1, \ldots, x_K) = \begin{cases} \frac{n!}{x_1! \ldots x_K!} \theta_1^{x_1} \ldots \theta_K^{x_K}, & \text{when } \sum_{i=1}^{K} x_i = n \\ 0, & \text{otherwise.} \end{cases}$$

Dirichlet Distribution

Dirichlet distribution is a continuous multivariate distribution. The Dirichlet distribution with parameters $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_K\}$ has the probability density function

$$f(x_1, \ldots, x_K) = \frac{1}{\beta(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}$$

such that

$$\begin{aligned} x_1, x_2, \ldots, x_{K-1} &> 0 \\ x_1 + x_2 + \ldots + x_{K-1} &< 1 \\ x_K &= 1 - x_1 - \ldots - x_{K-1} \end{aligned}$$

The function $\beta(\alpha)$ is the normalizing constant.

The mean of the random variable X_i from the Dirichlet distribution is given by

$$\mathbb{E}\left[X_i\right] = \frac{\alpha_i}{\alpha_o} \tag{2.1}$$

and the variance in X_i is given by

$$\texttt{Var}X_i = \frac{\alpha_i(\alpha_o - \alpha_i)}{\alpha_o^2(\alpha_o + 1)} \tag{2.2}$$

where $\alpha_o = \sum_i \alpha_i$.

2.1.3 Poisson Process

A family $\{X_t : t \in T\}$of random variables is called a random process. If $T = \{1, 2, , \ldots\}$, it is discrete random process whereas if $T = \mathbb{R}$ or $T = \mathbb{R}_+$, it is a continuous random process.

Poisson process is a continuous time random process that counts the number of events $N(t)$ that have arrived in the time interval $[0, t]$ with $N(0) = 0$. So, the process can be written as $\{N(t) : t \geq 0\}$.

A Poisson process with intensity λ takes values $\{0, 1, \ldots\}$ such that

1. $N(0) = 0$
2. If $s < t$, then $N(s) \leq N(t)$.
3. $$\mathbb{P}\left[N(t+h) = n + m | N(t) = n\right] = \begin{cases} \lambda h + o(h) & if m = 1 \\ o(h) & if m > 1 \\ 1 - \lambda h + o(h) & if m = 0 \end{cases} \tag{2.3}$$
4. If $s < t$, then number of events arriving in $(s, t]$ is independent of the number of events in $[0, s]$.

The distribution of Poisson process is given by

$$\mathbb{P}\left[N(t) = j\right] = \frac{(\lambda t)^j}{j!} e^{-\lambda t}$$

Let $T_o = 0$ and T_i be the time of arrival of the i^{th} event. Then the random variable $X_n = T_n - T_{n-1}$ is called the n^{th} inter arrival time. The inter arrival times $X_1, X_2, \ldots$ of a Poisson process are independent and exponentially distributed with parameter λ.

The expected number of arrivals during an interval of length t for a Poisson process is

$$\mathbb{E}(N(t)) = \lambda t$$

The following result about Poisson processes will be useful in our analysis.

Lemma 2.1.15 (Poisson Arrival Time Distribution, [36]). *Given that k events arrived in the time interval $(a, b]$, the times $t_1, t_2, \ldots, t_k$ of these arrivals, considered as unordered random variables, are independent and uniformly distributed on $(a, b]$.*

A consequence of this result is that if we know that an event arrived in an interval of time, then its arrival time is uniformly distributed over that time interval.

Now we move on to a discrete random process called Markov chain which is helpful in modeling random paths on graphs.

2.2 Markov Chains

We will be using Markov chains to model random surveillance paths. This section gives the necessary background on Markov chains.

A Markov chain is a discrete stochastic process $X = \{X_o, X_1, \ldots\}$ where X_i takes values in a countable set S called the state space.

Definition 2.2.1 (Markov Chain). *A Markov chain is random process X that satisfies the Markov condition*

$$\mathbb{P}\left[X_n = s | X_o = x_o, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}\right] = \mathbb{P}\left[X_n = s | X_{n-1} = x_{n-1}\right]$$

for all $n \geq 1$ and for $s, x_1, x_2, \ldots, x_{n-1} \in S$

Definition 2.2.2 (Homogeneous Markov Chain). *A Markov chain is called homogeneous if the transition probability from one state to the other is independent of time, i.e.*

$$\mathbb{P}\left[X_{n+1} = j | X_n = i\right] = \mathbb{P}\left[X_1 = j | X_o = i\right]$$

for all $n \geq 1$ and for all $i, j \in S$.

Definition 2.2.3 (Transition Matrix). *The transition matrix of a homogeneous Markov chain P is the $|S| \times |S|$ matrix of the transition probabilities. So,*

$$p_{ij} = \mathbb{P}\left[X_{n+1} = j | X_n = i\right]$$

We will only be considering homogeneous Markov chains and we will use notation P generally for the transition matrix.

The transition matrix is a row stochastic matrix, i.e.

- $p_{ij} \geq 0$, for all i, j.
- $\sum_j p_{ij} = 1$ for all i.

Definition 2.2.4 (Recurrent State). *A state $i \in S$ is called recurrent if the probability of eventually visiting state i, starting from state i is 1. If this probability is less than 1, then the state is called transient.*

In the following definition, the notation $p_{ij}^{(n)}$ will represent the element in the i^{th} row and j^{th} column of the matrix P^n.

Definition 2.2.5 (Communication). *We say that state i communicates with state j, if $p_{ij}^{(m)} > 0$ for some $m > 0$. The states i and j intercommunicate with each other if i communicates with j and j communicates with i.*

Definition 2.2.6 (Communicating Class). *A set of states C is called a communicating class if states i and j intercommunicate with each other for $i, j \in C$ and no state in C communicates with any state not in C.*

Notice that if a Markov chain has more than one communicating class, we can decompose the chain into two independent Markov chains. Hence, we will not be considering Markov chains with more than one communicating class in this thesis. However, a Markov chain with one communicating class can have a set of transient states as well. Once, the chain leaves the set of transient states and moves into the communicating class, it will not return to those states.

Definition 2.2.7 (Irreducible Chain). *A Markov chain is called irreducible if states i and j intercommunicate with each other, for all $i, j \in S$.*

Definition 2.2.8 (Regular Markov Chain). *A Markov chain is called regular if all sufficiently higher powers of the transition matrix have only positive elements.*

We will be dealing with the Markov chains with one communicating class in this thesis. The regular Markov chains are a subset of Markov chains with one communicating class. The work done in [2] relating to minimizing the Kemeny constant of a Markov chain considers regular Markov chains only. We will refer to their work for comparison purposes. It should be noted that the method we propose deals with Markov chains that have one communicating class and are not necessarily regular.

Definition 2.2.9 (Stationary Distribution). *The vector $\pi \in \mathbb{R}^{1\times|S|}$ is called the stationary distribution of Markov chain if*

1. $\pi_j \geq 0$, *for all* j.

2. $\sum_j \pi_j = 1$.

3. $\pi P = \pi$.

Definition 2.2.10 (First Passage Time, [2, 26])**.** *Let T_j denote the number of transitions after which the chain reaches state j for the first time, i.e. $T_j = \min\{k \geq 1 : X_k = j\}$. Then the conditional mean of this time $m_{ij} = \mathbb{E}[T_j | X_0 = i]$ is called the first passage time from state i to state j.*

The mean first passage time can be written as

$$m_{ij} = p_{ij} + \sum_{k \neq j} p_{ik}(m_{ik} + 1)$$

Theorem 2.2.11 (Kemeny Constant, [26])**.** *The quantity $\sum_j m_{ij}\pi_j$ is independent of the starting state i for a regular Markov chain.*

Definition 2.2.12. *The quantity $\sum_j m_{ij}\pi_j$ represents the mean of the first passage time from state i to any other state and is called the Kemeny constant of the Markov chain.*

For the Markov chains, where all the transitions do not take equal amount of time, the above defined first passage times will refer to the number of transitions instead of the time taken in going from one state to the other. Let w_{ij} be the time spent in the transition from state i to state j. Then the first passage time n_{ij} will be given by [2]

$$n_{ij} = p_{ij}(w_{ij}) + \sum_{k \neq j} p_{ik}(n_{kj} + w_{kj}).$$

This concept of first passage times for weighted graphs is used in [2] to find Markov chains with minimal first passage times.

The next section gives some basic definitions from graph theory that we will be using in the thesis.

2.3 Graphs

A graph G is defined as a pair $G = (V, E)$, where V represents the set of vertices or nodes of the graph and E represents the set of edges connecting the vertices of the graph. The vertices $u, v \in V$ are connected if $\{u, v\} \in E$

Definition 2.3.1 (Weighted Graph)**.** *A graph is weighted if there is a function $w : E \to \mathbb{R}$ assigning lengths to edges of the graph.*

We write a weighted graph as a triple $G = (V, E, w)$. In this thesis we may also treat the weights on the edges as a matrix $W = [w_{ij}]$ where the element w_{ij} gives the length on the edge $\{i, j\}$. Moreover, we will only deal with graphs that have non negative edge weights.

Definition 2.3.2 (Directed Graph)**.** *A graph is called a directed graph if edges are directed from one vertex to the other. In this case, each edge is an ordered pair of the form (u, v) which means that the edge points from u to v. So, $E \subseteq V \times V$.*

In this thesis, we only consider undirected graphs.

Definition 2.3.3 (Subgraph)**.** *A subgraph $H = (V_H, E_H)$ of G is a graph with $V_H \subseteq V$ and $E_H \subseteq E$.*

Definition 2.3.4 (Path)**.** *A path in a graph $G = (V, E)$ is a subgraph*

$$P = (\{v_1, v_2, \ldots, v_{k+1}\}, \{e_1, e_2, \ldots, e_k\})$$

such that $v_i \neq v_j, \forall i \neq j$ and $e_i = \{v_i, v_{i+1}\}$

Definition 2.3.5 (Walk)**.** *A walk is a sequence $v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1}$ such that $e_i = \{v_i, v_{i+1}\}$ and $e_i \neq e_j$, for all $i \neq j$.*

Definition 2.3.6 (Edge Progression)**.** *An edge progression is a sequence $v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1}$ such that $e_i = \{v_i, v_{i+1}\}$.*

In this thesis we often refer to the path as just a sequence of vertices or an edge progression in the graph. In this context a *random path* on a graph is just a random sequence of vertices of the graph.

The following section discusses a combinatorial problem related to graph theory that we will be referring to in the thesis.

2.3.1 Orienteering Problem

The ORIENTEERING PROBLEM [23] is a variant of the TRAVELING SALESMAN PROBLEM. The input to the problem is a weighted directed or undirected graph $G = (V, E, w)$, two vertices $s, t \in V$, and a time limit T_{max}. Each vertex $i \in V$ has a score ψ_i associated with it. The problem is to find an $s - t$ walk of length not exceeding T_{max} which maximizes the total score obtained by visiting the vertices. If a vertex is visited more than once in the walk, its score is counted only once.

A straightforward reduction from TSP renders the orienteering problem NP-hard. Chekuri *et al.* [15] give a $(2 + \epsilon)$ approximation algorithm for the problem. The existing exact and approximation algorithms are discussed and compared in [41].

We will refer to a slight variation of the ORIENTEERING PROBLEM where instead of finding an $s - t$ walk, the objective is to find any closed path(edge progression) in the graph of length at most T_{max} maximizing the collected score. So, the input to the problem does not contain the vertices s and t.

Chapter 3

Patrolling in Adversarial Settings

In this chapter we discuss the patrolling problem in adversarial settings where the intruders may be intelligent enough to increase the chances of undetected attacks on the environments. We define the problem in Section 3.1. Then we present the intruder models for different types of intruders and analyze their performance in Section 3.2. In Section 3.3 we characterize the objective function and advocate deterministic patrolling paths for a certain type of intruder. We present some numerical methods to optimize the objective function in Section 3.4. We give experimental results in Section 3.5.

3.1 Problem Statement

Consider a surveillance robot monitoring an environment to detect possible attacks on the areas of the environment by an intruder. We represent the environment by an undirected graph $G = (V, E)$ where the set of vertices $V = \{1, 2, ..., n\}$ represents the areas to be monitored and the edges represent the links along which robot can travel. Each vertex i of the graph has an importance associated with it given by ϕ_i. This importance represents the value of that vertex or equivalently intruder's reward for launching a successful attack on vertex i. The length of attack is defined as l and an attack is called *successful* if the surveillance robot does not visit the attacked vertex for the duration of the attack. An intelligent intruder would try to learn the surveillance path and use the information to predict the robot's location in the future. For example if the surveillance path is deterministic and the time taken between consecutive visits to some vertex is greater than the length of attack, then the intruder can time the attack to make sure that it is successful. In such a case, having a random patrolling path may decrease the chances of successful attacks.

We use first order Markov chains to define random patrolling paths on the graphs. The transition matrix of the Markov chain $P = [p_{ij}]$ governs the random path of the robot. The probability of going from vertex i to vertex j is independent of the history of the path and is given by p_{ij}. Note that higher order Markov chains can be employed to use the history of the path to decide the next move but we only consider first order chains in this work. We call the transition matrix P the *patrolling policy* since it completely characterizes the patrolling path to be followed by the robot.

The probability of successful attack on a vertex depends on the attack model being followed by the intruder. The attack model defines the procedure by which the intruder decides the time and vertex of attack. The model may or may not use the information the intruder has gathered about the robot's patrolling path. Our goal is to find a patrolling policy given an attack model that minimizes the expected reward of the intruder.

> **Problem Statement:** Given a graph $G = (V, E)$ and the intruder's attack model, determine a patrolling policy P that minimizes the expected reward of the intruder.

We will formally define some of the attack models in the next section and find expressions for the probability of successful attacks on the vertices of the graph.

3.2 Intruder Attack Models

This section discusses some of the possible policies that an intruder can follow to launch attacks on the environment. We will assume that there is only one attack on the environment at a time. The intruder model will govern the vertex at which the attack is to be launched and the time of launching the attack. To be precise, an intruder model will define the following:

- Vertex of attack: The intruder chooses the vertex of attack according to some probability distribution defined in the model. The probability of launching an attack on vertex i denoted as $\mathbb{P}[\texttt{attack} = i]$, and $\sum_{i \in V} \mathbb{P}[\texttt{attack} = i] = 1$
- Time of attack: This defines the vertex at which the patrolling robot is when the attack is launched. The intruder may wait until the robot reaches a certain vertex before launching an attack. In general, this will be a function of the vertex of attack.

We can find the probability of an attack being successful with the help of *success matrix* defined in the following section.

3.2.1 Success Matrix

The Markov chain transition matrix P can be used to determine the probability of launching a successful attack on vertex j when the patrolling robot is at vertex i. For a given Markov chain, we can calculate the probability of going from state i to state j in exactly k steps. This probability is computed in [25]. The time taken in each transition is called a step, therefore, these results will be valid for the Markov chains on unweighted graphs. We extend these results for graphs with edge weights where each transition can take a different amount of time. Unweighted graphs can model the environments in the form of grid graphs, whereas weighted graphs capture the more general problem since the travel times between different nodes of the environment can be different.

