Searching and Patrolling Dispersed Locations

Edward Mellor, MMath, MRes

Submitted for the degree of Doctor of Philosophy at Lancaster University.

September 2024

Abstract

There are many real-world scenarios in which hidden objects or targets must be found. In searching for these targets, it may be necessary to traverse great distances, and practical search strategies should take into account the costs of moving between different locations. In this thesis, we present a discrete-time search model and a continuous-time patrol model that both explicitly take costs of movement into account.

The search problem involves looking for a target which has been hidden in one of finitely many geographically distinct locations according to a known probability distribution. A searcher moves between these locations in order to find the target. For each location, a search takes some known amount of time to complete and independently finds the target with a known probability if the target is there. The searcher aims to minimise the expected total amount of time needed to find the target. The version of our problem without travel times can be solved to optimality using Gittins indices, which direct the searcher to always search the location that yields the maximal rate of target discovery. When travel times are included, the problem becomes much more challenging because the searcher becomes less willing to move to another location due to the potential future travel time back to the current location. In addition, when choosing the next destination, the searcher needs to take into account not only the distance to each destination, but also each destination's distance to all other locations. We draw upon restless bandit theory to derive an index heuristic that takes travel times into account, and show that this heuristic has a tendency to leave a location prematurely. Subsequently, we use a range of methods to improve the index heuristic and demonstrate its strong performance via computational experiments.

The patrol problem also involves searching for a target (now perhaps best thought of as an attacker) among finitely many geographically distinct locations. Rather than being present at the start of the search, these attackers arrive over time. It is therefore necessary for the patroller to patrol continuously, visiting all locations infinitely often, to catch these attackers. Once the patroller arrives at a location, they can spend any amount of time searching for targets with some detection rate at that location, before moving to a different location. The objective of the patroller is to minimize the expected time an attacker stays undetected at a location regardless of where the attack occurs. In the special case where all travel times are set to zero, we elucidate an optimal cyclic policy in which the patroller allocates a fixed fraction of their effort to each location continuously. As in the case of the search problem, this patrol problem becomes significantly more challenging when travel times are introduced. We define two cycle types based on common patrol practice for perimeter patrol and border patrol, respectively, and derive formulae for the expected time to detecting an attack in each case. We also provide an algorithm for finding the best parameters for each cycle type subject to some unimodality conditions. We give several examples where these cycle types perform well and numerically demonstrate that the optimal patrol policy depends highly on the structure and parameters of each patrol problem.

Acknowledgements

I would like to begin by expressing my gratitude to my supervisors Kevin Glazebrook, Rob Shone and Kyle Lin for their invaluable guidance and unwavering support throughout my PhD journey. It has been a privilege to work alongside each of you, and I am deeply thankful for the time, effort, and expertise you have invested in helping me grow as a researcher. Kevin, I wish you a long and fulfilling retirement. Rob and Kyle, I look forward to continuing our collaborations in the future.

I would also like to extend my thanks to my colleagues at the EPSRC-funded STOR-i Centre for Doctoral Training. I am especially grateful to the leadership and administrative teams for their dedication and hard work in ensuring the smooth running of the centre. To all the STOR-i students, past and present, thank you for creating such a welcoming and supportive environment. The camaraderie in the office has made this journey all the more enjoyable. A special mention goes to my 'support bubble' during the challenging first year of the PhD, marked by the COVID-19 pandemic: Dr. Matthew Darlington, Dr. Peter Greenstreet, and international athlete Dr. Hamish Thorburn. Your friendship and encouragement made all the difference.

To my friends outside the office, thank you for providing balance, joy, and perspective. Whether it was sharing moments of laughter, offering a listening ear, or helping me unwind after long days, your presence has been a cherished escape from the rigors of research life.

I would also like to take this opportunity to thank my family for their constant support and encouragement. Your belief in me and your unwavering confidence that I would succeed has kept me going through the most challenging moments. I am so grateful for your love, patience, and the countless ways you've helped me along the way.

Finally, to my partner Matthew, I cannot thank you enough for your love, patience and constant support. You have been my rock throughout this journey, always understanding and encouraging me even when things got tough. Your humour and silliness never fails to brighten my mood, and I am incredibly grateful for everything you've done to help me reach this point.

Declaration

I declare that the work in this thesis has been done by myself and has not been submitted elsewhere for the award of any other degree.

A version of Chapter 3 has been submitted for publication by Mellor, E., Glazebrook, K. D., Lin, K., and Shone, R. in 2024.

The word count for this thesis is approximately 44, 300 words.

Edward Mellor

Contents

Abstract **I**

Acknowledgements **III**

Declaration **V**

Contents **VIII**

List of Figures **IX**

List of Tables **X**

1 Introduction **1**

1.1 Motivation and Background 1

1.1.1 Search Problems 1

1.1.2 Patrol Problems 3

1.2 Contributions 4

1.2.1 Search Contributions 4

1.2.2 Patrol Contributions 5

1.3 Outline of Thesis 6

2 Literature Review **8**

2.1 Search Literature 8

2.1.1 A Classical Discrete Search Problem 9

2.1.2 Search Objectives 11

2.1.3 Search Space 15
2.1.4 Target Properties 16
2.1.5 Detection Process 19
2.2 Patrol Literature 21
2.2.1 Resource Allocation 21
2.2.2 Robotic Patrolling 23
2.2.3 Patrolling Games 25

3 Searching Dispersed Locations 29
3.1 Introduction 29
3.2 Model Formulation and Preliminaries 33
3.2.1 Formulation as a Semi-Markov Decision Process 33
3.2.2 On the Optimal Policy 35
3.2.3 Value Approximation via Finite-stage Dynamic Programming . 38
3.3 A Heuristic Method Based on Indices 40
3.3.1 Development of an Index Heuristic 40
3.3.2 Properties of the Index Heuristic 44
3.4 Search Sequence Generation and Improvement 46
3.4.1 Other Index-based Search Heuristics 48
3.4.2 Methods for Improving a Search Sequence 51
3.5 Numerical Study 53
3.5.1 Scenario Generation 54
3.5.2 Results for 4 Locations 55
3.5.3 Results for 8 Locations 58
3.6 Conclusions 60
3.7 Appendix 61
3.7.1 Proof of Theorem 3.1 61
3.7.2 Proof of Theorem 3.4 63

4 Patrolling Dispersed Locations 67
4.1 Introduction 67

4.2 The Case of No Travel Times 70
4.3 A Simple Cycle 74
4.3.1 Motivation for a Simple Cycle 75
4.3.2 Optimising a Simple Cycle 79
4.3.3 The Case of Homogeneous Locations 84
4.3.4 The Case of Heterogeneous Locations 88
4.4 A Sweep Cycle 89
4.4.1 Motivation of a Sweep Cycle 89
4.4.2 Optimising a Sweep Cycle 91
4.4.3 The Case of Homogeneous Locations 92
4.4.4 The Case of Heterogeneous Locations 93
4.5 Numerical Demonstration 94
4.5.1 Ternary Search 95
4.5.2 Nested Ternary Search 96
4.5.3 Examples 100
4.6 Conclusions 109
4.7 Appendix 111
4.7.1 Supporting Evidence of Conjecture 4.5 111
4.7.2 Proof of Proposition 4.10 116

5 Conclusions and Further Work **119**
5.1 Summary 119
5.2 Further Work 121
5.2.1 Extensions of the Search Model 121
5.2.2 Extensions of the Patrol Model 122
5.2.3 Extensions of Other Models 123

Bibliography **124**

List of Figures

4.1 Number of customers over time in steady state 77

4.2 Search time vs expected time to discovery 87

4.3 Search time vs absolute sub-optimality of the expected time to detection 87

4.4 Surface showing convexity for a simple cycle 98

4.5 Contour plot showing convexity for a simple cycle 98

4.6 Surface showing convexity for a sweep cycle 99

4.7 Contour showing convexity for a sweep cycle 99

4.8 Three location examples with unit travel times 100

4.9 Three location examples with points equally spaced along a line 102

4.10 A three location example with points unevenly spaced along a line . . . 104

4.11 A three location example in a stretched triangle 105

4.12 Four location examples in a ring 106

4.13 Six location examples in a line 108

4.14 Six location example in a ring 109

List of Tables

3.1 Comparisons between heuristics with respect to percentage suboptimalities and runtimes with $n = 4$. 57

3.2 Comparisons between heuristics with respect to percentage deficits and runtimes with $n = 8$ 59

4.1 Best search duration for a 2 location patrol problem 86

4.2 Three location results for simple cycles and sweep cycles 102

Chapter 1

Introduction

1.1 Motivation and Background

1.1.1 Search Problems

Searching has been an important human activity for tens of thousands of years. Early hunter-gatherers spent much of their time searching for food and shelter. In times of war, intelligence teams use drones and satellite imaging to search for hostile assets such as incoming missiles and hidden military installations. Today, there are many other scenarios where one or more hidden targets—which could be people, objects or even something intangible like solutions to a problem or a new job—need to be found. Examples of such targets include survivors of a natural disaster in a collapsed building, a concealed explosive planted by terrorists and a lost child in an unfamiliar city. In all of these scenarios, it is necessary to send one or more agents, usually called searchers, to investigate in the hope of discovering the target as soon as possible. A searcher could be a person or team of people, possibly with specialised equipment such as a metal detector or trained sniffer dogs. Alternatively an unmanned drone or directed satellite imagery could be used to explore a location. Failing to find the target quickly can have costly and potentially life-threatening repercussions.