First Visit Time for Unweighted Graphs

The following result gives an iterative method to calculate the probability of first visit from vertex i to vertex j. We have included the proof because the result on first visit time for weighted graphs relies on the extension of this proof.

Lemma 3.2.1. *[17] The probability of the first visit to the state $j \in V$ from the starting state $i \in V$ in k steps is given by*

$$F_k(i,j) = \begin{cases} p_{ij}, & k = 1 \\ \sum_{h \neq j} P_{ih} F_{k-1}(h,j), & k \geq 2 \end{cases} \tag{3.1}$$

Proof. For $k = 1$, $F_1(i,j)$ is simply the probability of going from j to i in one step, which by definition of P is p_{ij}.
For $k \geq 2$, let X_0 be the starting state of the chain and X_m be the state of the chain after m steps, then

$$\begin{aligned} F_k(i,j) &= \mathbb{P}\left[X_1 \neq j, \ldots, X_{k-1} \neq j, X_k = j | X_0 = i\right] \\ &= \sum_{h \neq j} \mathbb{P}\left[X_1 = h | X_0 = i\right] \mathbb{P}\left[X_2 \neq j, \ldots, X_{k-1} \neq j, X_k = j | X_0 = i, X_1 = h\right] \\ &= \sum_{h \neq j} p_{ih} \mathbb{P}\left[X_1 \neq j, \ldots, X_{k-2} \neq j, X_{k-1} = j | X_0 = h\right] \end{aligned}$$

because of the property of stationary Markov chains. □

This calculation can be done using matrix multiplication as

$$F_k = P(F_{k-1} - F_{k-1,d})$$

where $F_{k-1,d} = \text{diag}(F_{k-1}(1,1), F_{k-1}(2,2), \ldots, F_{k-1}(n,n))$.

First Visit Time for Weighted Graphs

The expression for the probability of first visit to a state given the starting state for weighted graphs can be extended from the unweighted graphs case. We will assume that the weights on the edges of the graphs are integers i.e. the graph is given as $G = (V, E, w)$ where $w : E \to \mathbb{N}_{>0}$. So, we can calculate the F_k matrices for any integer value of $k > 0$. Now, instead of writing steps taken from state i to state j, we write time taken from state i to state j, where the time taken along one edge is a positive integer.

Given a graph $G = (V, E)$, edge weights W and the transition matrix P, The probability mass function of the time taken for the first visit from state i to state j is given by

$$F_k(i,j) = \begin{cases} p_{ij}(w_{ij} = 1), & k = 1 \\ \sum_{h \neq j} P_{ih} F_{k-w_{ih}}(h,j) + p_{ij}(w_{ij} = k), & k \geq 2 \end{cases} \tag{3.2}$$

where $(w_{ij} = 1)$ is a truth statement which evaluates to 1 or 0, and $F_k(i,j) = 0$ for non positive values of k.

Given a starting state i and a goal state j, the elements $F_k(i,j)$ give a probability mass function over the random variable K which is the amount of time/steps taken from in hitting the state j for the first time starting from i. Hence,

$$\sum_{k=1}^{\infty} F_k(i,j) = 1, \quad \forall i, j \in \{1, 2, \ldots, n\}.$$

In the subsequent sections, we will refer to the three dimensional matrix F as the *first visit matrix*. Most of the analysis and discussion will depend on F, but we will not need to explicitly mention whether the first visit matrix is for a weighted or unweighted graph.

Success Matrix

We will now use the first visit matrix F to calculate the probability of successful attacks by an intruder on the environment. An attack on a particular location is successful if the patrolling robot does not visit that location for the entire duration of the attack. So, for a specified length of attack l, the probability of attack being successful given the patrolling robot's position is i at the time of launching the attack can be simply calculated as

$$\mathbb{P}\left[\texttt{success at } j|i\right] = 1 - \sum_{k=1}^{l} F_k(i,j)$$

which is the probability of not visiting the state j in l consecutive steps from the state i. For notational convenience, we define the *success matrix* S, whose element s_{ij} gives the probability of a successful attack at vertex j of the graph if the robot is at vertex i at the time the attack is launched.

$$s_{ij} := \mathbb{P}\left[\texttt{success at } j|i\right]$$

Hence, we can write the matrix S as

$$S = J - \sum_{k=1}^{l} F_k \tag{3.3}$$

where J is a matrix of all ones. The matrix S may or may not be used by the intruders to decide on the location and the time of the attack. The following sections present some intruder models.

3.2.2 Attack Model 1: Naïve Attacks

The naïve attack model represents a naïve intruder that does not learn or use any information about the patrolling path of the robot. The time and vertex of attack for this model are decided as follows.

Vertex of attack: Whenever an attack is to be launched, the vertex of attack is chosen depending on the importance of that vertex. So, the probability of attacking vertex j is given by

$$\mathbb{P}\left[\texttt{attack} = j\right] = \frac{\phi_j}{\sum_i \phi_i}$$

Time of attack: In this model, the intruder launches the attack randomly and independent of robot's location. The location of the robot at the time of attack is a random variable, and the probability that the robot is at location i when the attack arrives is given by the frequency with which robot visits location i, which is π_i where π is the stationary distribution of the Markov chain being followed by the robot. Using the expression $\mathbb{P}[\texttt{robot at } i]$ for the probability of the robot being at i when the attack is launched, we can write

$$\mathbb{P}[\texttt{robot at } i] = \pi_i.$$

Objective function: Using the time and vertex of attack, the reward function for this intruder model can be written as

$$\begin{aligned} f(P) &= \sum_{j=1}^{n} \phi_j \mathbb{P}[\texttt{success at } j]\, \mathbb{P}[\texttt{attack} = j] \\ &= \sum_{j=1}^{n} \phi_j \mathbb{P}[\texttt{success at } j]\, \phi_j \end{aligned}$$

assuming ϕ is normalized. Then using Bayes theorem,

$$\begin{aligned} f(P) &= \sum_{j=1}^{n} \phi_j^2 \sum_{i=1}^{n} \mathbb{P}[\texttt{success at } j | i]\, \mathbb{P}[\texttt{robot at } i] \\ &= \sum_{j=1}^{n} \phi_j^2 \sum_{i=1}^{n} s_{ij} \pi_i \\ &= \pi^\top S \phi^2 \end{aligned}$$

where π and ϕ are vectors. The reward function is a function of the transition matrix P, since the matrix S and the vector π are functions of P.

3.2.3 Attack Model 2: Deterministic Intruder

The deterministic intruder model is motivated by the intruder that attacks a vertex immediately after the patrolling robot departs that vertex. This is an effective strategy if the patrolling robot is following a cyclic tour of the graph.

Vertex of attack: The vertex of attack is chosen depending on the importance of that vertex just like the naïve policy, so

$$\mathbb{P}\left[\texttt{attack} = j\right] = \frac{\phi_j}{\sum_i \phi_i}.$$

Time of attack: The intruder waits and launches the attack on vertex i as soon as the patrolling robot leaves vertex i on its path. Hence,

$$\mathbb{P}\left[\texttt{robot at } i\right] = \begin{cases} 1 & \text{attack is on vertex } i \\ 0 & \text{otherwise} \end{cases}$$

Objective function: The reward function for such an intruder will be

$$f(P) = \sum_{i=1}^{n} \phi_i^2 s_{ii}.$$

If the patrolling path of the robot is a TSP tour and the length of the tour is greater than the length of attack, then this attack policy is optimal in terms of choosing the time of attack.

3.2.4 Attack Model 3: Intelligent Intruder with Assigned Locations

Let us consider an intruder that has observed the patrolling path of the robot and wants to use the learned information to increase the chances of a successful attack. The models presented in this and the next section address such intruders. We will assume that the intruder has observed the surveillance path long enough to have modeled the transition matrix P of the patrolling Markov chain. The learning of the Markov chain is discussed in Section 5.1.1.

Vertex of attack: According to this model, the vertex to attack is selected based on its importance like the previous two models. So, the intruder does not use the patrolling path information to choose the vertex of attack. Hence,

$$\mathbb{P}\left[\texttt{attack} = j\right] = \frac{\phi_j}{\sum_i \phi_i}.$$

This can be realized in a scenario where the intruder has been pre-assigned the locations that it has to attack, and the intruder then uses the information about the patrolling path to choose the time of attack for those locations.

Time of attack: Since the intruder has access to the transition matrix P, it can calculate the success matrix S (3.3) and use it to figure out the best possible time to attack. The element s_{ij} gives the probability of a successful attack on j if the attack is launched when the robot is at vertex i. The intruder wants to maximize its chances of a successful attack at vertex j, so it attacks when the robot is at vertex i such that

$$i = \arg\max_x s_{xj}.$$

So, given the vertex j to attack, the location of the robot at the time of attack is given by

$$\mathbb{P}\left[\texttt{robot at } i\right] = \begin{cases} 1, & i = \arg\max_x s_{xj}. \\ 0, & \text{otherwise} \end{cases}$$

Hence, the intruder looks at the column of S corresponding to the vertex it has to attack and the index of the maximum element in that column gives the vertex at which the robot should be when that attack is launched. The intruder waits until the robot reaches vertex i to launch the attack at j.

Objective function: The expected reward for the intruder following this attack model will be

$$\begin{aligned} f(P) &= \sum_{j=1}^{n} \phi_j^2 \sum_{i=1}^{n} \mathbb{P}\left[\texttt{success at } j|i\right] \mathbb{P}\left[\texttt{robot at } i\right] \\ &= \sum_{j=1}^{n} \phi_j^2 \max_i \{s_{ij}\} \\ &= \max(S)\phi^2 \end{aligned}$$

where $\max(S)$ is a row vector with j^{th} element being the maximum entry in the j^{th} column of matrix S.

3.2.5 Attack Model 4: Intelligent Intruder

This model also uses the transition matrix P learned by observing the patrolling path. Unlike the previous model, the intruder has not been assigned the locations to attack.

Instead, the intruder has to maximize its reward from the environment and hence this model allows the intruder to choose the vertex of attack as well.

Vertex of attack: The intruder has the matrix S calculated using the transition matrix, and the maximum expected reward the intruder can obtain is by attacking the best possible vertex at the best possible time. The vertex with the maximum expected reward is given by

$$j = \arg\max_{y} \left\{ \phi_y \max_{i} \{s_{iy}\} \right\}.$$

Here $\max_i\{s_{iy}\}$ gives the highest probability of attack on vertex y. So, $\phi_y \max_i\{s_{iy}\}$ gives the expected reward for attacking the vertex y and the vertex to attack j is chosen as the vertex that gives the most expected reward.

Time of attack: The best time to attack vertex j is given by the index of the maximum entry in the j^{th} column of S. The position of robot at the time of attack is then given by

$$i = \arg\max_{x} s_{xj}.$$

Objective function: So, the maximum expected reward for the intruder is

$$\begin{aligned} f(P) &= \max_{j} \left\{ \phi_j \max_{i} \{s_{ij}\} \right\} \\ &= \max_{i,j} \{\phi_j s_{ij}\} \end{aligned}$$

Among the attack models that decide the time and vertex of attack based on the patrolling policy P only, this attack model is optimal in the sense that it maximizes the intruder's reward. Given any patrolling policy P, the expected reward of any of the given intruders can not be greater than that of the intelligent intruder.

The summary of the intruder attack models is provided in Table 3.1. We presented the reward functions for some of the intruder attack models in this section. The expected reward of the intruder is a function of patrolling policy P and we hope to find a transition matrix which gives a reasonably low value for the expected intruder reward. In the next section we look at some possible ways to find a paroling strategy for the robot.

3.3 Patrolling Policy

The objective of patrolling is to minimize the loss incurred by the intrusion on the environment. Given one of the intruder models, our task is to find the patrolling policy P which

Intruder model	Vertex of attack $\mathbb{P}[\texttt{attack at } j]$	Time of attack $\mathbb{P}[\texttt{robot at } i]$	Objective function $f(P)$
1	$\frac{\phi_j}{\sum_i \phi_i}$	π_i	$\pi^\top S\phi^2$
2	$\frac{\phi_j}{\sum_i \phi_i}$	$\begin{cases} 1, & \text{attack on } i \\ 0, & \text{otherwise} \end{cases}$	$\sum_{i=1}^n \phi_i^2 s_{ii}$
3	$\frac{\phi_j}{\sum_i \phi_i}$	$\begin{cases} 1, & i = \arg\max_x s_{xj}. \\ 0, & \text{otherwise} \end{cases}$	$\max(S)\phi^2$
4	$\begin{cases} 1, & j = \arg\max_y \left\{\phi_y \max_i\{s_{iy}\}\right\} \\ 0, & \text{otherwise} \end{cases}$	$\begin{cases} 1, & i = \arg\max_x s_{xj}. \\ 0, & \text{otherwise} \end{cases}$	$\max_{i,j}\{\phi_j s_{ij}\}$

Table 3.1: Summary of the attack models

minimizes the expected reward for that intruder. So, we seek to minimize the objective function $f(P)$ that is given for each intruder model. In this section, $f(P)$ will generally refer to the reward function independent of the attack model unless a model is specified.

The Optimization Problem

The transition matrix P is a stochastic matrix defined on the graph $G = (V, E)$. The optimization problem is to minimize the objective function $f(P)$ for the given intruder model, over the constrained set of transition matrices. The objective function $f(P)$ operates on the transition matrices P of Markov chains. The argument P of the function has the properties that $p_{ij} \geq 0, \forall i, j \in \{1, \ldots, n\}$ and $\sum_j p_{ij} = 1, \forall i \in \{1, \ldots, n\}$. Moreover, if edge $\{i, j\}$ is not in the edge set of the graph then $p_{ij} = 0$. So the domain of the objective function is the closed convex set

$$\mathfrak{C} = \{P \in \mathbb{R}^{n\times n} : p_{ij} \geq 0, \sum_j p_{ij} = 1, \{i, j\} \notin E \implies p_{ij} = 0\}$$

Note that if a chain has more than one communicating class, we can decompose that chain into separate single communicating class Markov chains. Then the reward for each of those Markov chains can be calculated and the minimum of those rewards can be used as the objective value. Then the single communicating class Markov chain that gave the minimum value of the reward will govern the patrolling path of the robot. So, although the Markov chains of our interest are the chains with a single communicating class, the domain of the objective function can be the set of all transition matrices on the given graph.

The optimization problem is to minimize $f : \mathfrak{C} \subset \mathbb{R}^{n\times n} \to \mathbb{R}_{\geq 0}$. The optimization problem can be written as

$$\begin{aligned}
&\underset{P}{\texttt{minimize}} \quad f(P) \\
&\texttt{subject to} \\
&\qquad \sum_{j=1}^{n} p_{ij} = 1 \quad \textbf{for all } i \\
&\qquad p_{ij} \geq 0 \quad \text{for all } i, j \\
&\qquad p_{ij} = 0 \quad \text{for } \{i,j\} \notin E.
\end{aligned} \tag{3.4}$$

given the graph $G = (V, E, w)$, and the importance ϕ_i for each vertex i.

Non Convexity of the Problem

The optimization problem 3.4 is not, in general, a convex problem. The space of allowed transition matrices is a convex set, but the objective function for any of the intruders is not a convex function in general. The following example demonstrates this.

Example 3.3.1. *The element s_{ij} of the success matrix is a function of the problem variables P_{xy} for $i, j, x, y, \in \{1, \ldots, n\}$. Consider the problem with length of attack $l = 2$. Then*

$$\begin{aligned}
s_{ij} &= 1 - F_1(i,j) - F_2(i,j) \\
\frac{\partial^2 s_{ij}}{\partial p_{xy}\partial p_{kl}} &= \frac{-\partial^2 F_1(i,j)}{\partial p_{xy}\partial p_{kl}} - \frac{\partial^2 F_2(i,j)}{\partial p_{xy}\partial p_{kl}}
\end{aligned}$$

Consider s_{11} and the variables p_{12} and p_{21} as the first two variables in the indexing of the Hessian of s_{11}. Then

$$\begin{aligned}
\frac{\partial^2 F_1(1,1)}{\partial p_{12}\partial p_{21}} + \frac{\partial^2 F_2(1,1)}{\partial p_{12}\partial p_{21}} &= \frac{\partial^2 p_{11}}{\partial p_{12}\partial p_{12}} + \frac{\partial^2 (p_{12}p_{21} + \sum_{h=3}^{n} p_{1h}p_{h1})}{\partial p_{12}\partial p_{21}} \\
&= 1
\end{aligned}$$

and

$$\frac{\partial^2 F_1(1,1)}{\partial p_{12}\partial p_{12}} + \frac{\partial^2 F_2(1,1)}{\partial p_{12}\partial p_{12}} = \frac{\partial^2 F_1(1,1)}{\partial p_{21}\partial p_{21}} + \frac{\partial^2 F_2(1,1)}{\partial p_{21}\partial p_{21}} = 0$$

So, the first and the second principal minor of the Hessian of s_{11} evaluates to 0 and -1 respectively. So, s_{11} is not a convex function of P and hence $f(P)$ is not in general a convex function of P.

Hence we cannot use convex programming to find the global optimal patrolling policy P. Therefore we will look into nonlinear optimization techniques to locally minimize the intruder reward function. But let us first look at an interesting result about the optimal patrolling policy for the intruder model given in Section 3.2.4.

3.3.1 Deterministic Patrolling Policy for Intelligent Intruder with Assigned Locations

The motivation behind randomizing the patrolling path is that if the path were deterministic, then this information could be exploited by the intruder. However, in the case where the intruder is following attack model 3, a deterministic path minimizes the expected reward. Informally, that is because the intruder has to attack all the vertices according to their importance, and if the patrolling robot just visits the 'most important' vertices with tour length less than l, the intruder can never successfully attack those vertices. The following proposition formally presents the result.

Proposition 3.3.2. *The solution to the* ORIENTEERING PROBLEM *for the graph $G = (V, E, w)$ with score on vertex $i \in V$ given by $\psi_i = \phi_i^2$, and the time limit $T_{max} = l$ minimizes the expected reward for the intruder following attack model 3 (intelligent intruder with assigned locations).*

Proof. The intruder under consideration can only choose the time of attack to increase the chances of successful attack. The vertex of attack is decided by the weights of the vertices. Therefore, we can construct the same objective function for an intruder that picks the vertex to attack uniformly at random, by changing the weights on the vertices of the graph to $\psi = \phi^2/n$.

$$\begin{aligned} f =& \sum_{j=1}^{n} \phi_j^2 \mathbb{P}\left[\texttt{success at } j\right] \mathbb{P}\left[\texttt{attack} = j\right] \\ =& \sum_{j=1}^{n} \frac{\phi_j^2 \max_i\{s_{ij}\}}{n} \\ =& \max(S)\psi \end{aligned}$$

We get the same reward function with ϕ^2 replaced by ϕ, and the two cases are equivalent. So, each vertex of the graph is attacked with equal probability.

Let the vertices belonging to solution of ORIENTEERING PROBLEM be $U \subseteq V$. Since the robot will follow a deterministic tour of vertices in U and the tour time is not greater than the length of attack l, the probability of successful attack on the vertices of the tour is 0, whereas that of the vertices not in tour is 1. So, $s_{ij} = 0, \forall i \in \{1, \ldots, n\}, j \in U$, and $s_{ij} = 1, \forall i \in \{1, \ldots, n\}, j \in V \setminus U$. So, the value of reward for the intruder will be

$$f = \sum_{j \in V \setminus U} \psi_j$$

So, trivially any random path which visits only the vertices in U cannot perform better than the orienteering tour.

Now assume that there is a vertex $r \in V \setminus U$ outside the orienteering tour by visiting which the robot can decrease the intruder's reward. Consider the suboptimal attack time selection strategy by the intruder to attack vertex r and each vertex in U when the robot leaves vertex r. Then the probability of successful attack on r is s_{rr}. Moreover, whenever the robot comes back to r from r in less than l time(with a probability of $1 - s_{rr}$), it is sure to miss the vertices in U with a collective score of $\psi_{missed} \geq \psi_r$, by the definition of orienteering tour. So the reward among the vertices $U \cup r$ which was originally ψ_r becomes

$$\begin{aligned} f' &= s_{rr}\psi_r + (1 - s_{rr})\psi_{missed} \\ &\geq s_{rr}\psi_r + (1 - s_{rr})\psi_r \\ &\geq \psi_r \end{aligned}$$

Hence, by visiting the vertex outside the orienteering tour, the reward of the intruder following a suboptimal strategy can only increase. Therefore, it cannot decrease for the intruder following the optimal attack time strategy. So, the deterministic path in the orienteering tour minimizes the objective function. □

Note that the deterministic path may or may not be written as a Markov chain. So, this optimal strategy does not necessarily fall into the set of the policies over which we have defined our optimization problem. However, the objective function is well defined for the deterministic tours, where the probability of successful attack on a vertex is given as

$$\mathbb{P}\left[\texttt{success at j}\right] = \begin{cases} 1, & \tau_j > l \\ 0, & \tau_j \leq l \end{cases}$$

where τ_j is the maximum time between consecutive visits to vertex j. So, the result presented above shows that a deterministic strategy can minimize the expected reward of

a certain type of intruder and it also shows that the deterministic path which minimizes the reward is given by orienteering tour.

3.4 Numerical Optimization of the Objective Function

Finding the optimal policy P for patrolling given an attack model is a constrained optimization problem where the function $f(P)$ is a nonlinear and non convex function. Therefore, to optimize the function over P, we employ numerical optimization methods. In this section we investigate several numerical optimization techniques and tailor them for our function. We then advocate the use of pattern search method for the optimization of intruders reward.

3.4.1 Gradient Projection Algorithm

The gradient descent algorithm is one of the common methods used for the optimization of continuous differentiable functions. The basic idea is to move along the direction of negative gradient of the function at each step until the gradient becomes zero. In the case of constrained optimization where the feasible region is a convex set, the gradient projection method can be used [10]. The gradient descent or projection algorithms require the gradient of the function to be defined in the feasible set. Our objective function is non smooth because it picks the maximum element among the columns of S. In these cases when the objective function is not continuously differentiable, we can use the generalized gradient descent for locally Lipschitz functions. The following result shows that the objective functions that we are dealing with are locally Lipschitz.

Lemma 3.4.1. *Let $f_h : \mathbb{R}^d \to \mathbb{R}$ be locally Lipschitz functions at $x \in \mathbb{R}^d$ for $h = \{1, \ldots, m\}$ then $f_{\max}(x) = \max\{f_h(x)\}$ is locally Lipschitz at x [18].*

Proposition 3.4.2. *The objective functions $f : \mathfrak{C} \subset \mathbb{R}^{n \times n} \to \mathbb{R}$ for intruder models* 1,2,3 *and* 4 *are locally Lipschitz at each $P \in \mathfrak{C}$.*

Proof. Each element $F_k(i, j)$ is a polynomial in the variables p_{xy} for $x, y \in \{1, \ldots, n\}$. So, $F_k(i, j)$ is continuously differentiable with respect to the transition probabilities of the Markov chain. Therefore, the functions $s_{ij} : \mathfrak{C} \to \mathbb{R}$ are continuously differentiable and hence locally Lipschitz at any $P \in \mathfrak{C}$. Applying lemma 3.4.1 on the functions s_{ij} proves that the objective functions under consideration are locally Lipschitz. □

So, the objective function for any given intruder model is locally Lipschitz and hence we can define the generalized gradient of $f(P)$ at each point P.

Definition 3.4.3 (Generalized Gradient). *[19] The generalized gradient $\partial f : \mathbb{R}^d \to \mathfrak{B}(\mathbb{R}^d)$ of a locally Lipschitz function $f : \mathbb{R}^d \to \mathbb{R}$ is defined as*

$$\partial f(x) = co\{\lim_{i\to\infty} \nabla f(x_i) : x_i \to x, x_i \notin T \cup \Omega_f\}, \tag{3.5}$$

where co denotes convex hull, $\mathfrak{B}(\mathbb{R}^d)$ denotes the collection of subsets of $\mathbb{R}^d$, $\Omega_f \subset \mathbb{R}^d$ denotes the set of points where f is not differentiable and $T \subset \mathbb{R}^d$ is a set of measure zero that can be arbitrarily chosen to simplify the computation.

Informally, the generalized gradient of f at x is the convex combination of all the possible limits of the gradient at neighboring points of x where f is differentiable. Using this definition, we can define the gradient for our non-smooth objective function and use it for the gradient projection method. The gradient descent method uses the gradient ∇f where it is defined, since it gives the direction of steepest descent of the function. At the non-smooth points, where we have to consider the generalized gradient, the direction of maximum descent is given by the least norm element in the generalized gradient, denoted by $-Ln(\partial f)$ [19].

In the subsequent discussion of the gradient projection method, we will use the term *gradient* to refer to the actual gradient of the function when its differentiable and to the generalized gradient at the points where it is not. Let us now define the projection and then discuss the gradient projection method for our objective function.

Definition 3.4.4 (Projection on a Convex Set). *Given a point x in $\mathbb{R}^d$, the projection of point x on a closed convex set $X \subset \mathbb{R}$ is the point $[x]_X \in X$ which is at the minimum distance from x and is the optimal point of the following problem.*

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \|z - x\|^2 \\ \text{subject to} \quad & \\ & z \in X \end{aligned}$$

We will use the notation $[x]_X$ for the projection of point x on the set X.

An iteration of the gradient projection method is of the form [10]

$$P^{k+1} = P^k + \alpha^k(\bar{P}^k - P^k) \tag{3.6}$$

where

$$\bar{P}^k = [P^k - s^k \nabla f(P^k)]_{\mathfrak{C}}$$

The variables $\alpha^k \in (0, 1]$ and s^k are step sizes. So, the method moves current point P^k in the direction of negative gradient by the amount $s^k \nabla f(P^k)$ and hence moves in the direction of steepest descent. The resultant point may not be a feasible point, so it is projected back onto the set of transition matrices. The vector $\bar{P}^k - P^k$ is a feasible vector, as in moving along this vector with $\alpha^k \in (0, 1]$ will keep the point in the feasible set.

Proposition 3.4.5. *The direction* $(\bar{P}^k - P^k)$ *given in* (3.6) *is a direction of descent for the function* $f(P)$.

Proof. By the projection theorem [10], a point $x^* \in X$ is equal to the projection $[z]_X$ of point z if and only if

$$(z - x^*)^\top (x - x^*) \leq 0, \quad \forall x \in X$$

since our function's domain is matrix, let us denote the dot product of two matrices A and B of same dimensions as $\langle A, B \rangle = \sum_{i,j} (a_{ij} b_{ij})$. Then, since the point $\bar{P}^k$ is the projection of $P^k - s^k \nabla f(P^k)$, using the projection theorem, we get

$$\langle P^k - s^k \nabla f(P^k) - \bar{P}^k, P - \bar{P}^k) \rangle \quad \leq 0, \quad \forall P \in \mathfrak{C}$$

Plugging in $P = P^k$, we obtain

$$\langle \nabla f(P^k), P^k - \bar{P}^k \rangle \quad \leq -\frac{1}{s^k} \langle P^k - \bar{P}^k, P^k - \bar{P}^k \rangle$$

Since $\langle P^k - \bar{P}^k, P^k - \bar{P}^k \rangle$ is the norm of the vector $P^k - \bar{P}$, it is always positive and hence the dot product of the direction $(\bar{P}^k - P^k)$ with the gradient is negative, implying that it is a direction of descent of the function. □

So, taking the projection after moving in the direction of steepest descent and then moving towards the projected point decreases the function. Hence the method will terminate if the projected point is the same as the current point, which means that the gradient when projected onto the set is 0.

At each iteration, the resultant point after moving in the direction of gradient may not be a feasible point and hence it needs to be projected onto the set of allowed transition matrices. The following section gives a method to project any point $\tilde{P} \in \mathbb{R}^{n \times n}$ onto the set $\mathfrak{C}$.

Projection Onto the Set of Allowed Transition Matrices

The projection onto the convex set can be written as the following optimization problem.

$$\begin{aligned} \underset{\bar{P}}{\texttt{minimize}} \quad & \|\texttt{vec}(\bar{P}) - \texttt{vec}(\tilde{P})\| \\ \texttt{subject to} \quad & \\ & \sum_{j=1}^{n} \bar{p}_{ij} = 1 \quad \textbf{for all } i \\ & \bar{p}_{ij} \geq 0 \quad \text{for all } i, j \\ & \bar{p}_{ij} = 0 \quad \text{for } \{i,j\} \notin E. \end{aligned}$$

where $\texttt{vec}(P)$ vectorizes the matrix $P \in \mathbb{R}^{n \times n}$ to a vector in $\mathbb{R}^{n^2}$. Note that

$$\|\texttt{vec}(\bar{P}) - \texttt{vec}(\tilde{P})\|^2 = \sum_{i=1}^{n} \|\bar{P}_i - \tilde{P}_i\|^2$$

where P_i is the i^{th} row of matrix P. Moreover, the constraints of a row are independent of the constraints of the other rows. Hence, we can project each row i of the matrix $\tilde{P}$ separately.

Each row i of the transition matrix P should lie in the q_i dimensional simplex $\sum_j p_{ij} = 1, p_{ij} \geq 0$, where q_i is the degree of vertex i. So, we can use the following simple algorithm for the projection of each row of the point onto the simplex [22].

Let v_j be the elements of the vector to be projected onto the simplex. In our case they will be the elements of a row i of the point $\tilde{P}$ such that $(i,j) \in E$. Then the corresponding projected vector w_j is given by

$$w_j = \max\{v_j - \theta, 0\}$$

where θ is found as follows. Sort v into μ, i.e. $\mu_1 \geq \mu_2 \geq ... \geq \mu_{q_i}$, and evaluate

$$\rho = \max\Big\{ j \in \{1, 2, ...q_i\} : \mu_j - \frac{1}{j}\big(\sum_{r=1}^{j} \mu_r - 1\big) > 0 \Big\}$$

$$\theta = \frac{1}{\rho}\big(\sum_{r=1}^{\rho} \mu_r - 1\big).$$

So, we can project each row i of the matrix onto the q_i dimensional simplex using this algorithm. This method is efficient as compared to the general convex optimization since it employs simple vector operations and its runtime is dominated by sorting the elements of v.