Search problems are a broad class of problems which involve searching for one or more targets that are hidden, either intentionally or via some natural process, within a region known as a search space. Searchers are able to traverse the search space and may expend some time or effort, which could be physical or fiscal, to search a section of the space. The search procedure continues until either the target is found or the searcher gives up, possibly due to reaching the limits of a predetermined budget. The quality of a search policy is quantified by a search objective. Developing optimal or heuristic strategies for these problems falls within the purview of an area of operations research called search theory.

The properties of the search space can have a huge impact on what sort of search policies perform well. Some search problems, such as a marine salvage project or a country-wide manhunt for a dangerous criminal, take place over very large areas. This can be compounded by the requirement for specialist search equipment that can be difficult to transport. If the effort required for the searcher to traverse the search space is not taken into account, then a policy cannot be expected to perform well. Despite this, travel times are often not taken into account in the existing search literature. This is largely because of the additional complexity added by including travel times in mathematical formulations of search problems.

Motivated by this gap in the literature, a major part of this thesis is concerned with the following discrete search problem. A searcher wishes to find a static target hidden in one of several locations according to a known probability distribution. The searcher's goal is to minimise the expected time to find the object. She is able to travel between locations, which takes some time and depends on the locations being travelled between. She can also carry out searches at each individual location. Each search of a location takes a known amount of time and, if the target is there, the search will be successful with a known probability. The search times and detection probabilities are fixed throughout the search but may vary between locations.

An optimal solution to the version of this problem without travel times was first derived in 1962 by David Blackwell in some unpublished notes (see Blackwell (1962)) and reported by Matula (1964). This policy mandates always searching a location with the highest probability of detection (taking into account any previous unsuccessful searches) per unit of time.

A little over a decade later Frank Kelly commented on Gittins (1979) that the version of our problem without travel times is equivalent to a particular type of multi-armed bandit (MAB) problem. The MAB problem is a mathematical model first proposed by Thompson (1933) that takes its name from slot machines. When activated, a one-armed bandit yields a reward which is drawn randomly from an unknown distribution. This reward may also depend on the state of the bandit, which evolves each time the bandit is activated. In a MAB problem a decision maker is faced with multiple one-armed bandits which they must activate sequentially to maximise the long-term reward. The more times the decision maker activates a particular bandit, the better their understanding of that bandit's reward distribution. Kelly observed that our classical search problem is equivalent to a MAB problem where searching a location is equivalent to activating a corresponding bandit and that the index solution of Gittins (1974) for the MAB problem is identical to the policy derived by Blackwell. This index policy is commonly referred to as the Gittins' index policy.

1.1.2 Patrol Problems

In Section 1.1.1 we introduced a class of search problems where one or more targets, which are hidden prior to the search, need to be found. An intrinsically related class of problems involves searching for targets that appear over time. Throughout such a process, there could be times where multiple targets are present and other times where none are present. Examples of targets in this type of problem include mechanical faults in a complex system of machines, smugglers attempting to cross a border and customers in a department store that are in need of assistance.

In some contexts, we would expect that after a target arrives somewhere they remain there until they are discovered. This may be the case for a mechanical fault that remains broken until it is detected and repaired, or for a spy who continues to leak information until they are discovered. In other contexts, a target may have a specific task they wish to accomplish such as stealing a painting or crossing a border. In these cases, a suitable model will cause the target to disappear after some amount of time. Detecting the target before they have completed their designated task will usually negate all negative effects associated with their appearance. In either case it is necessary to patrol the search space continuously. Thus, we describe such problems as patrol problems and rename the agent who searches for the targets as the patroller.

In this thesis we consider a patrol problem where a patroller wishes to protect a set of dispersed locations from attacks. She can travel between locations and spend any amount of time searching any individual location. The times required for the patroller to travel between pairs of locations are known and the detection rates, which may vary from location to location, are also known. The patroller wishes to minimise the maximum expected amount of time to discover an attacker after they arrive at a particular location, where the maximum is considered over all locations.

1.2 Contributions

The main aim of this thesis is to explore search and patrol problems that explicitly take into account the additional time and cost of moving between locations. To this end, we focus on two problems. The first is the search problem described in Section 1.1.1 that generalises an extensively studied classical discrete search problem to include travel times. The second is the patrol problem described in Section 1.1.2 that involves continuously patrolling a set of dispersed locations to protect them from attacks. The contributions in each of these areas are summarised in Sections 1.2.1 and 1.2.2 respectively.

1.2.1 Search Contributions

To give context to our contribution to the search literature we provide a thorough review of existing work, with a particular focus on the search problems similar to the classical search problem that our problem generalises.

After formulating our search problem we show that, despite not taking travel times into account, the index policy for the version of the problem with no travel times still has some useful properties when applied to a problem with travel times. In particular, we show that a searcher following this policy only remains at her current location when it is optimal to do so. Motivated by this insight, we draw upon the theory of restless bandit problems to derive an index-based heuristic policy that explicitly takes into account the travel times between different locations. To gain further insights into the performance of this policy, we analytically derive the form of the optimal policy in the special case of two locations with identical parameters. We show that our index heuristic has a similar structure to the optimal policy in this case but a searcher following the index heuristic tends to leave her current location prematurely. Additionally, we are able to show that for this simple two-location problem, our index heuristic only instructs the searcher to remain at her current location when it is optimal to do so. We are also able to show this property holds in more general n-location problems with heterogeneous parameters provided that all travel times are the same.

In order to address the shortcomings of the new index heuristic we introduce two additional heuristics that make use of these indices, and show how the performance of any search sequence can be further improved by performing extra steps based on insertion heuristics and policy improvement. We provide extensive numerical results to evaluate the performances of our heuristics with respect to the approximated optimal values. By examining these results, we highlight the benefits of the improved methods using insightful comparisons.

1.2.2 Patrol Contributions

Our contribution to the patrol literature begins with a brief review of existing work. Unlike the search problem discussed previously, the version of our patrol problem without travel times has not been extensively studied. Thus, we initially study this special case. When all travel times are set to zero, locations can be moved between instantly, allowing the patroller to effectively divide her search effort between multiple locations simultaneously. We show that in this case it is optimal whenever operating according to a patrol pattern which repeats (which we call a cycle) to continuously allocate a constant fraction of effort to each location. We show that at each location the fraction of effort that should be allocated is inversely proportional to the detection rate.

For the more general case we introduce two cycle types, referred to as simple cycles and sweep cycles. A simple cycle is a patrol pattern that visits each location exactly once in a set order and then repeats. Thus, each search of a given location lasts for the same amount of time and then every other location is searched before the patroller returns to that location. To support the intuition of simple cycles being effective, we show that if a cycle visits a location twice then both of these visits should be of the same duration. We show that the order in which locations are visited in the best simple cycle must correspond to a minimum Hamiltonian cycle. We also show that in the special case where all detection rates are the same, all locations should be visited for the same amount of time. A sweep cycle is motivated by scenarios where locations are spread out along a line. The patroller moves back and forth along the line in searching each location when passing in each direction. We show that when the locations are arranged in a line, the best sweep cycle will always perform better than the best simple cycle.

For both cycle types we derive a formula for the maximum expected time to discovery across all locations. We present a nested ternary search algorithm to find the vector of search durations that minimises this expression. We then use this algorithm to find the best simple cycle and the best sweep cycle in a variety of numerical examples. For some specifically chosen scenarios we compute the best instances of other cycle types in order to demonstrate that an optimal patrol pattern does not necessarily need to be either a simple cycle or a sweep cycle.