Complexity of Gradient Projection Method: Calculating the gradient of the function at each iteration in gradient descent algorithm is too expensive. If we assume matrix multiplication of two $n \times n$ matrices takes $O(n^3)$ time, then the gradient calculation for one element of S with respect to P takes $O(n^5 l)$ time if the graph has $O(n^2)$ edges. That is because we have to take derivative of $F_k(i, j)$ for all $i, j \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, l\}$. So, we have to calculate the derivative of $O(n^2 l)$ elements. Each of these derivatives involves matrix multiplication and hence the total runtime becomes $O(n^5 l)$. For planar graphs, with $O(n)$ edges, the runtime for gradient calculation is $O(n^4 l)$. This is quite inefficient and therefore, we look at some gradient free methods to optimize our objective function.

3.4.2 Gradient Free methods

The following optimization methods do not require the calculation of gradient of the function.

Approximating the Gradient and Nelder Mead Method

One way to avoid gradient calculation of function f is to approximate the value of ∇f by evaluating the function at $n + 1$ points where n is the dimension of the function's domain. The basic idea is to use the Taylor's theorem.

$$f(x + p) \approx f(x) + \nabla f(x)^\top p$$

So, using this we get that partial derivative of f is approximated by [31]

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x + \epsilon e_i) - f(x)}{\epsilon}$$

where e_i is the unit direction vector.

Another method of derivative free optimization is called Nelder-Mead method [31]. In this method, we keep track of $n + 1$ points, whose convex hull forms a simplex. In each

iteration, the vertex of the simplex with the worst function value is replaced by a better point or we shrink the simplex around the best available point.

However, in our case, these methods are also inefficient. We seek gradient free methods because calculating the gradient is too expensive for our objective function. Both of these methods require the calculation of the objective function at n^2 points for our case(the objective function is defined on a $n \times n$ matrix, where n is the number of vertices in the graph). The calculation of the function in the general case takes $O(n^3 l)$ time and therefore, each iteration of these methods requires $O(n^5 l)$ time, which is the same as gradient projection method.

Pattern Search

At each iteration of the pattern search method, a set of search directions are chosen and the function is evaluated at a given step length along these direction. As soon as a better point is found, it is chosen as the current point and the method proceeds to the next iteration.

Let $\mathcal{D}_k$ denote the set of possible directions to chose from at k^{th} iteration and let the step length at iteration k is given by γ_k. The algorithm 1 gives the pattern search method [31].

Algorithm 1 Pattern Search Method

Pick starting point $P \in \mathfrak{C} \subset \mathbb{R}^{n \times n}$ and γ_o
for $k = 1, 2, \ldots$ **do**
    if $\gamma_k < \gamma_t$ **then**
        stop
    end if
    if $f(P_k + \gamma_k d_k) < f(P_k) - \rho(\gamma_k)$ for some $d_k \in \mathcal{D}_k$ **then**
        $P_{k+1} \leftarrow P_k + \gamma_k d_k$
        $\gamma_{k+1} \leftarrow \phi_k \gamma_k$
    else
        $P_{k+1} \leftarrow P_k$
        $\gamma_{k+1} \leftarrow \theta_k \gamma_k$
    end if
end for

In the algorithm, $\phi_k > 1$ is used to increase step length for the next iteration if a decrease direction is found. The parameter $\theta_k < 1$ decreases the step length if no decrease direction is found in the said iteration. If the step length becomes less than a set value

γ_t, the algorithm terminates. The sufficient decrease function $\rho(\gamma_k)$ is chosen to be $M\gamma_k^{3/2}$ where M is a constant, as suggested by [31] for the algorithm to converge.

The set of directions should have the property that at least one of the directions should be a direction of descent whenever the gradient of the function is not zero. So, a proposed set is the set of all unit directions. In our case, the optimization is constrained to the set of stochastic matrices. So, we define the set $\mathcal{D}_k$ as follows. For each row i of the transition matrix, we have a simplex in $\mathbb{R}^{q_i}$ dimensions where q_i is the number of neighbors of vertex i. So, we can define $q_i - 1$ mutually perpendicular directions on the simplex for each i and use them as the possible directions. In our implementation, we also include some random directions in $\mathcal{D}_k$.

Pattern search method moves to the next iteration as soon as a better point is found instead of calculating the objective function exactly n^2 times for each iteration. Hence one iteration of pattern search is less expensive then that of the methods discussed earlier. On the other hand, the gradient projection method may require fewer iterations to converge although each iteration is much more expensive due to the calculation of the gradient. We used both the gradient projection and pattern search methods to find the patrolling policies for a variety of graphs and found the pattern search method to be much faster in practice.

3.5 Experimental Results

We performed the optimization of the expected reward for a given intruder using pattern search method discussed in Section 3.4.2 on some instances of the problem. This section presents and examines the results of those experiments.

Random problem instances

The sample instances of the graph were created randomly following a procedure similar to [37]. A graph of n vertices was constructed using n Euclidean points in a plane of size $MAX \times MAX$. Each point was at least MIN distance apart from all the other points. Each point was connected to k of its nearest points where k was chosen randomly between $NBR_{\texttt{min}}$ and $NBR_{\texttt{max}}$. Moreover, each point was connected to some other random points by a probability $P_{\texttt{conn}}$. The Euclidean distance between the connected points was calculated and then rounded off to give integer weights on the edges.

Starting point	Kemeny	best of 1000	Random	Uniform
Best point(for 40 runs)	3	8	27	2

Table 3.2: The table depicting the performance of the pattern search method from different starting points.

Comparison with other policies

In our first experiment, we ran the pattern search method on small instances of the graph for different starting points and compared the results with some other patrolling policies. The intruder model chosen was the 'intelligent intruder' given in 3.2.5 and so the objective function was the expected reward for the intelligent intruder. The graph had $n = 10$ vertices, placed in a plane of size $MAX = 10$. Each point was at least $MIN = 3$ units of distance apart from the others. The number of nearest neighbors was taken to be between $NBR_{\texttt{min}} = 3$ and $NBR_{\texttt{max}} = 5$. By a probability of $P_{\texttt{conn}} = 0.1$ each point was connected to some other random point as well. The wights on the vertices of the graph were randomly picked to be between 1 and 3. The length of the attack was taken to be the total length of the minimum spanning tree of the graph.

The experiment was performed for 40 different graphs, and for each of the graph instance, the pattern search was run from 4 different starting points mentioned below.

1. Kemeny starting point: This starting point was picked as a point with the minimum Kemeny constant. The point was calculated using a semidefinite optimization given in [2].
2. Best of 1000: This point was chosen out of 1000 randomly picked points in $\mathfrak{C}$ with the best value for the objective function.
3. Random: A randomly picked point in the domain $\mathfrak{C}$.
4. Uniform: The entry p_{ij} in each row i of the starting point P was picked to be $1/q_i$ where q_i is the number of neighbors of i^{th} vertex.

For each instance of the graph, the best starting point is given as the point from which the pattern search found the local optimal point with minimum value for the function. The performance of the method from different starting points is given in table 3.5 and it can be seen that a randomly picked point serves as the best starting point most of the time.

For each of the 40 instances of the graph, we also compared the performance of the policy(transition matrix) found by the pattern search method to the following two policies.

Policy	Minimum Kemeny Constant	Uniform	Pattern Search
1	1.42	1.56	1
2	1.21	1.43	1
3	1.36	1.41	1
4	1.32	1.34	1
5	1.56	1.86	1

Table 3.3: The expected reward of the intruder for different policies.

1. Minimum Kemeny Constant Policy: The policy given in [2] of finding the transition matrix with minimum Kemeny constant.

2. Uniform Policy: The transition matrix P is chosen such that the entry p_{ij} in each row i of P is $1/q_i$ where q_i is the number of neighbors of i^{th} vertex.

The pattern search method found a better policy for all of the 40 instances since it particularly minimizes the objective function. The comparison for 5 of those instances is given in table 3.5. The expected reward is normalized such that it is 1 for the pattern search method.

A randomly generated graph with 30 vertices is shown in Figure 3.1. The performance and convergence of the pattern search method on this graph is shown in Figure 3.2. The length of attack is half the length of minimum spanning tree of the graph, and the weights on the vertices are randomly chosen between 1 and 3. The plot is normalized such that the random starting point has the objective value of 1.

Patrolling in indoor environment using unweighted graph

For the next experiment we consider the intruder model presented in Section 3.2.4. Here the intruder cannot choose the vertex to attack but can choose the time to launch the attack. We consider an unweighted grid graph which can be used for patrolling in indoor environments. The graph has $n = 52$ vertices and the graph is shown in Figure 3.3. The weights on the vertices of the graph are 1 for all the vertices. The minimized expected rewards through pattern search are plotted against different lengths of attack in Figure 3.4. It can be seen that as the length of attack increases, the expected reward for the intruder decreases. A deterministic tour can cover all of the vertices in 58 steps and therefore for lengths of attack greater than 58, the expected reward of intruder can be made 0 by following a deterministic path. However, for the random path, the expected reward for

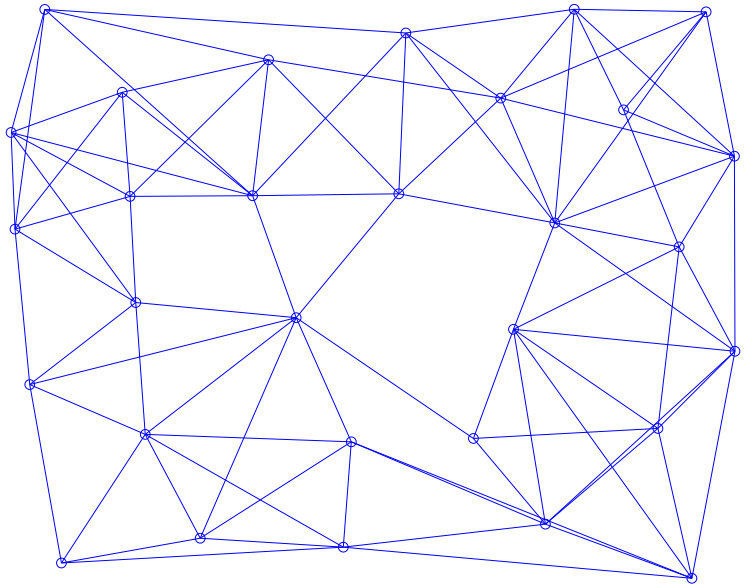

Figure 3.1: The randomly generated graph with 30 vertices.

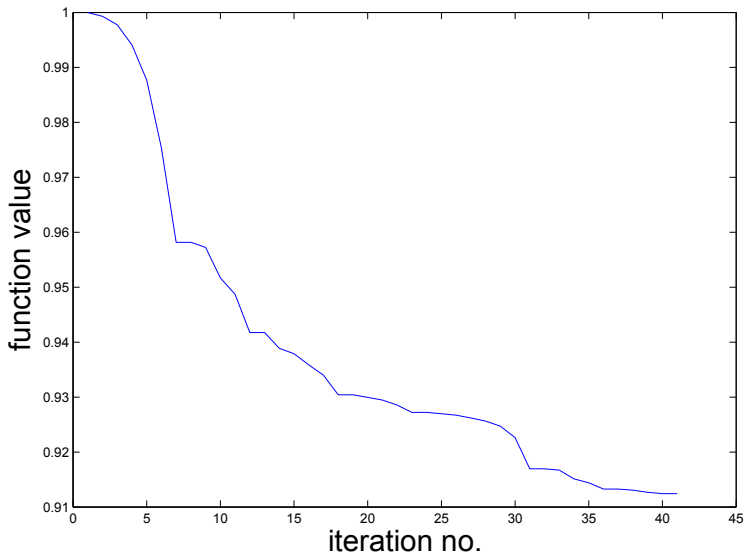

Figure 3.2: Performance of Pattern search on a random graph of 30 vertices.

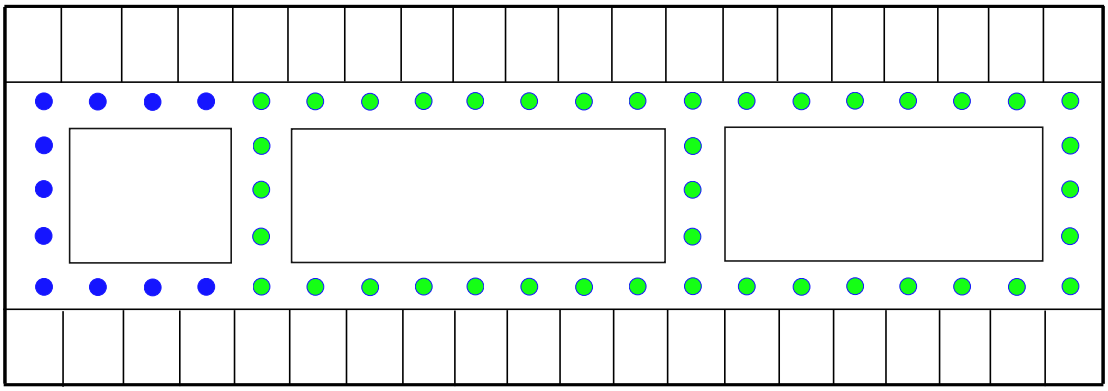

Figure 3.3: The grid graph representing an indoor environment to be monitored. The vertices of the graph are represented by discs. The green colored discs represent the vertices in the orienteering tour of length 44.

those lengths of attack is non zero. Hence, for very large lengths of attack, deterministic paths are obvious better choices.

In the Figure 3.5, we have plotted the performance of pattern search on the grid graph for length of attack $l = 44$. The starting point of the pattern search was a random Markov chain, and it can be observed that the expected reward for the intruder was almost 1 for that chain. It means that the intruder was always able to find a suitable time to attack for any of the vertices of the graph. Hence, for all the vertices of the graph, the probability of successful attack for the intruder following model 3 was almost 1. The pattern search decreased the expected reward of the intruder to 0.278. Notice that since we are considering the intruder model 3, the orienteering tour of the graph with time limit 44 is the optimal solution. The green vertices of the graph in Figure 3.3 represent the vertices in the orienteering tour. If the robot follows that deterministic path, the intruder can attack the blue vertices with 100% success rate and it can never successfully attack the green vertices. So the expected reward for the intruder will be $11/52 = 0.2115$.

Comparison of attack models

Now we compare the expected rewards of different attack models for a given patrolling policy. The policy P was determined by running pattern search for intruder model 4 with length of attack $l = 17$. For this experiment we considered a random weighted graph with $n = 20$ vertices. The rewards of the intruders are plotted against length of attack in

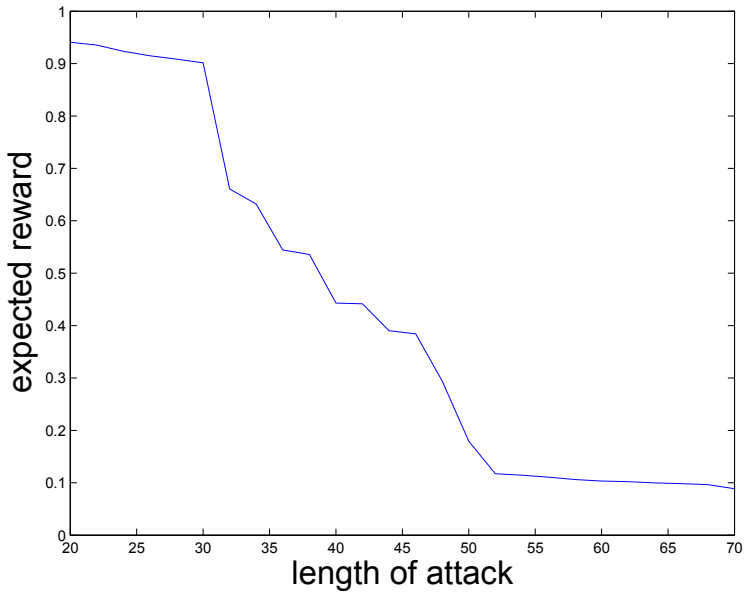

Figure 3.4: The expected reward for intruder following attack model 3 vs. length of attack for a grid graph.

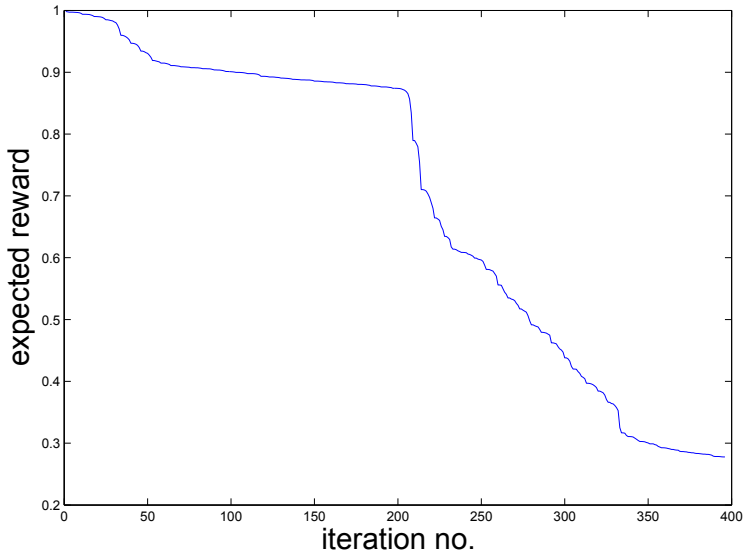

Figure 3.5: The performance of pattern search method for grid graph with $l = 44$.

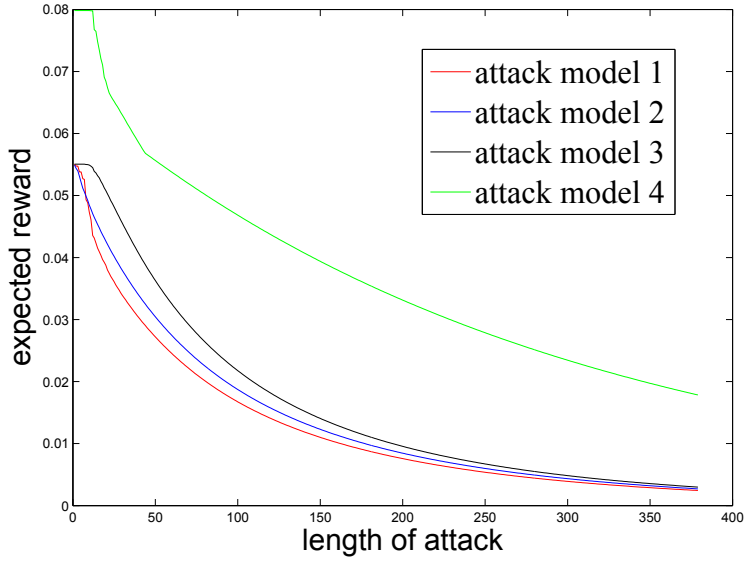

Figure 3.6: The expected reward of different attack models versus the length of attack.

Figure 3.6. Although the policy P was optimized for attack model 4 for $l = 17$, it still gives the maximum reward among the four models.

Chapter 4

Robot Monitoring for Detection and Confirmation of Stochastic Events

We define and study the Event Detection and Confirmation Problem in this chapter. The problem is formally defined in Section 4.1. In Section 4.2, we discuss the complexity of the off-line version of the problem. In Section 4.3 we derive the probability of successfully classification for a single vertex of the graph. We calculate the probability expression for the whole graph and propose a TSP based policy in Section 4.4. Section 4.5 presents the optimal spacing between two robots following the same tour on the graph. Finally, in Section 4.6, we give a working example of the problem.

4.1 Problem Statement

The Event Detection and Confirmation problem is defined on an undirected weighted graph $G = (V, E, w)$, where the vertex set $V = \{1, 2, \ldots, N\}$ represents the locations to be monitored, E is the set of edges connecting the vertices and $w : E \to \mathbb{R}$ represents edge weights. The edge weight w_{ij} on the edge $\{i, j\}$ can be thought as the distance the robot has to travel to go from i to j. We take the metric closure [5] of the graph G in order to obtain a complete graph, in which the length of each edge is equal to the shortest path distance in the original graph. Since we will be discussing the problem for this complete graph, we will refer to it as $G = (V, E, w)$ for simplicity.

The events arrive at each vertex $v \in V$ randomly and the time of arrival of events is not known to the patrolling robot even after the event has arrived. We model the arrival

of events at each vertex as a Poisson process with the arrival rate at vertex v given by λ_v. We assume that this arrival rate is known to the patrolling robot.The events stay active at a vertex for a random amount of time and can only be observed by the robot if the robot visits that vertex during the activity period of the event. We assume that the activity period of an event at vertex v is exponentially distributed with parameter μ_v. The Poisson distribution and the exponential distribution are often used to model customer arrival times and customer service times in queuing theory [27]. Most of the analysis in this chapter holds for more general distributions as well, however, the ability of obtain closed form expressions leverages these specific distributions.

The events are distinct and they can be identified uniquely by the robots. Moreover, only one event can be active at a vertex at a time and the next event can only arrive at that vertex after the previous one has gone inactive. The arrival times and the activity times of the events at different vertices are independent.

A critical time T is also the input to the problem which is used to define the following terms.

True Event: We call an event a true event if it remains active for an amount of time that is greater than or equal to T.

Detection: The detection of an event is the discovering of that event by a robot for the first time at its vertex.

Confirmation: Observing an event at a vertex at least T time later than detection of that event is called confirmation.

A robot, while on its patrolling path performs detection and confirmation of events. It can classify an event as true if and only if it confirms that event. Notice that if a true event becomes inactive before being confirmed by a robot, it cannot be classified as a true event.

When a robot reaches a vertex in its tour, it faces one of the following scenarios.

1. The vertex is empty (no event at the vertex): then, the robot can delete the event from its database which was recorded to be at that vertex (if any).

2. There is a new event at the vertex: then the robot stores it against that vertex with the current time stamp.

3. There is an event at the vertex which was detected at some previous check at that vertex: In this case the robot looks up the time stamp of that event and compares it with current time to see whether it is a true event or not.

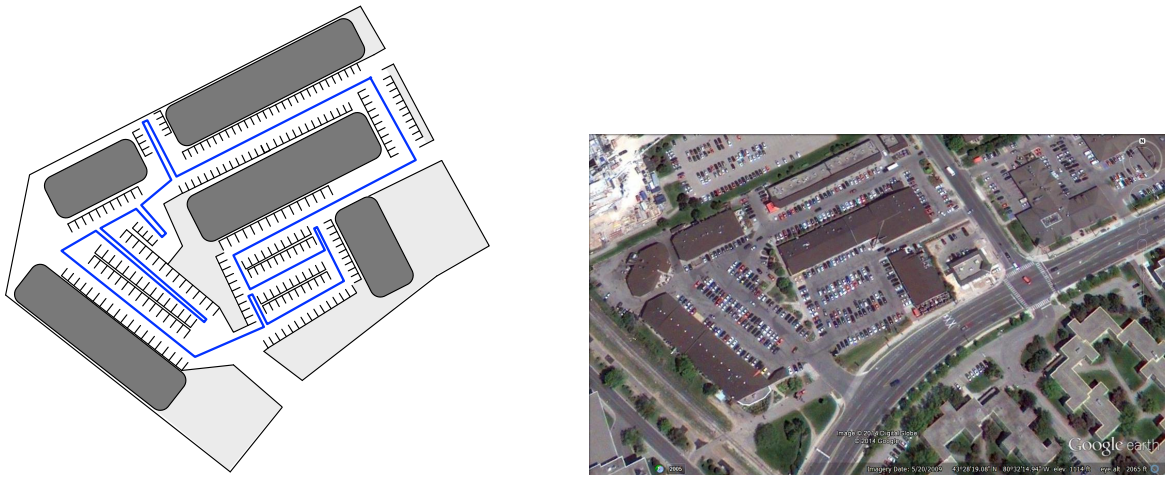

Figure 4.1: A parking lot environment and a possible patrolling tour based on the TSP.

Now, we can formally write the objective of the problem.

> **Event Detection and Confirmation Problem:** Find patrolling paths for the robots to minimize the probability of incorrectly classified events. The problem does not allow the robots to classify a false event as true, so the optimization task can be stated as maximizing the probability of correctly classifying true events.

The following example is helpful in depicting the features of the problem.

Example 4.1.1. *The problem is analogous to the problem of finding routes in a parking lot where cars are arriving and departing according to some random process and we have to ticket the cars staying more than the allowed parking time. The vertices of the graph represent individual parking spaces and the arrival and activity time of events at a vertex is equivalent to arrival and staying time of vehicles at a parking spot. The cars are identifiable by their license plates and there can be only one car at a spot. Moreover, the robot can ticket a car only if the time between its detection and ticketing is more than the allowed parking time. The cars that stay more than the allowed parking time but leave before a robot confirms it cannot be ticketed. An example parking lot environment and a possible patrolling route are shown in Figure 4.1. Here we assume that when moving along an aisle in the parking lot, the robot can view the parking spots on either side of the aisle.*

4.2 Complexity of the Problem

Let us now characterize the complexity of the problem. Let us consider the off-line version of the problem to understand the complexity of the problem.

4.2.1 Off-line Version

In the off-line version of the problem, all the required data is available beforehand. So the arrival times and activity periods of events at each vertex of the graph are known before devising a patrolling path. An input instance to the off-line problem is:

- An undirected weighted graph $G = (V, E, w)$.
- The critical time T.
- Total number of events M.
- The events $E_i, i \in \{1, 2, \ldots, M\}$ where each event is given by $E_i = (v, t_s, t_f), v \in V$. The vertex of the event is represented by v, and $[t_s, t_f]$ gives the interval during which the event remains active.

More than one event can arrive at a vertex, but their active intervals need to be disjoint. Given this data, the events with active times less than the minimum staying time T can be ignored. For the other(true) events with $t_f - t_s \geq T$, the monitoring robot needs to visit their vertex twice: once after their arrival to detect them and then before t_f but at least time T after the first visit to confirm them. This means that we can assign two time windows at each vertex and the robot has to visit each vertex during those time windows. We name the windows as detection window and confirmation window, given by

Detection Window: $[t_s, t_f - T]$

Confirmation Window: $[t_{det} + T, t_f]$ where $t_{det} \in [t_s, t_f - T]$ represents the actual visit time of vertex v by the robot during the first window.

Note that the confirmation window is dynamic in the sense that its length depends on the visit time during the detection window.

Now we show that the off-line version of the problem is NP-Complete.

4.2.2 Hardness of Off-line Problem

The following result shows that the off-line version of the Event Detection and Confirmation problem is NP-Complete.

Proposition 4.2.1 (Hardness of Offline Problem)**.** *The problem of finding a feasible tour for the off-line version of Event Detection and Confirmation Problem(EDC) is NP-Complete.*

Proof. The following reduction from the decision version of TRAVELING SALESMAN PROBLEM WITH TIME WINDOWS(TSPTW) to the off-line EDC shows that the off-line EDC problem is at least as hard as the problem of finding a feasible solution in TSPTW, which is known to be NP-Complete [38].

The input instance of TSPTW takes is

- An undirected weighted graph $G = (V', E', w')$.
- A time window for each vertex $i \in V'$, denoted by $[e_i, l_i]$.

The decision version of the problem is to find whether a tour exists which visits each vertex $i \in V'$ of the graph within the time window $[e_i, l_i]$ associated with that vertex.

Given an instance of TSPTW, we generate the following instance of off-line EDC.

$$\begin{aligned}(V, E, w) &= (V', E', w') \\ T &= \sum_{i,j} w_{ij} + \max_{i \in V'}\{l_i\} - \min_{i \in V'}\{e_i\} \\ E_i = (v, t_s, t_f) &= (i, e_i, l_i + T) \quad \forall i \in V'\end{aligned}$$

The detection window for the off-line EDC at vertex v will therefore be

$$[t_s, t_f - T] = [e_i, l_i].$$

If a patrolling path for off-line EDC exists, that means that the robot can visit each vertex in its detection window, which is same as the time window for TSPTW.

Similarly, if a feasible solution to the TSPTW exists, then the robot can visit all the vertices of EDC during their detection windows. Let that path be called 'detection path'. Then it can also visit all the vertices in their confirmation windows by following

the detection path with a time lag of T. This is possible because the robot will cover all the vertices in $t_1 \leq \max_i\{l_i\} - \min_i\{e_i\}$, and it can go to the first vertex on the detection path from the last in $t_2 \leq \sum_{i,j} w_{ij}$. So, once it reaches the starting vertex of the detection path, it can wait for $T - t_1 - t_2 > 0$ time and then follow the same path to visit all the vertices in their confirmation windows.

Therefore, the decision version of TSPTW is true if and only if off-line EDC is true. Moreover, given any certificate of off-line EDC, it can be easily checked in polynomial time whether it visits all the vertices in their corresponding detection and confirmation windows. Hence, off-line EDC is NP-Complete. □

The above result shows that it is computationally intractable to even determine a feasible patrolling path given all the problem data beforehand. This implies that the optimization version of the off-line problem is NP-hard. Thus, we do not expect there to exist a tractable algorithm for optimally solving the online problem, in which events arrive over time and the robot does not know their arrival times or activity periods.

In the following section we look at the probability of confirmation at a single vertex in the graph when one robot is patrolling the graph. Using this we can analyze the performance of a policy based on a traveling salesman problem (TSP) tour.

4.3 Analysis of a Periodic Policy for a Single Vertex

Let us consider a deterministic patrolling policy that visits the vertices of the graph periodically and let the period for a vertex v be τ_v. The arrival of events is governed by the Poisson process, so the inter-arrival times are exponentially distributed with parameter λ_v for vertex v. The staying time of the events is also exponentially distributed with parameter μ_v. Since the arrival and departure process at a vertex are independent of the states of the other vertices, we can focus on the analysis of a single vertex. For the simplicity of notation, we will drop the subscript v and use the parameters λ and μ for the arrival and departure rates for the vertex under study and τ as the period for that vertex.