1.3 Outline of Thesis

The remaining chapters of the thesis are as follows:

Chapter 2 provides an in-depth review of the relevant literature. The first section of this chapter focuses on search theory and starts with a more detailed account of the version of the search problem described in Section 1.1.1 without travel times. Particular emphasis is given to the index solution and its connection to the multi-armed bandit problem. The section also explores a variety of related search problems to give context for our contribution. The second section of this chapter focuses on the patrol literature. This part is separated into three subsections on resource allocation, patrolling games and robotic patrol.

In Chapter 3 we study the search problem described in Section 1.1.1. We initially establish some theoretical results about the performance of the Gittins' index policy which is known to be optimal for the version of the problem without travel times, when applied to a problem with dispersed locations. Motivated by these results we develop an index heuristic and compare it to an optimal policy in some special cases. To address the shortcomings of this heuristic we develop two other heuristics that make use of these indices. We also present two further algorithms that can be used separately or together to augment the performances of these base heuristics. The chapter concludes with a numerical study evaluating the effectiveness of these heuristics.

In Chapter 4 we study the patrol problem described in Section 1.1.2. We initially establish the form of the optimal cyclic policy for the case with no travel times. We then introduce two cycle types: simple cycles and sweep cycles. For each cycle type we provide some theoretical results and then derive a formula for the maximum expected time to discovery across all locations. We present an algorithm based on ternary search to find the best instance of each of these cycle types. We then examine their performances in a series of specially chosen examples and discuss how the ternary search algorithm can be used to find the best patrol pattern of any given cycle type.

Chapter 5 concludes the thesis and provides some ideas for further work.

Chapter 2

Literature Review

This chapter is divided into two sections. Section 2.1 reviews the relevant literature on search problems. Since this body of work is so extensive, with contributions made across nine decades, providing a full account is beyond the scope of this chapter. We therefore focus on studies that are particularly relevant to the discrete search problem discussed in Chapter 3. In Section 2.2 we turn our attention to patrol. While the literature on patrol problems is substantial, it is not as extensive as the work on search. This allows us to present this section as a more general study of patrol rather than focusing on a specific type of problem as was required for the literature on search.

2.1 Search Literature

The Anti-Submarine Warfare Operations Research Group was founded during World War II by the US Navy. The group were tasked with developing novel methods for the rapid detection of hostile marine vessels. During this time, they established many of the underlying principals of search theory. After the war, Bernard Koopman, one of the group's lead contributors, compiled this work into a book called 'Searching and Screening' (Koopman, 1946) which was classified as confidential by the US Navy until 1958. In the decades since, search theory has been applied to countless other problems which has led to the development of an expansive body of literature. See Enslow Jr (1966), Dobbie (1968), Stone (2004), Washburn (2014) and Stone et al. (2016) for surveys of this work.

While the many diverse contexts for search problems make it necessary to use a wide variety of different models, there are often the following common elements. A decision-maker, who is usually referred to as the searcher, wishes to find a target which has been hidden in a region known as the search space. A probability distribution over the search space, known as the target distribution, indicates how likely the target is to be in different parts of the search space. The target distribution may be known to the searcher or determined by another decision maker who may wish to positively or negatively affect the outcome of the search. The searcher's actions are determined by a search policy. As she traverses the state space, the searcher can expend some resource, usually time or money, to search her immediate vicinity for the target. If the target is present, the success of the search is determined by a detection function. This process continues until either the target is found or the searcher gives up, possibly after reaching some limit on the resource being used to search. A search objective is used to quantify the effectiveness of a particular search policy.

In this section, we review elements of the search theory literature. Section 2.1.1 presents a classical discrete search problem that was first studied in the 1960s. This search problem and its solutions are of particular significance to us as it is a special case of the search problem considered in Chapter 3 where all travel times are set to zero. The remaining subsections consider extensions and variations of this problem. Section 2.1.2 considers a variety of different search objectives. Section 2.1.3 considers a variation of this problem with a continuous search space. Section 2.1.4 discusses several extensions that make different assumptions about the target. Finally, Section 2.1.5 considers alternative detection processes.

2.1.1 A Classical Discrete Search Problem

In this section we consider a classical discrete search problem. Extending this problem to include travel times is the focus of Chapter 3.

In this problem, a target is hidden randomly in one of n distinct sub-regions, for some $n \in \mathbb{N}$, which are labelled 1 to n arbitrarily. In the literature these sub-regions are usually called either ‘boxes’, ‘cells’ or ‘locations’. For consistency, the word ‘locations’ will be used throughout the rest of this thesis. For ease of notation, we use $[n]$ to denote the set $\{1, \cdots, n\}$. We assume that the target is hidden at random according to a known probability distribution $P = (p_1, \cdots, p_n)$ where, for each $i \in [n]$, $p_i \in (0, 1)$ denotes the probability that the target is hidden in location i and $\sum_{i=1}^{n} p_i = 1$. The target remains hidden and does not move until it is found. It is assumed (in the classical problem) that these locations are sufficiently close together that the searchers do not need to expend any time or effort to move between them. For each $i \in [n]$ the searcher can choose to spend t_i units of time to carry out a single search of location i. If the target is in location i, then this search is successful with a probability q_i regardless of the number of times that location i has previously been searched. The exact position within the location that the target is hidden has no effect on the search time or detection probability. Also, if the target is discovered part way through a search we assume that the searcher must complete her current search, which is included in the time to detection. The searcher’s objective is to determine a sequence of locations to be searched to minimise the expected time to discovery.

As discussed in Chapter 1, the expected search time for this problem can be minimised using an index policy. This result was first derived by David Blackwell using dynamic programming techniques. This result was published by Matula (1964). Black (1965) independently derived a graphical proof of this result. Frank Kelly also commented on Gittins (1979) that this problem is equivalent to a particular multi-armed bandit problem and so the results of Gittins (1974) could be used to obtain the same index policy. Since then this index policy has become commonly known as a Gittins’ index policy. Almost two decades after this result was first proved, Ross (1983) found another proof using a simple interchange argument.

Formally, the Gittins’ index policy works as follows. At the start of the search procedure and following each unsuccessful search, the searcher is directed to search a location with the highest value of the index $G_i(s_i)$, where for each $i \in [n]$,

$$G_i(s_i) = \frac{p_i q_i (1 - q_i)^{s_i}}{t_i} \tag{2.1}$$

and s_i denotes the number of times that location i has previously been searched. If multiple locations have the maximal value of $G_i(s_i)$ then searching either will result in an optimal policy.

At the start of the search the target distribution P represents the searcher's beliefs about the location of the target. Using this as a prior, the searcher can use Bayes' theorem to update this belief distribution following each unsuccessful search. Write $P' = (p'_1, \cdots, p'_n)$ for the posterior belief distribution. If, for each $j \in [n]$, location j has previously been searched s_j times then for each $i \in [n]$,

$$p'_i = \frac{p_i(1-q_i)^{s_i}}{\sum_{j=1}^{n}(1-q_j)^{s_j}}.$$

Since the denominator is the same across all locations it follows that $p'_i \propto p_i(1-q_i)^{s_i}$ and so an index policy with indices given by

$$\frac{p'_i q_i}{t_i}$$

is equivalent to the index policy using (2.1). This latter index has a simple interpretation as the probability of detection of the target at i per unit of time.

Search problems can also be formulated as a game between the searcher and an adversary who chooses where the target is hidden. These games are commonly referred to as search games or as two-sided search problems. Hohzaki (2016) and Alpern and Gal (2002) provide extensive literature reviews on search games. Clarkson et al. (2023) formulate the classical discrete search problem described above as a search game between the searcher and an adversary who chooses the initial hiding distribution P with the objective of maximising the expected time to discovery. Neither decision maker has any knowledge of the other's chosen policy, making this a simultaneous two-person zero-sum game.

2.1.2 Search Objectives

A search objective is used to assess the quality of a search policy. The search objective of the classical discrete search problem discussed in Section 2.1.1 is to minimise the expected time to discovery. In this section we introduce some alternative search objectives.

Maximising Probability of Discovery by a Deadline Maximising the probability of discovery subject to a resource constraint is another common search objective. Having a time limit for a search makes sense in scenarios where finding an object is only useful if done within a given time-frame. For example, when searching for a bomb, the searcher needs to find it before it detonates and with enough time for a bomb disposal unit to safely defuse it. Alternatively, there may be practical reasons why the searcher would only have access to the search space for a certain period of time. This is likely to be the case if the search space has some other public or commercial use and must be closed while the search is active. This objective is also useful for search procedures with a financial constraint. Where the target has some fiscal value, this may be used to inform how much an organisation is willing to spend on the search procedure. In other cases, particularly where human life is at stake, governments are often forced to face difficult decisions about how much they are willing to spend to find a missing person. While having a lower expected time to discovery is beneficial in this case, this would be a secondary objective and is not addressed within the objective function.

Chew et al. (1967) introduces a variation of the classic discrete search model which uses this objective. He shows that using the index policy defined by (2.1) until the next proposed search exceeds the constraint is optimal under certain conditions. The index policy is not optimal in general because the final search may not use up the full budget. If the finite resource is not used in its entirety, it is possible that the time expended on the last few searches could be more effectively utilised by a sequence of searches that are able to use it more completely. If, by coincidence or by design, the budget is fully expended during the final search of the index policy then this truncated search sequence is optimal. Additionally, if all searches have the same duration then the truncated index policy will also be optimal. Some other papers that use this objective function are Kadane (1968), which considers a more general problem with time-dependent detection functions, and Wegener (1982), which constructs an algorithm for a search model with time-dependent search times.

Lau et al. (2008) also also consider a search problem where the searcher wishes to maximise the probability of discovery by a known deadline. In this model, the target is hidden in a discrete search space according to a known probability distribution. The distance between these locations is represented by an adjacency matrix, where only adjacent locations can be moved between. The search takes place over discrete, evenly spaced time steps. At the start of each time step the target moves to a new location according to a known Markov model. At the same time, the searcher chooses to either remain at their current location or to move to an adjacent location. The searcher then spends the time step searching the chosen location. In the first part of the paper, the searcher can move instantly between locations. In the second part of the paper, travel times are included for the searcher. For each pair (i, j) of adjacent locations it takes a number of time steps $d_{ij} \in \mathbb{N}$ to move between these locations. While moving the searcher is unable to take any further actions. For both versions of the problem, a branch and bound approach is used to find search strategies.

Subelman (1981) introduces a search game with unit search times where the searcher wishes to maximise the probability of finding the target by a known deadline. Lin and Singham (2016) also consider a search game with this objective but where the searcher does not know the deadline.

Maximising a Reward Ross (1969) considers a version of the classical discrete search problem presented in Section 2.1.1 with search costs in place of search times. The searcher also receives a reward R_i if the target is discovered in location i. The searcher wishes to maximise their expected reward minus total cost of the search. It is clear that if the reward is not sufficiently high then the expected cost of the search will exceed the expected reward. Thus the searcher is also allowed to stop the search at any point. Ross (1969) shows that an optimal policy will either search the location with the largest index as defined by (2.1) or remove that location from consideration. A location may be removed from consideration if the reward for finding the target there is low. Moreover, if the reward is the same for all locations then the optimal policy either searches the location with the highest index or the search is abandoned.

Sweat (1970) considers an alternative objective where the searcher wishes to maximise the expected reward which is received upon finding the target. For each $i \in [n]$

a search of location i discounts the reward by a factor of β for some known $\beta \in (0, 1)$. Thus, if s_i is the number of times that location i has been searched when the target is discovered then the final reward is discounted by $\prod_{i=1}^{n} \beta^{s_i}$. Sweat (1970) shows that the searcher can maximise the expected reward by using a variation of the index policy given by (2.1) where t_i is replaced by $1 - \beta$.

Information-Based Objectives In some cases, it is not necessary for the searcher to find the target. Mela (1961) introduces the notion of instead maximising the information gain by a deadline, and demonstrates with a few simple examples that a policy that maximises the probability of detection by a certain deadline may not maximise the information gain. In one particular example the searcher has some amount of time to search for the target and when her time is up she must guess the target's location. Her objective is to maximise the probability that this guess will be correct. Since the target is stationary, finding the target before the deadline means that she is certain that her guess is correct. However, if she is unable to find the target she can take the initial target distribution and perform a Bayes' update using the unsuccessful searches she was able to complete. She then guesses that the target is in the location with the largest posterior probability. An example of where this might be used is in a hostage rescue, where an initial search is done using remote sensing equipment and then an evacuation team is sent to the location where the hostages are most likely to be. Such problems have since been called whereabouts searches.

Mela (1961) considers examples of whereabouts searches with at most three locations and unit search times. Tognetti (1968) generalises this work to a problem with n locations, still with unit search times. He observes that an optimal strategy for this problem should not include any searches of the location that has the highest posterior probability when the guess is made. This is because if the target is hidden there then the whereabouts search will definitely succeed even if the target is not found. On the other hand, if the target is not there then there is no benefit to searching it and by spending time searching some other locations the searcher still has a chance to find it. Kadane (1971) generalises this to the classical discrete search problem. He demonstrates that the optimal policy is as follows. First, identify the location with the highest prior probability and temporarily remove it from consideration. Secondly, construct a policy that maximises the probability of discovery by the decision deadline using the results of Kadane (1968). If the target is found, then guess the location where it was detected. Otherwise, guess the location which was removed at the start of the search.

2.1.3 Search Space

When modelling a search procedure, it is necessary to limit the scope of the search to a clearly defined region called a search space. A search space is a set representing all feasible places a target could be hidden and can be categorised as either discrete or continuous.

The classical discrete search problem introduced in Section 2.1.1 uses a discrete search space. A discrete search space is a countable set of distinct points and is typically used when there are clearly distinct locations in which the target can be hidden. Some applications that a discrete search space would be suitable for are a search for a hidden stash of contraband in the rooms of a hotel and an intelligence service searching for a possible mole among their staff.

Where a discrete search space is not applicable we can instead use a continuous search space. A continuous search space is a subset of $\mathbb{R}^m$ for some $m \in \mathbb{N}$ and can be used to represent a projection of the ocean floor when searching for the wreckage of a ship or an airport when searching for hidden explosives following a tip-off.

Much of the early work in search theory focused on continuous search spaces. Koopman (1957) considers the problem of distributing a fixed amount of effort across a continuous search space to maximise the probability of detecting the target. This model relies on the assumption that the detection functions, which give the probability of discovery at a given point depending on the amount of effort allocated to it, are exponential. De Guenin (1961) extends this model to include more general detection functions.

Often when faced with a problem that would be best represented by a continuous search space it is sufficient to partition the space and thereby approximate it by a discrete search space. Richardson et al. (1971) describes how this approach was used during the underwater search for the remains of the submarine Scorpion.

2.1.4 Target Properties

For some problems, it is reasonable to assume that the target is equally likely to be in any location whereas in other problems we may have some knowledge about the hiding process that we wish to include in the model. The mechanism we use to incorporate this information is a target distribution. A target distribution is a probability distribution over the search space that indicates how likely a target is to be in a particular region. Defining the target distribution will often require specialist knowledge from a subject matter expert. Another property that needs to be considered about the target is whether it is static or mobile. A moving target may move at random or strategically. If the targets are moving with purpose it is important to consider what that purpose is, as well as what information the target has about the searcher.

Missing Target In the classic discrete search problem we assert that $\sum_{i=1}^{n} p_i = 1$ where for each $i \in [n]$, p_i is the probability that the target is hidden in location i. In other words, the target must be in one of the n locations. Now we present a model where there is a possibility that $\sum_{i=1}^{n} p_i < 1$ and so there is some non-zero probability that the target does not exist in the search space. The possibility that the target is not in any of the established locations can be incorporated into the model as an additional location which cannot be searched. The searcher's objective remains to minimise the expected search time. However, since there is a possibility that the target is in a location that cannot be searched, the expected time to discovery under any policy is infinite. It is therefore necessary to include the option to terminate the search by incurring some time penalty c which is known. The objective is therefore updated to minimise the expected time to conclude the search with the understanding that the search will only end when either the target is found or after the searcher has given up and paid the additional cost c. Chew (1973) shows that a truncation of the sequence generated by following the index policy defined by (2.1) is optimal for this problem. Rather than determine the optimal point of truncation directly, Chew (1973) instead considers a stopping region S_B. This is a set of posterior probabilities for which the expected time to discovery is equal to the penalty time c. In order to 'bound' this set Chew (1973) is able to prove that $S_L \subset S_B \subset S_U$, where

$$S_U = \left\{ P : P = (p_1, \cdots, p_n),\ \max_{i \in [n]} (p_i q_i) \leq 1/c \right\},$$

and

$$S_L = \bigcap_{i \in [n]} \left\{ P : P = (p_1, \cdots, p_n),\ \left[p_i q_i + \frac{1}{c \min_{i \in [n]} q_i} \sum_{j \in [n] \setminus \{i\}} p_j q_i \leq \frac{1}{c} \right]\ \forall i \in [n] \right\}.$$

Thus, if we define s_L and s_U as the numbers of searches conducted while following the index policy until the posterior probabilities are contained in S_L and S_U respectively, then the optimal termination point falls between these values.