4.3.1 Confirmation Avoiding Interval for True Events

The goal of the problem is to find a policy that maximizes the probability of correctly classifying true events. There is a chance that a true event becomes inactive without being confirmed, because the robot does not know the exact time of the arrival of the event and

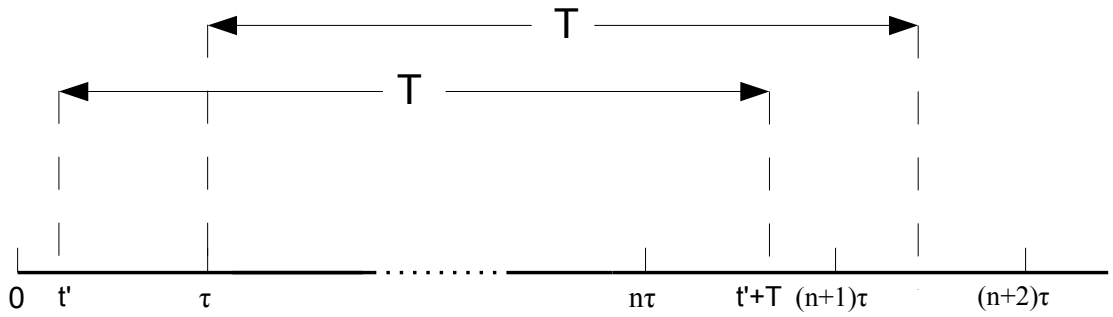

Figure 4.2: The events remaining active until $t'+T$ are true, but they cannot be confirmed unless they stay until $(n+2)\tau$.

it may not be able to visit the vertex exactly T time after the detection. We characterize the time interval during which a true event can become inactive and avoid confirmation.

Proposition 4.3.1 (False Negatives)**.** *Suppose an event arrives at a vertex between two consecutive visits made by the robot at* 0 *and* τ*, and the arrival time of the event is given by* $t' \in (0, \tau]$*. Then, if that event becomes inactive in the time interval*

$$\big(t'+T, (n+2)\tau\big), \text{ where} \tag{4.1}$$

$$n = \begin{cases} \frac{T}{\tau} - 1, & \text{if } T \text{ is a multiple of } \tau \\ \lfloor \frac{T}{\tau} \rfloor, & \text{otherwise,} \end{cases} \tag{4.2}$$

it will be a true event that can not be confirmed.

Proof. The starting point of the interval $(t'+T, (n+2)\tau)$ is trivial since the event will become true after $t = t'+T$. For the end point, notice that the robot detected the event at $t = \tau$, and it will only be able to confirm the event on times that are integer multiples of τ. By the definition of n, $n\tau < T \leq (n+1)\tau$. So when the robot observes the same event at $(n+2)\tau$, it confirms that is has been there for more than T, since $(n+2)\tau - \tau = (n+1)\tau \geq T$. Moreover, there can be cases for some T and t' when $t'+T < (n+1)\tau$, as shown in Figure 4.3.1, but since the robot detected the event at τ and $(n+1)\tau - \tau = n\tau < T$, the robot does not know that the event has been active for more than T and cannot confirm it. So it will only be able to confirm that event at $(n+2)\tau$. □

The events which become inactive in the interval (4.1) will be true events, but cannot be correctly classified by the robot as true. We will use this fact along with the exponential active times of the events in the following section to calculate the chances of correctly classifying a true event.

4.3.2 Probability of Correctly Classifying True Events

The events which were detected at $t = \tau$ will be classified as true if they remain active until $t = (n+2)\tau$, as shown in Proposition 4.3.1. If the robot detects an event, then it knows that the event arrived in the interval between the last two visits to its vertex. By the property of stationary increments, the time scale can be shifted to say that the arrival time of the event is given by $t' \in (0, \tau]$. Using the consequence of Lemma 2.1.15, the arrival time t' is uniformly distributed over $(0, \tau]$. We write this distribution of t' in $(0, \tau]$ as

$$f(t') = \frac{1}{\tau} \quad \text{for } t' \in (0, \tau]. \tag{4.3}$$

We will use this uniform density along with the interval (4.1) to find the probability of confirming true events.

Proposition 4.3.2 (Probability of Successful Classification)**.** *The probability of confirming a true event at vertex v with arrival rate λ, departure rate μ, for a robot with visit period τ is*

$$\mathbb{P}\left[\texttt{confirm|v}\right] = \frac{e^{-\mu[(n+2)\tau - T]}(e^{\mu\tau} - 1)}{\mu\tau}, \tag{4.4}$$

where n is defined in (4.2).

Proof. According to the confirmation avoiding interval given in (4.1), the events arriving at time $t' \in (0, \tau]$ and departing after $(n+2)\tau$ will be confirmed by the robot. Using the exponential staying time distribution ,we can find the probability of confirming a true event given that it arrived at time $t' \in (0, \tau]$.

$$\begin{aligned} \mathbb{P}\left[\texttt{confirm|v and t'}\right] &= \frac{\displaystyle\int_{(n+2)\tau}^{\infty} \mu e^{-\mu(t-t')} dt}{\displaystyle\int_{T}^{\infty} \mu e^{-\mu t} dt}, \\ &= e^{-\mu[(n+2)\tau - T]} e^{\mu t'} \end{aligned} \tag{4.5}$$

The numerator in (4.5) represents the events that will stay long enough to be confirmed, and the denominator represents all the events that are true. Using the arrival time density in the interval $(0, \tau]$ from (4.3), $f(t') = \frac{1}{\tau}$ where $t' \in (0, \tau]$, and un-conditioning the arrival time

$$\mathbb{P}\left[\texttt{confirm|v}\right] = \int_0^{\tau} \mathbb{P}\left[\texttt{confirm|v and t'}\right] f(t')dt',$$

we get

$$\begin{aligned}\mathbb{P}\left[\texttt{confirm|v}\right] &= \int_0^{\tau} \frac{e^{-\mu[(n+2)\tau - T]} e^{\mu t'}}{\tau} dt' \\ &= \frac{e^{-\mu[(n+2)\tau - T]}(e^{\mu\tau} - 1)}{\mu\tau}.\end{aligned}$$

□

The Event Detection and Confirmation problem seeks to maximize the number of true events that are confirmed. So, one would want to maximize the probability of confirmed true events given in (4.4). However, this expression is just for a single vertex on the path of the robot. We will extend it to the complete path, and then try to maximize the probability of confirming true events over the whole graph.

4.4 Single Robot Policy Based on TSP

The expression for the probability of successfully confirming a true event for one vertex can be used to find the probability of confirming true events over the whole graph. In this section, we will derive the expression for the said probability and then use it in a particular case to recommend a policy based on the cyclic tour of the graph.

4.4.1 Probability of Correct Classification for the Tour

We start with the analysis of any patrolling policy with possibly different periodic visit times to vertices, and then specialize the equation for the case when the periodic visit times to the vertices are equal and the events' activity period is governed by the same process for all the vertices.

Using equation (4.4), for a vertex v with arrival and departure rates given by λ_v and μ_v respectively, the robot visiting that vertex with a period τ_v will confirm true events on that vertex with a probability given by

$$\mathbb{P}\left[\texttt{confirm}\,|\,\texttt{v}\right] = \frac{e^{-\mu[(n_v+2)\tau_v - T]}(e^{\mu_v\tau_v}-1)}{\mu_v\tau_v},$$
$$\text{where } n_v = \begin{cases} \frac{T}{\tau_v} - 1, & \text{if } T \text{ is a multiple of } \tau_v \\ \left\lfloor \frac{T}{\tau_v} \right\rfloor, & \text{otherwise,} \end{cases} \tag{4.6}$$

Proposition 4.4.1 (Probability Expression)**.** *The probability of correctly classifying true events for the periodic tour is given by*

$$\mathbb{P}\left[\text{confirm}\right] = \frac{\sum_v \mathbb{P}\left[\texttt{confirm}\,|\,\texttt{v}\right]\lambda_v}{\sum_v \lambda_v}, \tag{4.7}$$

where $\mathbb{P}\left[\texttt{confirm}\,|\,\texttt{v}\right]$ *is given in equation (4.6). Moreover, in the special case where* $\tau_v = \tau$*, and* $\mu_v = \mu$*, for all* $v \in V$*, then*

$$\mathbb{P}\left[\text{confirm}\right] = \frac{e^{-\mu[(n+2)\tau - T]}(e^{\mu\tau}-1)}{\mu\tau},$$
$$\textit{where } n = \begin{cases} \frac{T}{\tau} - 1, & \textit{if } T \textit{ is a multiple of } \tau \\ \left\lfloor \frac{T}{\tau} \right\rfloor, & \textit{otherwise.} \end{cases} \tag{4.8}$$

Proof. We want to remove the condition of arrival being on a certain vertex v from equation (4.6). We know that

$$\mathbb{P}\left[\texttt{confirm}\right] = \sum_v \mathbb{P}\left[\texttt{confirm}\,|\,\texttt{v}\right]\mathbb{P}\left[\texttt{arrival at v}\right].$$

Since the arrivals of events at different vertices are independent processes, the probability of arrival of an event being on vertex v is

$$\mathbb{P}\left[\texttt{arrival at v}\right] = \frac{\lambda_v}{\sum_v \lambda_v}.$$

Therefore,

$$\mathbb{P}\left[\texttt{confirm}\right] = \frac{\sum_v \mathbb{P}\left[\texttt{confirm}\,|\,\texttt{v}\right]\lambda_v}{\sum_v \lambda_v}.$$

If we consider the special case when the staying times on all the vertices are identically distributed and the robot follows a cyclic policy, i.e. it visits all the vertices in some

order with the same period for each vertex, then $\mu_v = \mu, \tau_v = \tau, \forall v \in V$. In this case, $\mathbb{P}[\texttt{confirm|v}]$ is same for all the vertices, and can be factored out of the summation, giving us equation (4.8).

□

Notice that the probability expression (4.8) is independent of λ. The reason for this can be understood by noting that in equation (4.7), the number of confirmed events depends on λ, but so does the number of total events. As a result, the probability remains unaffected.

4.4.2 Policy based on a TSP tour

Expression (4.8) holds for the case when the period of visits for the robot is same for all vertices and the activity times of events at different vertices are identically distributed. A TSP tour (which is the shortest tour to visit all vertices in the graph) minimizes the time τ for a given speed of the robot.

However, there are cases when decreasing the robot speed and thus increasing τ results in a higher probability in expression (4.8). This is due to the discontinuity of n in equation (4.2) and can be seen in Figure 4.3. Intuitively it means that timing the visits such that T is a multiple of τ tends to decrease the chances of missing the confirmation of true events. That is because the robot will reach the vertex of the event at the earliest possible time the robot can confirm that event.

Thus, we need to check two possible robot speeds: maximum speed, or a speed such that τ is a multiple of T. Based on this observation, we arrive at following single robot policy.

Policy for a Single Robot:

1. Calculate the TSP tour of the graph, and find the minimum time τ_{min} to complete that tour by the robot at its maximum speed.
2. Find $\tau_{\text{peak}} \geq \tau_{\text{min}}$ such that T is a multiple of τ_{peak} and there is no other τ between τ_{min} and τ_{peak} which is a multiple of T.
3. Calculate the probability of correctly classifying true events from equation (4.8) for $\tau = \tau_{\text{min}}$ and $\tau = \tau_{\text{peak}}$.

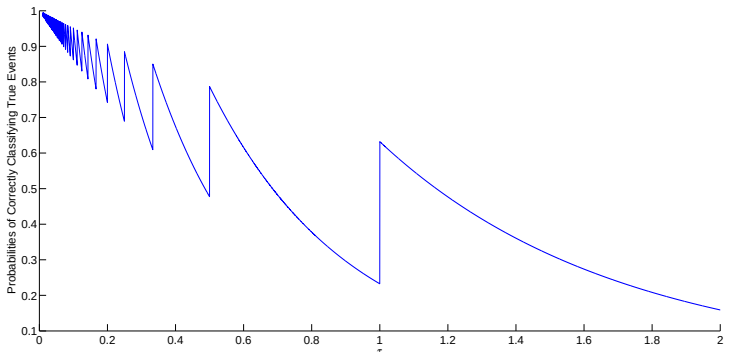

Figure 4.3: The probability of correctly classifying true events versus τ, with $T = 1$, $\mu = 1$.

4. Choose the tour period which gives greater probability and adjust the speed of the robot to the optimal speed to match the chosen period.

Remark 4.4.2 (Omitting Vertices from Tour)**.** *An instance of the problem with equal arrival rates can be constructed where missing a far away vertex in the graph will result in a tour which has a much lower value of τ for the other vertices. Then the probability of confirming true events on the missed vertex becomes* 0 *since the robot never visits that vertex and hence, the probability of correctly classifying true events on the whole graph can be written as*

$$\begin{aligned}\mathbb{P}\left[\text{confirm}\right] =& \frac{\sum_{v=0}^{N-1} \mathbb{P}\left[\texttt{confirm}|\texttt{v}\right] \lambda + 0}{\sum_{v=0}^{N} \lambda} \\ =& \frac{N-1}{N} \frac{e^{-\mu[(n+2)\tau - T]}(e^{\mu\tau} - 1)}{\mu\tau}.\end{aligned} \tag{4.9}$$

If the period τ is reduced enough for the other vertices by omitting some vertex, the probability of confirming true events can be increased. However, such policies raise the possibility that "intelligent" events would begin to choose this unvisited vertex more frequently, altering the arrival rates.

4.5 Multiple Robots

In this section we discuss the case of multiple robots. We give some general results and then give the optimal lag between the robots for the two robot scenario. We then generalize the result for the optimal lag for multiple robot case.

We assume that the communication graph between the robots is strongly connected, so that any two robots can communicate without significant delay. Thus, we can assume that the database containing all active events is shared among the robots. This means that it is possible to have robot i detect an event and robot j confirm it and vice versa.

4.5.1 Specializing Robot Capabilities

One possible solution in the multi-robot case is to utilize *specialization* in which a robot performs exclusively detection, or exclusively confirmation.

Definition 4.5.1 (Specialized robot capability)**.** *We say that a robot is a detection (confirmation) robot if it is capable of performing only event detections (confirmations).*

First, it is easy to see that specialization cannot be optimal. In specializing we eliminate the possibility of a confirmation robot detecting an event, even if it is the first robot to visit the vertex after the events' arrival. Similarly, we eliminate the possibility of a detection robot confirming an event, even when it visits the event vertex more than T time units after detection.

However, there are cases where specialization may be required, for example, if the sensors needed for detection and confirmation differ. For example, in the case of patrolling a parking lot to ticket overstaying vehicles, the confirmation robot might require additional capabilities to be able to ticket the vehicle, whereas the detection robot just has to observe the vehicles. The following simple lemma shows that when specializing, detection is the bottleneck.

Lemma 4.5.2 (Specialization among robots)**.** *Given n_d detection robots, confirmation can be performed optimally using only n_d confirmation robots.*

Proof. Let the paths followed by the n_d detection robots be $P_1, \ldots, P_{n_d}$. We then create n_d confirmation paths by placing a confirmation robot on each detection path, but with a time lag of exactly T seconds.

An event that is detected on a given path P_i will be confirmed exactly T time units later by the corresponding confirmation robot. In other words, given the detection times, each confirmation will be performed optimally using the n_d confirmation paths, with a time lag of T. □

The consequence of this result is that detection is the bottleneck when looking at specialized robots. Given detection patrolling paths, the corresponding optimal confirmation paths are defined. Thus, in this case, one can use existing techniques to design patrolling paths for detection, and then use Lemma 4.5.2 for the confirmation paths. In the next section we focus on the more complex case in which each robot can both detect and confirm.