Multiple Targets Assaf and Zamir (1985) present a variation of the classical discrete search problem introduced in Section 2.1.1 where there are m targets, for some known $m \in \mathbb{N}$. The searcher's objective is to find the sequence of locations to search that minimised the expected time to discover the first target. Each target is hidden independently using the same target distribution. For each $i \in [n]$, let X_i denote the number of targets in location i. Then $\mathbf{X} = \{X_1, \cdots, X_n\}$ has a multinomial distribution. When a location is searched, the probability of discovering each target is q_i independently. Thus, the probability of finding at least 1 is $1 - (1 - q_i)^{X_i}$. The authors show that the expected search time can be minimised by following an index policy with indices given by

$$\frac{1 - \mathbb{E}_{\mathbf{X}}[(1 - X_i q_i (1 - q_i)^{s_i})^m]}{t_i},$$

where for each $i \in [n]$, s_i indicates the number of times that location i has previously been searched and the expectation in the numerator is taken over the current posterior for $\mathbf{X}$.

Smith and Kimeldorf (1975) consider a variation of the classical discrete search problem introduced in Section 2.1.1 where there is an unknown number of targets. Let M be a random variable with a known distribution indicating the number of targets.

Each target is hidden independently using the same target distribution. The searcher's objective is to minimise the expected time to discover one target. The authors show that if there are at least three locations, then there exists an optimal policy in the form of an index policy if and only if the number of hidden objects has a zero-truncated Poisson distribution. The indices for this index policy are given by

$$\frac{1-\mathbb{E}_M[(1-p_iq_i(1-q_i)^{s_i})^M]}{t_i}, \tag{2.2}$$

where for each $i \in [n]$, s_i denotes the number of times that location i has previously been searched. Here the numerator is the probability that a search will find a target and therefore cause the search to end. If the random variable M follows a zero-truncated Poisson distribution with parameter $\lambda > 0$, then (2.2) becomes

$$\frac{1-e^{-\lambda p_iq_i(1-q_i)^{s_i}}}{t_i}.$$

Assaf and Zamir (1987) consider the two-location version of this problem and show that the index policy defined by (2.2) is optimal for the zero-truncated Poisson distribution, and more generally for any distribution such that $\log(\mathbb{E}[M^x])$ is concave for $x \in [0,1]$.

Kimeldorf and Smith (1979) consider a variant of the model presented in Smith and Kimeldorf (1975) where the objective is to minimise the expected time to discover all targets. The authors show that, similar to when minimising the expected time to discover the first target, an optimal policy exists in the form of an index policy for any problem with $n \geq 3$ if and only if M is a zero-truncated Poisson random variable. In this case the indices are given by

$$\frac{e^{\lambda p_iq_i(1-q_i)^{s_i}}-1}{t_i},$$

where for each $i \in [n]$, s_i is the number of times that location i has been searched and λ is the parameter of the zero-truncated Poisson random variable.

Lidbetter (2013) and Lidbetter and Lin (2019) both consider search games where multiple objects are hidden among multiple locations. Lidbetter (2013) initially proposes a search game where k objects are hidden among n locations (with $n > k$). Each location i has a search cost c_i and can contain at most one object. When the searcher searches a location he finds any object hidden there. The searcher's objective is to find all objects as quickly as possible in the worst case scenario. An adversary, who chooses where the objects are hidden, wishes to maximise the time to find all objects in the worst case scenario. The author shows that the optimal search strategy is to randomly select a subset of k locations to search in any order. Each possible subset should be chosen with probability proportional to the product of their search costs. If not all objects are discovered then the remaining locations should be searched in a random order. A more general search game on a network is then introduced. The game considered by Lidbetter and Lin (2019) also includes k objects hidden among n locations. However, in this model, one location may contain multiple objects. Each time the searcher searches a location they pay a cost and if that location contains at least one object the searcher finds one of them.

2.1.5 Detection Process

In any search problem, the searcher is required to expend some resource to attempt to discover the target. In a discrete search space, the searcher is usually restricted to search only one location at a time. The expected resource could be time spent searching, the cost of the search, physical effort or some function of all of these. Throughout this section we will use time as the expended resource unless specifically stated otherwise.

Clearly, if the searcher searches location i and the target is not hidden there, then the probability of finding the target is zero regardless of how long the searcher searches. If the target is in the location being searched it may be appropriate to relate the amount of time spent searching a location to a probability of detection.

In the classical discrete search problem the searcher is restricted to searches of a certain length. This is called a discrete-time search model. Thus, if we wanted to write the detection functions for the classical discrete problem, they would be piecewise constant with jumps at regularly-spaced intervals.

Imperfect Specificity Kress et al. (2008) considers an extension to the classical discrete search problem presented in Section 2.1.1 where, for each location $i \in [n]$, there is a probability r_i that the searcher incorrectly believes they have detected the target. Thus, before ending the search, an additional verification step is required. For each $i \in [n]$, it takes v_i units of time to verify the discovery at location i. This verification process cannot overlook the target if it is there and has perfect specificity. Thus, if the target is not found during this step, the location can be removed from consideration going forward. The authors show that the optimal policy is an index policy. In particular, if for each $i \in [n]$, s_i is used to denote the number of times location i has previously been searched, then it is optimal to search the location with the largest value of

$$\frac{p_i q_i (1-q_i)^{s_i}}{t_i + r_i v_i}.$$

Note that this is the same as the index policy given by (2.1) for the classical problem but with the expected cost of the verification process added to the denominator.

Kress et al. (2008) also consider the case where it is only possible to carry out one verification process. This may be the case in a hostage rescue problem were any raids of locations other than those where the hostages are being held is likely to escalate the situation. In this case the authors suggest that the objective function should be to maximise the probability that the first detection is correct. They show that the optimal policy for this objective function is an index policy with indices given by a variation of (2.1), with the search time replaced by the false positive probability.

Different Search Modes Shechter et al. (2015) presents a search problem with two search modes where the act of carefully searching a location is dangerous due to enemy fire. Both search modes have perfect detection, meaning that any search of the target's location will discover it. If the searcher uses the fast mode to search a location then the searcher is not at risk but there is a probability p_Q that the target is damaged and the search results in failure. On the other hand, if the slow mode is used then there is no risk of the target being damaged but there is a probability p_S that the searcher is killed, which also means that the search fails. The objective in this problem is to maximise the probability that the search is successful.

Clarkson et al. (2020) extend the classical discrete search problem to include two search modes. A fast search of a location takes less time compared to a slow search but has a lower conditional probability of discovery. Clarkson et al. (2020) prove that if for each location the sequence of fast and slow searches to be carried out is fixed then a suitable version of the Gittins' index policy described in Section 2.1.1 is optimal. The authors also derive sufficient conditions for either mode to dominate the other, which optimally solves the problem for some specific examples. They also provide heuristic approaches for more general problems.

Following our review of contributions related to the discrete search model we now turn to a discussion of the literature related to problems in optimal patrol.

2.2 Patrol Literature

Patrol theory arose as an area of research in the 1970s. Much of the early work on patrol operations is focused on the optimal allocation of police patrol vehicles in various environments. This work is discussed in Section 2.2.1. A more recent area of research is robotic patrol. This involves one or more robots which are programmed to patrol an area to prevent attacks. Attackers can observe the robots' movements and may even have knowledge of the patrol algorithm being used. This knowledge can be taken into account when programming the robots to ensure a robust patrol pattern. This work is discussed in Section 2.2.2. The last few decades have seen a renewed interest in patrol problems. Much of this newer work takes a game-theoretic approach. The literature on patrol games is discussed in Section 2.2.3.

2.2.1 Resource Allocation

In this section we review the literature on resource allocation for patrol.

Local police departments are required to have an active presence in their community. Having police vehicles actively patrolling acts as a deterrent to potential criminals, allows for rapid response when crimes are reported and increases the chance that officers will witness a crime in progress. Police jurisdictions are often too large to be patrolled entirely in one go and so are divided into smaller regions. Larson (1972) reviews much of the early work on urban police patrol analysis, with a particular emphasis on informing strategic decisions such as the number of patrol vehicles to allocate to different regions. Both Chelst (1978) and Chaiken and Dormont (1978) present algorithms for allocating multiple patrol vehicles among several regions with known crime rates in an urban environment. These algorithms are shown to give significant improvements compared to simpler approaches based on allocating patrol resources in proportion to predicted crime rates for different regions.

Patrol models outside of urban environments often require special treatment. Lee et al. (1979) and Taylor III et al. (1985) consider the problem of patrolling highways. The work of Birge and Pollock (1989) involves developing specialised patrol methods for rural areas that put a greater emphasis on travel times between locations.

Some more recent papers in the patrol area are also concerned with resource allocation. Szechtman et al. (2008) considers a problem where a sensor sweeps along a section of a border which is chosen by the patroller to maximise the rate at which attackers are detected. Attackers arrive at a constant rate and the location of each attack is determined by a known probability distribution. Once an attacker arrives at the border they stay there for an amount of time given by another known probability distribution. Initially the authors assume that the sensor will instantly detect and apprehend any attacker it passes. Szechtman et al. (2008) also consider two extensions of this problem. The first extension allows the sensor to move more quickly at the risk of overlooking attackers, and the second extension requires the sensor to pause after discovering an attacker.

Olson and Wright (1975) consider a patrol problem in an urban environment where the patrol space is divided into discrete street segments, each of which has a known crime rate. In this problem, the objective is to maximise the rate at which patrol vehicles enter a segment in which a crime is in progress. The authors model the patroller's movement as a Markov Chain and use Monte Carlo techniques to create randomised patrol schedules which are shown to significantly increase the chances of detecting crimes compared to previous methods.

2.2.2 Robotic Patrolling

Another formulation for patrol problems involves programming robots, such as drones, to patrol some physical space. Chapter 2 of Basilico et al. (2012) reviews much of this work. Using a robot rather than a human to patrol can be particularly advantageous when the region being patrolled is potentially dangerous. This may be the case when attackers are armed or when performing surveillance in a hostile environment. Robotic patrollers also have the benefits of not experiencing any dip in performance as they get distracted or tired. However, robotic patrollers also have drawbacks when compared to their human counterparts. Human patrollers may sometimes move around the space under cover and thus the attackers are unaware of their presence, which is not the case for a drone flying overhead.

Random Attackers Many of the early contributions in this area assume that attackers do not take patrollers into account when choosing when and where to attack. Patrollers therefore wish to maximise the frequency of their visits to all locations in order to catch these attackers as quickly as possible.

Yanovski et al. (2003) consider the problem of n robotic patrollers who are required to patrol the edges of an arbitrary graph in such a way that the frequency between visits to each edge is approximately uniform. The authors introduce an evolutionary algorithm where each time a patroller arrives at a vertex they chose to next move along the edge that has been traversed least recently. They show that the algorithm performs well after an initial transient period and can adapt if edges are added to or removed from the graph.

Elmaliach et al. (2009) consider the problem of n robotic patrollers who are required to patrol a grid where some cells in the grid are blocked by obstacles and the remaining cells are vulnerable to attacks. The robots are able to move only in the cardinal direction. They may move at different speeds depending on the direction they are moving and changing direction may also take some additional time. The authors consider three possible objectives:

1. minimising the variance between the frequencies of visits to each location,
2. maximising the average frequencies between visits to each location and
3. maximising the minimum frequency between visits to each location.

The authors utilise an algorithm that finds the quickest cyclic path visiting all target areas. Patrollers are then distributed uniformly along this cycle and all move in the same direction. The authors show that this patrol pattern optimises all three objectives.

Strategic Attackers Other contributions assume that the attackers take the patrollers into account when planning their attacks. In most cases, the robots are programmed by a centralised decision maker and then sent out into the environment. The adversary (a decision-maker coordinating the actions of all attackers) can then observe the robots behaviour before choosing where to attack. It is often assumed that the adversary has full knowledge of the patrollers policy when choosing their own. This type of game is called a sequential game, a leader-follower game or a Stackelberg game.

Agmon et al. (2008a) and Agmon et al. (2008b) consider k robots patrolling a perimeter in discrete time. The perimeter is divided into segments that each take one time step for a robot to traverse. Attackers choose one segment to attack and the attack lasts t units of time. Agmon et al. (2008a) assumes that the k robots are equally spread out and move in such a way that they maintain their distance from one another. Three movement models are considered, each using a probability p to determine the direction of movement. The optimal value p is found in each case. Agmon et al. (2008b) considers the case where the adversary has no knowledge of the patroller's policy and shows that a deterministic policy is optimal in this case. This algorithm patrols only a subset of the perimeter segments, meaning that if the adversary were to gain any knowledge of the patroller's policy then they would be able to exploit this information to ensure all attacks target undefended segments. They also consider what happens when the adversary has partial knowledge of the patroller's policy and construct an algorithm for this case.

Basilico et al. (2009) and Basilico et al. (2012) consider a single robot patrolling a graph in discrete time. At each time step the patroller can move to an adjacent location.

Portugal and Rocha (2013) consider multiple robots patrolling a map. Special consideration is given to practical aspects such as scalability as the number of robots increases, recharging robots and decentralised decision-making which makes it harder for an adversary to discover the patrol pattern ahead of time.

2.2.3 Patrolling Games

A patrolling game involves two decision makers: a patroller (or a defender that coordinates multiple patrollers) and an attacker (or an adversary that coordinates multiple attackers). These decision-makers are usually in direct opposition to each other, with the attacker wanting to maximise the probability of attacks being successful and the patroller wanting to minimise this probability. The game is often played on a graph. In discrete patrolling games attacks can only take place at the nodes of the graph, whereas in continuous patrolling games, attacks can also occur at the edges. Games can also take place in discrete time, where the patroller can move between adjacent locations at each time step, or in continuous time, where the patroller traverses the graph by travelling along its edges.

Discrete Patrolling Games Motivated by recent developments in search games, Alpern et al. (2011) propose a discrete patrolling game. This patrolling game is set on a graph, where the n nodes represent the locations that can be attacked and the edges represent routes between them. The game takes place in discrete time. An attack at any location takes m time steps to complete, for some known $m \in \mathbb{N}$. At each time step, the patroller can move to and search one location that is connected by an edge to their current location. The attacker wins if the patroller does not arrive at the location under attack during the m time steps; otherwise, the patroller wins. Alpern et al. (2011) consider two versions of this game. The first version is called a one-off game and takes place over T time periods. Since the attack must complete before the game finishes, the attacker must choose a start time in the first $T-m$ time steps. The second version is called a periodic game and is played on a time circle $\{1, \ldots, T\}$. For example, if $T = 24$, these time steps could represent the 24 hours in the day. In a periodic game, attacks can start at any time but the patroller is constrained to choose from patrol cycles of length T, where an edge exists between the first and last location. Bounds are derived for the probability of the patroller winning each version of the game as well as some strategies for both players. Additional results are derived for patrol games played on Hamiltonian, bipartite and line graphs.

Lin et al. (2013) considers a generalised version of the periodic game presented by Alpern et al. (2011). Rather than an attack taking a fixed amount of time m, the time taken to attack each location is instead distributed randomly according to a known arbitrary distribution that can vary between locations. If the attacker remains in this location for the full duration without a patroller's interruption, then the attack is successful and the patroller incurs a cost that can depend on the location. As in Alpern et al. (2011), if the patroller detects the attacker before the attack is completed, then the cost is zero. The patroller is also no longer constrained to only consider patrol cycles of length T. Rather than choosing when they will attack, the attacker arrives at their location of choice after some random amount of time. The patroller's objective is to minimise the expected cost of attack. Lin et al. (2013) also considers the case where the location that is attacked is determined by a known probability distribution rather than a second decision-maker. In both cases, the authors present an exact linear program to compute the optimal patrol pattern. This algorithm scales very poorly with the number of locations due to the curse of dimensionality, so a heuristic index policy is also given. Lin et al. (2014) generalises this model further to allow for imperfect detection. Both the game theoretic and random attacker cases are considered and an index heuristic is derived for each case.

McGrath and Lin (2017) considers a different extension of Lin et al. (2013) in which there is a deterministic travel time between each pair of locations. The authors show that when there is only one patroller an optimal patrol pattern can be found using linear programming. McGrath and Lin (2017) also consider the case of multiple patrollers. In this case, finding an optimal policy is intractable so heuristic approaches which involve partitioning the locations into smaller patrol regions are considered.

Papadaki et al. (2016) expands on the work of Alpern et al. (2011) by finding an optimal policy for both the attacker and patroller in a one-off game when the game is played on a line graph. Special attention is given to this scenario due to its potential application to patrolling a border. Similarly, Alpern et al. (2019) focuses on line graphs but considers the periodic version of the problem. The authors find an optimal policy for both decision-makers in the special case where m, the number of time periods that the attacker needs to be in a location to complete an attack, is two. Alpern et al. (2022b) considers a patrolling game in which the patroller wears a uniform, so the attacker can decide when to start the attack based on when the patroller last visited.