4.5.2 Optimal Spacing Between Two Robots on a Common Path

In this section we look at the case where each robot can both detect and confirm events. We focus on the special case in which there are two robots moving along the same tour with a period of $\tau > 0$. Moreover, we assume that the staying times at different vertices are identically distributed. We look to optimize the spacing between the robots along the tour. We discuss the extension to m robots at the end of this section.

To this end, define the variable to optimize as t_{lag} which is the time lag between the first and second robot on the common tour. Since the robots travel the tour with period τ, we have $t_{\text{lag}} \in (0, \tau)$. Consider an event that arrives at a vertex at time $t' \geq 0$. We can shift the time scale such that $t' \in [0, \tau)$.

Consider the times at which that event can be detected and confirmed: we call these times t_{det} and t_{conf}, respectively, where $t_{\text{conf}} \geq t_{\text{det}} + T$ and $t_{\text{det}} \geq t'$. If these times are known, then the probability of confirming a true event given the arrival time of the event t' is

$$\begin{aligned}
\mathbb{P}\left[\texttt{confirm} | t'\right] &= \frac{\mathbb{P}\left[\texttt{active} > t_{\text{conf}} - t'\right]}{\mathbb{P}\left[\texttt{active} > T\right]} \\
&= \frac{e^{-\mu(t_{\text{conf}} - t')}}{e^{-\mu T}} \\
&= e^{\mu t'} e^{-\mu(t_{\text{conf}} - T)}, \qquad (4.10)
\end{aligned}$$

where we have used the fact that an event's active time is distributed according to an exponential random variable with parameter μ. Also, recall that n is defined in equation (4.2).

Notice that the robots visit the vertex of interest at the times 0, t_{lag}, τ and so on. We call the robot that visits the vertex at t_{lag} robot 1, whereas the robot that visits at τ is called robot 2. Now, we calculate t_{det} and t_{conf} as a function of t_{lag} using the following two

cases, each containing two sub-cases.

Case 1: Robot 1 detects the event
If $t' \in (0, t_{\text{lag}}]$ then the event will be detected by robot 1 and $t_{\text{det}} = t_{\text{lag}}$. There are two further sub-cases.

1. **Robot 2 confirms the event:** If $t_{\text{lag}} + T \leq (n+1)\tau$ then the earliest time that the event can be confirmed is $t_{\text{conf}} = (n+1)\tau$ and robot 2 will confirm the event.

2. **Robot 1 confirms the event:** If $t_{\text{lag}} + T > (n+1)\tau$, then robot 1 will confirm the event and the confirmation time is $t_{\text{conf}} = (n+1)\tau + t_{\text{lag}}$.

Case 2: Robot 2 detects the event
If $t' \in (t_{\text{lag}}, \tau]$ then the event will be detected at $t_{\text{det}} = \tau$ by robot 2. Again, there are two sub-cases:

1. **Robot 1 confirms the event:** If $\tau + T \leq (n+1)\tau + t_{\text{lag}}$ i.e., $t_{\text{lag}} \geq T - n\tau$, then the earliest time that the event can be confirmed is $t_{\text{conf}} = (n+1)\tau + t_{\text{lag}}$ and it will be confirmed by robot 1.

2. **Robot 2 confirms the event:** If $t_{\text{lag}} < T - n\tau$, then the event will be confirmed by robot 2 at $t_{\text{conf}} = (n+2)\tau$.

Based on the four cases and equation (4.10), we can compute the probability of detection as a function of t_{lag} as

$$\mathbb{P}\left[\texttt{confirm}\right] = \int_0^\tau \mathbb{P}\left[\texttt{confirm}|t'\right] f(t')dt',$$

where $f(t')$ is the uniform distribution from equation (4.3).

When evaluating this integral, there are two more cases, depending on whether or not $T - n\tau \leq (n+1)\tau - T$:

- If $T - n\tau \leq (n+1)\tau - T$ we get

$$\mathbb{P}\left[\texttt{confirm}\right] = \begin{cases} \frac{1}{\mu\tau}(e^{-\mu((n+1)\tau - T)}(e^{\mu t_{\text{lag}}}(1 - e^{-\mu\tau})), & t_{\text{lag}} \leq T - n\tau, \\ \frac{1}{\mu\tau}(e^{-\mu((n+1)\tau - T)}(e^{\mu t_{\text{lag}}} + e^{\mu(\tau - t_{\text{lag}})} - 2), & T - n\tau < t_{\text{lag}} \leq (n+1)\tau - T, \\ \frac{1}{\mu\tau}(e^{-\mu((n+1)\tau + t_{\text{lag}} - T)}(e^{\mu\tau} - 1), & t_{\text{lag}} > (n+1)\tau - T. \end{cases} \tag{4.11}$$

- If $T - n\tau > (n+1)\tau - T$, we get

$$\mathbb{P}\left[\texttt{confirm}\right] = \begin{cases} \frac{1}{\mu\tau}(e^{-\mu((n+1)\tau - T)}(e^{\mu t_{\text{lag}}}(1 - e^{-\mu\tau})), & t_{\text{lag}} \leq (n+1)\tau - T, \\ \frac{1}{\mu\tau}(e^{-\mu((n+1)\tau - T)}(2 - e^{-\mu t_{\text{lag}}} - e^{-\mu(\tau - t_{\text{lag}})}), & (n+1)\tau - T < t_{\text{lag}} \leq T - n\tau, \\ \frac{1}{\mu\tau}(e^{-\mu((n+1)\tau + t_{\text{lag}} - T)}(e^{\mu\tau} - 1), & t_{\text{lag}} > T - n\tau. \end{cases} \tag{4.12}$$

Notice that in the case when $t_{\text{lag}} = \tau/2$, the expressions in (4.11) and (4.12) both simplify to equation (4.8) with τ replaced by $\tau/2$. We can now use these expressions to find the optimal t_{lag}.

Proposition 4.5.3 (Optimal value of t_{lag})**.** *The equations* (4.11) *and (4.12) achieve their global maxima at one (or more) of the following points: i)* $t_{\text{lag}} = \frac{\tau}{2}$*; ii)* $t_{\text{lag}} = T - n\tau$*; iii)* $t_{\text{lag}} = (n+2)\tau - T$*.*

Proof. The equations (4.11) and (4.12) are piecewise continuous and have discontinuities at points $t_{\text{lag}} = T - n\tau$ and $t_{\text{lag}} = (n+2)\tau - T$. The continuous pieces defined on the first and third interval of the equations are strictly monotone and achieve their maximum values at the discontinuities. The third continuous part has an extremum at $t_{\text{lag}} = \frac{\tau}{2}$ which can be easily verified by equating the derivative of the function to zero. Thus the maximum of the probability lies either at $\frac{\tau}{2}$ or at one of the discontinuities. So, for one of these values of t_{lag}, the probability will be maximized. □

These values of t_{lag} will optimize the probability for a given value of τ. However, for the case when the robots are equally spaced, equation (4.8) gives the probability of confirming events with τ replaced by $\tau/2$. We have observed in the single robot case that increasing τ can result in a higher probability. Similarly, in the two robot case, when the robots are equally spaced, decreasing the speed of the robot to increase τ to a divisor of $2T$ can result in a higher probability. Based on this, we arrive at the following policy for two robots:

Policy for Optimizing the Spacing of Two Robots:

1. Evaluate the expression (4.11) or (4.12) depending on whether $T - n\tau \leq (n+1)\tau - T$ or not, at the points $t_{\text{lag}} = \tau/2$, $t_{\text{lag}} = T - n\tau$ and $t_{\text{lag}} = (n+2)\tau - T$. Choose the value of t_{lag} that gives the maximum probability. Call it t_{lag_1}.

2. Increase τ to the nearest divisor of $2T$, call it τ_n and evaluate equation (4.12) at $t_{\text{lag}} = \tau_n/2$ and $\tau = \tau_n$.

3. If $t_{\text{lag}} = \tau_n/2$ gives the higher probability, choose $\tau = \tau_n$ and $t_{\text{lag}} = \tau_n/2$. Otherwise, choose $t_{\text{lag}} = t_{\text{lag}_1}$.

The following remark discusses the extension to m robots.

Remark 4.5.4 (Generalizing to m robots)**.** *In the case that there are m robots, there are $m-1$ variables to optimize, each for the lag between robots on a cyclic path. The number of cases to consider becomes too large to complete the same analysis. However, based on the observations made for two robots, the following can be said for the n-robot case:*

1. *If τ/m is a multiple of T, then equally space the robots on the path.*
2. *Otherwise, if $\tau < mT$, decrease τ to the nearest divisor of mT and using this new period, equally space the robots.*
3. *If $\tau > mT$, choose the spacing such that the robots follow each other by a time lag of T.*

This policy follows from the observation that in the two robot case this procedure often yields the optimal t_{lag}. However, it is not, in general guaranteed to find the optimal spacing.

4.6 Application Example

Let us consider the parking lot shown in Figure 4.1 and apply the patrolling policies to that parking lot. The parking lot is situated at a market place and for the purposes of simulation, we assume the expected staying time of vehicles to be around 75 minutes and the allowed parking time to be two hours. This gives $\mu = \frac{1}{75}$ and $T = 120$. Moreover, the length of the patrolling path shown in Figure 4.1 is calculated to be approximately 870 meters.

Figure 4.4 shows the plot of probabilities of ticketing overstaying vehicles versus patrolling period of the robot in the cases of i) a single robot; ii) two robots with a spacing of $\tau/2$; and iii) two robots with optimized spacing.

We assume the robot has a maximum speed of 1 m/s, which gives a minimum tour period of 14.5 minutes. The probability of ticketing for a tour with this period comes out to be 0.7905 from equation (4.8). However, if we increase the period to 15 minutes by decreasing the speed of robot to 0.967 m/s, using equation (4.8) the probability increases to 0.9063 — an increase of 14.6%.

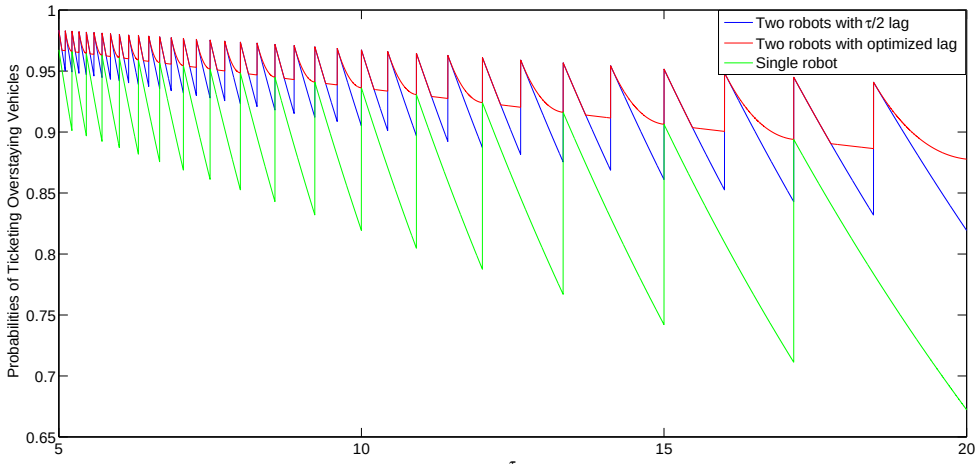

Figure 4.4: The probability of ticketing overstaying vehicles as a function of τ for 1) a single robot, 2) two robots with lag of $\tau/2$, and 3) two robots with optimized lag. Here $T = 120$, and $\mu = 1/75$.

In case of two robots, the probability of ticketing an overstaying vehicle using a period of 14.5 minutes and the robots equally spaced will be 0.9128 from equation (4.12) . But, if one robot follows the other with a time lag of $T - n\tau = 4$ minutes or $(n+1)\tau - T = 10.5$ minutes (since the robots are on a cycle, these configurations are equivalent), then the probability increases to 0.9221 from equation (4.11). However, there is still room for improvement. If both the robots decrease their speed to make their period a multiple of T and then follow each other with a lag of $\tau/2 = 7.5$ minutes, the probability will be 0.9515 which can be calculated using equation (4.12) or equation (4.8) with τ replaced by $\tau/2$.

This example shows that decreasing the speed can be helpful in terms of increasing the probability of confirming a true event. However, it may not always be true. Looking at Figure 4.4 that most of the times a lag of $\tau/2$ would result in better probability, and even if it does not, the optimal lag seems to provide very little advantage. However, Figure 4.5 shows that in the cases when $\tau > 2T$ in a two robot scenario, optimal lag provides much better results. Such cases can arise when robots have to monitor very large environments.

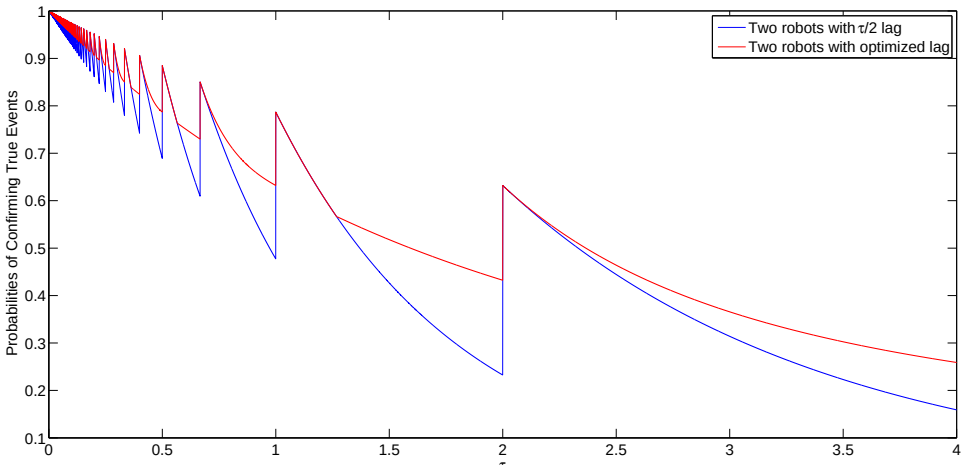

Figure 4.5: Comparison of probabilities of confirming events for two robots with different lags versus τ. Parameter values are $T = 1$, $\mu = 1$.

Chapter 5

Conclusions and Future Work

We presented and analyzed two patrolling problems in this thesis. The first problem discussed stochastic patrolling strategies in adversarial environments. Markov chains were used to represent stochastic patrolling paths on the graphs. We modeled four different types of intruders that can attack the environment, each with a different level of intelligence. The expected reward for each of the intruders was characterized as a function of the transition matrix of the Markov chain being used for patrolling. We discussed the application of different numerical optimization techniques to optimize the expected reward for any given type of intruder, and propose the pattern search method to determine a stochastic patrolling policy. We present the performance of the pattern search method on some example problem instances. We also show that for a certain type of intruder given in Section 3.2.4, a deterministic patrolling path given by the orienteering tour of the graph is the optimal patrolling strategy.