Continuous Patrolling Games Alpern et al. (2016) extend the model of Alpern et al. (2011) by allowing the attacker to attack points along the edges of the graph. Rather than having discrete time steps the game takes place in continuous time with the patroller moving at a constant speed. The authors derive optimal strategies for both the attacker and patroller when the graph is a line. Garrec (2019) expands on the work of Alpern et al. (2016) and derives optimal solutions for Eulerian graphs and for a graph with two locations connected by three lines of the same length. Alpern et al. (2022a) derive three further results for the continuous patrolling game in specific cases. The authors also develop a patrol policy that they conjecture will be optimal for all games played on a tree graph. Bui and Lidbetter (2023) prove that this conjecture is true.

Lin (2022) considers a patrol game where a defender is required to protect a perimeter. The perimeter is continuous and any point along it could be subject to attack. At some fixed point along the perimeter is a base. The defender can dispatch patrollers—who wear uniforms—that depart from the base and each patroller travels around the perimeter once in a chosen direction at a chosen speed. The defender can choose when to send out patrollers as long as the long-run average rate that patrollers leave the base is equal to a given value. The attacker can choose to attack any point along the perimeter at any time. Since the patrollers wear uniforms, the attacker can observe and learn about the patrol pattern before deciding when to start the attack. Each patroller passing by the location under attack independently detects the attack with probability p. Each time a patroller passes the attacker while the attack is taking place, there is a probability p that the attacker will be discovered. The problem is modelled as a sequential game over an infinite time horizon where the defender chooses their policy first. The attacker can then take the defender's patrol policy into account when devising their attack policy. Lin (2022) proves that the defender can always find an optimal policy where all patrollers are sent out from the base in the same direction and at the same speed.

This concludes our review of the relevant literature on search and patrol problems.

Chapter 3

Searching Dispersed Locations

3.1 Introduction

Governments, military services, and other organizations are often required to spend vast amounts of energy and resources on expensive search operations. The search for the missing Malaysia Airlines Flight 370 is believed to have cost more than 100 million US dollars prior to its suspension in 2017 (New Straits Times, 2017). In many situations, minimizing search times not only saves resources but also optimizes other outcomes. For example, in disaster relief contexts, a person's survival may depend on how quickly they are located. Moreover, a searcher traversing a dangerous environment is at greater risk of personal injury the longer the search lasts (Lidbetter, 2020). In military and anti-terrorism contexts, detecting hostile assets quickly reduces the risk of a successful attack. Scenarios such as these are relevant to the topic of *search theory*, an area of operations research dating back to the Second World War when the US Navy studied how to detect German U-boats (Koopman, 1946). Since then, search theory has been applied in many other settings, leading to the development of a rich body of literature; see Stone (2004) and Washburn (2014) for comprehensive reviews.

In this paper, we consider a discrete search problem in which a stationary object is hidden in one of n geographically distinct locations. Initially, a prior distribution $(p_1, \ldots, p_n)$ is available to the searcher, where p_i is the probability that the object is hidden in location i, for $i = 1, \ldots, n$, with $\sum_{i=1}^{n} p_i = 1$. In order to locate the object, the searcher carries out a sequence of searches. Each search of location i requires t_i units of time, $i = 1, \ldots, n$. Additionally, if the object is in location i then the object is discovered with probability q_i, regardless of the number of times that location i has previously been searched, for $i = 1, \ldots, n$. We refer to q_i as the *detection probability* for location i, for $i = 1, \ldots, n$. After carrying out an unsuccessful search, the searcher updates her belief about the object's location using Bayes' theorem and then chooses whether to carry out another search at the same location or move to a new location. Our formulation differs from much of the previous work in this area by including *travel times* to move between different locations. Specifically, we assume that it takes d_{ij} units of time to move from location i to location j, for $i, j \in \{1, \ldots, n\}$. The searcher's objective is to find a search sequence that minimizes the expected amount of time until the object is discovered.

If the travel times d_{ij} are all set to zero, then our problem reduces to one that has been studied extensively in the literature and is equivalent to a particular multi-armed bandit problem. This connection was first established by Kelly in his comment in Gittins (1979). Expanding upon this observation, Gittins (1989) gives a full account of this problem. The optimal search policy states that at each epoch the best option is to search the location that maximizes the index $G_i(s_i)$, where

$$G_i(s_i) = \frac{p_i q_i (1 - q_i)^{s_i}}{t_i}, \qquad i = 1, \ldots, n, \tag{3.1}$$

and s_i is the number of times that location i has previously been searched. A full derivation of these indices and proof of optimality can be found in Gittins et al. (2011).

Several extensions of the problem without travel times are addressed in the literature. Chew et al. (1967) include the possibility that the object is not in any of the n locations. In this case, a cost is incurred for ending the search without discovering the object and the authors derive an optimal stopping condition. Kress et al. (2008) consider the case of imperfect specificity, where searches can potentially yield false positives. After a detection is made, a lengthier verification process must be carried out before the search is completed. Subelman (1981) and Lin and Singham (2016) consider the objective to maximize the probability of finding the object before a deadline. Clarkson et al. (2020) allow each location to be searched using either a *fast mode* or a *slow mode*, where the fast mode implies a shorter search time but also a smaller probability of discovering the object.

Alpern et al. (2009) consider search games on a network with perfect detection. They show that if the network is symmetric, than the hider should hide uniformly at random. They then introduce a variant of the Chinese Postman Problem called the Utilitarian Postman Problem (UPP) where the objective is to minimise the mean time to deliver all mail. They demonstrate that the optimal solution to this problem also provides an optimal search policy for their search game. Jotshi and Batta (2008) also use the UPP to address search problems where the searcher is assumed to be hiding uniformly at random on an arbitrary network. Alpern and Lidbetter (2014) extend the study of network based search games to include a notion of variable speed.

Several other recent developments consider game theoretic formulations. Dagan and Gal (2008) formulate a zero-sum game between a blind unit-speed searcher and a stationary hider. Hohzaki (2016) reviews the literature on search games, where an adversary chooses where to hide an object in order to make it difficult for the searcher to find it. Clarkson et al. (2023), Clarkson and Lin (2024), and Bui et al. (2024) consider a game-theoretic version of the special case of our problem with all travel times set to zero, while Lidbetter (2013) and Lidbetter and Lin (2019) consider variations where the searcher needs to find multiple hidden objects. Baston and Kikuta (2013) apply game theory to another special case of our problem with perfect detection. McGrath and Lin (2017) study a patrolling game that takes into account travel time between locations. Although some of the aforementioned game theoretic models take into account geographical aspects by using a network formulation and others allow for imperfect detection, to the best of our knowledge there are no previous studies that simultaneously consider a network formulation, an infinite time horizon, and imperfect detection.

The inclusion of travel times in our problem makes it much more challenging. Without travel times, the additional cost (in terms of time required) of the next search has no dependence on the location of any previous searches. When travel times are included, however, the cost of the next search includes both the time required to travel to a new location (if applicable) and the time to carry out the search itself. In addition, when considering a particular location as a candidate to be searched next, the searcher also needs to take into account not only the probability of discovering the object there, but also its position relative to other locations that might be chosen for future searches. The Gittins' index policy, based on the indices defined in (3.1), is myopic in nature and lacks the ability to plan ahead. For these reasons, our problem demands a non-myopic approach even in the special case where all travel times are equal and non-zero (that is, $d_{ij} = d > 0$ for $i, j \in \{1, \dots, n\}$) since the current location can always be reached without traveling.

The principal contributions of this paper including the following:

1. We show that the Gittins' index policy for the version of the problem with no travel times also yields useful insight into the version with travel times, as it only instructs the searcher to remain at her current location when it is optimal to do so.

2. We draw upon the theory of restless bandit problems to derive an index-based heuristic policy that explicitly takes into account the travel times between different locations.

3. We derive the form of the optimal policy in the special case of two locations with identical parameters and show that the index heuristic has a similar structure to the optimal policy in this case, but with the index heuristic the searcher tends to leave the current location prematurely.

4. We show that, in the simple two-location problem described above, the new index heuristic only instructs the searcher to remain at her current location when it is optimal to do so. This property also holds in more general n-location problems with heterogeneous parameters provided that the travel times between all pairs of locations are the same.

5. We introduce two additional heuristics in order to address the shortcomings of the new index heuristic mentioned above, and show how the performance of a search

sequence can be further improved by performing extra steps based on insertion heuristics and policy improvement.

6. We provide extensive numerical results to evaluate the performances of our heuristics with respect to the approximate optimal values and demonstrate the benefits of the improved methods using insightful comparisons.