In the second part of the thesis we consider the Event Detection and Confirmation Problem. The events arrive randomly on the vertices of a graph and stay active for a random amount of time. The events that stay active for longer than a certain amount of time are called true events. The monitoring robot has to classify the events as true or false. We showed that the off-line version of the problem is NP-complete. We considered a simple patrolling policy based on the TSP tour of the graph and characterized the probability of correctly classifying true events. We gave some insight into the multiple robot problem and showed that if all robots follow the same path, the optimal spacing between the robots can be non uniform.

5.1 Future Directions for Patrolling in Adversarial Settings

In this section we discuss two possible future directions for the patrolling problem in adversarial settings.

5.1.1 Real Time Learning of Markov Chains

The intelligent intruder models given in Sections 3.2.4 and 3.2.5 assume that the intruder has access to the transition matrix of the Markov chain that the robot is following. However, in a real scenario, the intruder might have to learn the Markov chain by observing the patrolling path of the robot. Then the success matrix S calculated by the intruder will depend on the learned transition matrix $\hat{P}$ instead of the actual transition matrix P. So, the vertex and time to attack selected by the intruder based on the learned model might not be optimal, since the robot is following a different chain from the one modeled by the intruder. Therefore, one of the possible future directions is to look for patrolling policies that are hard to learn. In the following section we present a learning technique from [30, 6], and then we discuss the future direction in a detailed manner.

Bayesian Estimation for Markov Chains

Bayesian Estimation is used to estimate the parameters of a probability density given the samples drawn from that distribution. If we denote the observed samples by $\mathcal{D}$ and the actual parameters of the distribution by θ, then using Bayes' law, we can calculate the posterior estimate of the parameters.

$$\mathbb{P}\left[\theta|\mathcal{D}\right] = \frac{\mathbb{P}\left[\mathcal{D}|\theta\right]\mathbb{P}\left[\theta\right]}{\mathbb{P}\left[\mathcal{D}\right]}$$

This can be interpreted as

$$\texttt{posterior} = \frac{\texttt{likelihood of data given parameters} \times \texttt{prior}}{\texttt{marginal likelihood}}$$

For Markov chains, each row i of the transition matrix can be thought of as a multinomial distribution with the number of parameters equal to the number of neighbors of vertex i. Given that the chain is in state i, the next state is drawn from the multinomial

distribution. So estimating the transition matrix is equivalent to estimating n independent multinomial distributions.

The prior used to estimate multinomial distributions is given by the Dirichlet distribution [20]. So, for the multinomial parameters $\theta_{i1}, \theta_{i2}, \ldots, \theta_{iq_i}$ that correspond to the transition probabilities from a state i to its q_i neighbors, the prior is

$$\mathbb{P}\left[\theta_{i1}, \ldots, \theta_{iq_i}\right] = \texttt{Dir}(\theta_{i1}, \ldots, \theta_{iq_i} | \alpha_{i1}, \ldots, \alpha_{iq_i})$$

where $\alpha_{i1}, \ldots, \alpha_{iq_i}$ are the parameters of the Dirichlet distribution. The parameter α_{ij} for the multinomial distribution i of the Markov chain can be intuitively thought of as the number of transitions from i to j in prior observations.

If we use the above prior, the maximum a posteriori(MAP) estimate comes out to be

$$\hat{p}_{ij} = \frac{y_{ij} + \alpha_{ij}}{\sum_{k=1}^{n}(y_{ik} + \alpha_{ik})} \tag{5.1}$$

where y_{ij} is the number of observed transitions from state i to state j. The Laplace prior is $\alpha_{ij} = 1$, for $i \in \{1, \ldots, n\}$, $j \in \{1, 2, \ldots, q_i\}$ and is used when no information about the chain is available beforehand.

The MAP estimate given in equation (5.1) gives a point estimate of the transition probabilities. The posterior estimate calculated using Bayes' law gives a distribution over the parameter values. If the Dirichlet distribution is used as the prior for estimating a multinomial distribution, the posterior is also a Dirichlet distribution. So,

$$\mathbb{P}\left[\theta_{i1}, \ldots, \theta_{iq_i} | \mathcal{D}\right] = \texttt{Dir}(\theta_{i1}, \ldots, \theta_{iq_i} | \alpha_{i1} + y_{i1}, \ldots, \alpha_{iq_i} + y_{iq_i})$$

The MAP estimate is the mean of this posterior Dirichlet distribution. This distribution can be used by the intruder to compute the confidence in its estimate of the parameters of the transition matrix. We can write the variance in the parameter θ_{ij} in terms of the MAP estimate $\hat{p}_{ij}$ as

$$\texttt{Var}(\theta_{ij}) = \frac{\hat{p}_{ij}(1 - \hat{p}_{ij})}{y_i + q_i + 1} \tag{5.2}$$

where $y_i = \sum_j y_{ij}$ is the number of times state i is visited and q_i is the number of neighbors of state i. So, as the length of the observed path increases, y_i increases, and the variance in the estimate decreases. In the Sections 3.2.4 and 3.2.5, we assumed that the intruder has been observing the patrolling path long enough to have perfectly modeled the Markov chain.

If the intruder has been observing the patrolling path for some time T_o, then it will use the estimated transition matrix denoted by $\hat{P}_{T_o}$ to decide the vertex and time of attack. If we are looking for patrolling policies that are hard to learn, we would want to design policies such that for a given T_o, the variance (5.2) is maximized. However, this might not reduce the expected reward of the intruder as illustrated by the following example.

Example 5.1.1. *Consider a complete unweighted graph of n vertices and two different Markov chains defined on that graph. Chain 1 is completely random and has transition probabilities $p_{ij} = 1/n$ for all i, j. Chain 2 is deterministic and will have transition probabilities as either 1 or 0. It can be seen from the equation (5.2) that an intruder learning chain 2 will have a much higher confidence in its estimate after time T_o as compared to an intruder learning chain 1. However, that does not imply that the intruder attacking chain 1 will make wrong choices of the time and vertex of attack. In fact, if $T_o = 0$, the intruder will model the chain 1 perfectly using MAP estimate and hence can make the best decisions. On the other hand, the estimate of chain 2 will be far off from the actual chain and the intruder might not make the best possible choice for the time of attack.*

The example shows that designing a chain that is hard to learn does not necessarily imply that the performance of the intruder will initially suffer. The final goal of the patrolling problem is to minimize the intruder's reward that it can obtain by intelligently attacking the environment. Given a time T_o, the intruder will launch attacks using the learned model $\hat{P}_{T_o}$, and the expected reward will be given by $f(\hat{P}_{T_o})$. We would like to design a policy that minimizes this reward. The problem discussed in this thesis is a specific case of this proposed problem with T_o approaching infinity.

5.1.2 Multiple Robots

Another possible direction for future work is the design of patrolling policies for multiple robots in adversarial environments. Given a graph and R agents to patrol it, one has to decide how to assign the graph to the agents and then to define a patrolling policy for each agent on its specified graph. The following two approaches are discussed in [37].

Shared Graph Policy: The graph is shared by all the agents i.e. all the agents patrol the whole graph. The agents can either follow their independent patrolling paths, or they can cooperate and define a combined strategy to patrol the graph.

Partitioned Graph Policy: The graph is partitioned into R partitions and each partition is assigned to one agent. In [37] the authors discuss these strategies and present some graph partitioning algorithms such as K-means clustering algorithm and Multilevel graph partitioning. By partitioning, we get R single agent problems and we can define the patrolling policy for each agent on its graph.

For a given length of attack l, as the graph size decreases, the maximum probability of successful attacks intuitively decreases. Similarly, if more robots are patrolling a graph, the chances of a successful attack should reduce. However, the following example suggests that as the number of robots increases, partitioning will become better than sharing in terms of minimizing the maximum probability of successful attacks.

Example 5.1.2. *Consider the case when number of robots is such that length of attack l is greater than the tour length of the partitions but less than the tour length of the full graph. Then the probability of successful attacks in case of partitions is zero, whereas if the maximum probability of successful attacks on a graph is non zero with a single agent, it will remain non zero with multiple agents independently patrolling the graph.*

However, wait for the perfect time of attack for an intruder can be very large in case of shared strategy as compared to the partitioned one. To launch a best possible attack on the vertex j of an environment with multiple patrolling agents, the intruder has to

1. Compute the matrices S_k for $k \in \{1, \ldots, R\}$. the matrix S_k corresponds to the success matrix calculated using the transition matrix for robot k.
2. Find the row numbers $i(k), k \in \{1 \ldots R\}$ corresponding to the maximum elements in the j^{th} columns of the matrices.
3. Wait till the agent k is at $i(k)$, $\forall k \in \{1 \ldots R\}$.

Waiting time: The amount of time an intruder has to wait after deciding to attack a vertex until it can launch the best possible attack is called the waiting time. In case of partitioned graphs, the total number of vertices in the graph reduce and hence the waiting time for the intruder decreases. In the case of shared graphs, the intruder has to wait for all of the R agents to reach some specific vertices before it can launch the best possible attack. The shared graph by R agents will have $|V|^R$ possible states of the agents, and the intruder will have to wait till the stochastic system reaches one of those $|V|^R$ states. Hence the waiting time of the intruder increases exponentially with the number of robots

in case of shared graph strategy.

Hence, there is a trade off between the maximum probability of successful attacks and the wait time of the intruder to launch the best attack as we go from partitioned graphs to shared graphs. Th study of this trade off and designing optimal policies for multiple robots is an interesting topic for future work.

5.2 Future Directions for Event Detection and Confirmation Problem

We discussed the Event Detection and Confirmation Problem for two robots in Section 4.5 and we gave a generalization for multiple robots based on the observations made in the two robots case. However, theoretically analyzing the multiple robot scenario for this problem is an open problem. For k robots following each other on the same path, the number of lag variables is $k-1$. The expression for the probability of confirming true events as a function of lag variables becomes challenging to evaluate as the number of robots increases.

Another possible direction for future work is to solve Event Detection and Confirmation Problem in adversarial settings. The events can actively try to evade being confirmed by choosing the vertices intelligently. Randomizing the patrolling path can be effective in this case as well and similar techniques to the ones we used in Chapter 3 might be employed to study the problem.

References

[1] J. J. Acevedo, B. C. Arrue, I. Maza, and A. Ollero. Cooperative perimeter surveillance with a team of mobile robots under communication constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5067–5072, Chicago, USA, 2013.

[2] P. Agharkar, R. Patel, and F. Bullo. Robotic surveillance and Markov chains with minimal first passage time. In *IEEE Conference on Decision and Control*, pages 6603–6608, Los Angeles, CA, USA, December 2014.

[3] N. Agmon, Sarit Kraus, and G.A Kaminka. Multi-robot perimeter patrol in adversarial settings. In *IEEE International Conference on Robotics and Automation*, pages 2339–2345, Pasadena, CA, USA, May 2008.

[4] T. Alam, M. Edwards, L. Bobadilla, and D. Shell. Distributed multi-robot area patrolling in adversarial environments. In *International Workshop on Robotic Sensor Networks*, Seattle, WA, USA, 2015.

[5] S. Alamdari, E. Fata, and S. L. Smith. Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations. *The International Journal of Robotics Research*, 33(1):138–154, 2014.

[6] T. W. Anderson and L. A. Goodman. Statistical inference about Markov chains. *The Annals of Mathematical Statistics*, 28(1):89–110, 03 1957.

[7] A.B. Asghar and S.L. Smith. Robot monitoring for the detection and confirmation of stochastic events. In *IEEE Conference on Decision and Control*, pages 408–413, Los Angeles, CA, USA, Dec 2014.

[8] M. Baseggio, A. Cenedese, P. Merlo, M. Pozzi, and L. Schenato. Distributed perimeter patrolling and tracking for camera networks. In *IEEE Conference on Decision and Control*, pages 2093–2098, Atlanta, GA, USA, 2010. IEEE.

[9] Z. Benenson, E. Blaß, and F. C. Freiling. Attacker models for wireless sensor networks (angreifermodelle für drahtlose sensornetze). *it - Information Technology*, 52(6):320–324, 2010.

[10] D.P. Bertsekas. *Nonlinear programming*. Athena Scientific optimization and computation series. Athena Scientific, 1999.

[11] A. Bondy and U.S.R. Murty. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2008.

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

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

[14] Z. Burda, J. Duda, J. M. Luck, and B. Waclaw. Localization of the maximal entropy random walk. *Physical Review Letters*, 102:160602, Apr 2009.

[15] C. Chekuri, N. Korula, and M. Pál. Improved algorithms for orienteering and related problems. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 661–670, San Francisco, CA, USA, 2008. Society for Industrial and Applied Mathematics.

[16] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 302–308, Beijing, China, September 2004.

[17] E. Cinlar. *Introduction to Stochastic Processes*. Dover Books on Mathematics Series. Dover Publications, Incorporated, 2013.

[18] F. Clarke. *Optimization and Nonsmooth Analysis*. Society for Industrial and Applied Mathematics, 1990.

[19] J. Cortes. Discontinuous dynamical systems. *IEEE Control Systems*, 28(3):36–73, 2008.

[20] M.H. DeGroot. *Optimal Statistical Decisions*. Wiley Classics Library. Wiley, 2004.

[21] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[22] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the L1-ball for learning in high dimensions. In *International Conference on Machine Learning*, pages 272–279, Helsinki, Finland, 2008.

[23] B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34(3):307–318, 1987.

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

[25] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Probability and Random Processes. OUP Oxford, 2001.

[26] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer, 1976.

[27] L. Kleinrock. *Queueing Systems. Volume I: Theory*. John Wiley, 1975.

[28] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*, volume 21 of *Algorithmics and Combinatorics*. Springer, 4 edition, 2007.

[29] G. Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416, 2009.

[30] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[31] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[32] F. Pasqualetti, A. Franchi, and F. Bullo. On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms. *IEEE Transactions on Robotics*, 28(3):592–606, June 2012.

[33] M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler. A stochastic and dynamic vehicle routing problem with time windows and customer impatience. *ACM/Springer Journal of Mobile Networks and Applications*, 14(3):350–364, 2009.

[34] D. Portugal and R. P. Rocha. Multi-robot patrolling algorithms: examining performance and scalability. *Advanced Robotics*, 27(5):325–336, 2013.

[35] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006.

[36] S.M. Ross. *Stochastic processes*, volume 2. John Wiley & Sons New York, 1996.

[37] T. Sak, J. Wainer, and S. K. Goldenstein. Probabilistic multiagent patrolling. In *Proceedings of the 19th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, SBIA '08, pages 124–133, Savador, Brazil, 2008. Springer-Verlag.

[38] M.W.P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations research*, 4(1):285–305, 1985.

[39] S. L. Smith, M. Schwager, and D. Rus. Persistent robotic tasks: Monitoring and sweeping in changing environments. *IEEE Transactions on Robotics*, 28(2):410–426, 2012.

[40] K. Srivastava, D.M. Stipanovic, and M.W. Spong. On a stochastic robotic surveillance problem. In *IEEE Conference on Decision and Control and Chinese Control Conference*, pages 8567–8574, Shanghai, China, Dec 2009.

[41] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1 – 10, 2011.

[42] J. Yu, S. Karaman, and D. Rus. Persistent monitoring of events with stochastic arrivals at multiple stations. pages 5758–5765, May 2014.