The remainder of the paper is organized as follows. In Section 3.2, we formulate the search problem as a semi-Markov decision process, explain why the problem is difficult to solve, and use finite-stage dynamic programming to approximate optimal solutions. In Section 3.3, we develop an index heuristic based on restless bandit theory and show that it has a tendency to move away from the searcher's current location prematurely. In Section 3.4, we use these insights to develop a range of heuristic approaches that build upon the simple index policy in order to achieve a stronger performance. In Section 3.5, we carry out an extensive numerical study on a set of randomly-generated problem instances with 4 locations, and also another set with 8 locations. Section 3.6 concludes the paper and suggests directions for future work.

3.2 Model Formulation and Preliminaries

This section formulates the problem of searching for an object in several dispersed locations and develops some preliminary results. Section 3.2.1 formulates the search problem as a semi-Markov decision process (SMDP). Section 3.2.2 discusses why the SMDP is difficult to solve and offers some properties of an optimal policy. Section 3.2.3 explains how to approximate the optimal value via a finite-stage dynamic program.

3.2.1 Formulation as a Semi-Markov Decision Process

Consider a SMDP with the following features:

1. A stationary object is hidden in one of n distinct locations, labelled $1, \dots, n$, for some $n \in \mathbb{N}$. For ease of notation, we write $[n] = \{1, \dots, n\}$. For each $i \in [n]$, the

probability that the object is in location i is given by $p_i \in (0,1)$. Together these probabilities form a prior distribution P satisfying $\sum_{i=1}^{n} p_i = 1$.

2. The searcher is initially at location 1. Decision epochs occur at the start of the search procedure and after each unsuccessful search. At each decision epoch, one action is chosen from the action space $[n]$ where, for each $i \in [n]$, action i denotes moving to location i (if not already there) and searching location i once.

3. For each $i \in [n]$, a search in location i takes t_i units of time and is successful with probability q_i if the object is there. For each distinct pair $i, j \in [n]$ it takes d_{ij} units of time to travel from location i to location j. For completeness we define $d_{ii} \equiv 0$ for all $i \in [n]$.

4. The objective is to minimize the expected amount of time needed to discover the object, which we refer to as the *expected time to discovery* and abbreviate as ETD.

After each unsuccessful search, the probability distribution P is updated to P' according to the Bayes' Theorem. Thus, the state of the SMDP can be written as $(i, (p'_1, \ldots, p'_n)) \in [n] \times (0,1)^n$, where i is the searcher's current location and, for each $j \in [n]$, p'_j is the present (posterior) probability that the object is in location j. The initial state is $(1, (p_1, \ldots, p_n))$. If action j is chosen at the initial epoch and the search is unsuccessful then the state updates to $(j, (p'_1, \ldots, p'_n))$ where $p'_j = (1-q_j)p_j/(1-p_jq_j)$ and $p'_i = p_i/(1-p_jq_j)$ for each $i \neq j$.

In the general context of SMDPs, a decision-making policy should associate a decision rule with any given state, where the decision rule is a (possibly randomized) method of selecting an action. In our problem, the application of a policy to an initial state generates a search sequence $(i_0, i_1, \ldots)$, where $i_m \in [n]$ is the location to be searched at decision epoch $m \geq 0$ under the assumption that previous searches have been unsuccessful. We assume the initial state to be known, so to carry out a search plan it is sufficient to find an optimal *search sequence* rather than a full policy. Moreover, standard theory (Puterman, 1994) implies that an optimal procedure for our problem can be represented in such a form.

3.2.2 On the Optimal Policy

If $d_{ij} = 0$ for all $i, j \in [n]$ then it is well established that, given state $(i, (p'_1, \ldots, p'_n))$, it is optimal for the next search to take place at the location $j \in [n]$ that maximizes the ratio $p'_j q_j / t_j$, regardless of the current location i (Gittins, 1979). This ratio $p'_j q_j / t_j$ is called the index of location j, for $j \in [n]$, and indicates the relative attractiveness of choosing location j as the location of the next search. The policy that always searches the location with the highest index at any given moment is referred to as the Gittins' index policy.

For a search problem with non-zero travel times, the Gittins' index policy is not guaranteed to be optimal. To illustrate this point, consider a symmetric two-location search problem with parameters $q_1 = q_2 = q$, $t_1 = t_2 = t$ and $d_{12} = d_{21} = d$. If q and t are small and d is large then it is intuitive that an optimal policy will tend to carry out multiple consecutive searches at the same location in order to reduce the amount of time spent traveling. However, the Gittins' index policy would frequently switch between the two locations, regardless of the travel time required. In the more general context of n-location problems, the travel time required to move to the location with the largest index may motivate the searcher to visit a closer location first, or to carry out additional searches at the current location. It is also intuitive that a cluster of locations situated in proximity to each other should be more attractive to visit than an isolated location. Thus, an optimal policy should take into account the full matrix of travel times when making a decision.

Despite being suboptimal for our problem, the Gittins' index policy still offers some useful insights. Since it ignores travel times, it will naturally tend to overestimate the attractiveness of moving to a different location. Thus, if it recommends that the searcher should remain at the current location at a particular point in the search, then it is intuitive that it is an optimal action. Our first result formalizes this idea.

Theorem 3.1. *Consider the SMDP introduced in Section 3.2.1. Write $(i, (p'_1, \ldots, p'_n))$ for the current state. If for each $j \neq i$ we have*

$$\frac{p'_i q_i}{t_i} > \frac{p'_j q_j}{t_j},$$

then it is optimal to carry out the next search at location i.

Here we include an outline of the proof; further details are provided in Section 3.7.1. Suppose that the prerequisite conditions hold and consider an arbitrary policy. Let π be the search sequence generated when this policy is applied to the current state and suppose that the first location in this sequence is not i. Now let $\hat{\pi}$ be a modified version of this sequence that is constructed by identifying the first epoch at which location i is searched in π and bringing this search forward in the sequence so that it takes place immediately. We can show that $\hat{\pi}$ is superior to π by comparing the two sequences with respect to an equivalent criterion based on discounted rewards. The comparison is made by making a series of *maneuvers*, starting from the original sequence π and concluding with the modified sequence $\hat{\pi}$. We carry out the maneuvers in such a way that the search sequences obtained at intermediate stages are not admissible procedures themselves, since they sometimes ignore travel times, but the starting and ending procedures correspond to π and $\hat{\pi}$ which are both admissible. It can be shown that each maneuver results in a search procedure with a better performance than the previous one, and therefore $\hat{\pi}$ must be superior to π. Please refer to Section 3.7.1 for full details.

According to Theorem 3.1, if the Gittins' index of the searcher's current location is the largest, then it is optimal to search the current location again. If that is not the case, however, then it is not clear what the optimal action is. In theory, it is ideal to have a complete description of an optimal policy—mapping each state in the SMDP to an optimal action—so that the searcher can apply an optimal policy to an initial state to generate an optimal *search sequence*. In practice, however, it is sometimes possible to obtain an optimal search sequence without a complete description of an optimal policy, because not all states will be encountered when the searcher follows an optimal policy. We next present an example in which an optimal search sequence can be explicitly determined without a complete characterization of an optimal policy.

Consider a search problem with $n = 2$ locations with $p_1 = p_2 = 0.5$, $q_1 = q_2 = q$, $t_1 = t_2 = t$ and $d_{12} = d_{21} = d > 0$. We can exploit the symmetry of this problem to determine an optimal search sequence. Given that the searcher begins in location 1 and both locations have the same parameter values, it is clear that moving to location 2 immediately is suboptimal, as it unnecessarily adds to the total travel time. The searcher should therefore begin by searching the location 1 at least once. She must then decide whether to search location 1 again or move to the second location.

Suppose that it is optimal to search location 1 for m times before traveling to location 2, where $m \geq 1$. If the object has not been found after these m searches, then the posterior probability of the object being in location 2 is greater than 0.5. If the searcher then switches to location 2 and carries out m unsuccessful searches there, then the posterior probabilities for both locations are once again both equal to 0.5. At this point we are in the same situation that we started in, except that the searcher has switched from location 1 to location 2. It follows that location 2 should be searched a further m times before the searcher moves back to location 1. Thus, an optimal search sequence must instruct the searcher to search location 1 for m times for some $m \in \mathbb{N}$ and then alternate between the two locations, searching $2m$ times on each visit.

Write T for the time needed for the searcher to find the object with the preceding search sequence, which starts with m searches in location 1. Compute

$$\begin{aligned}
\mathbb{E}[T] &= \frac{1}{2}\sum_{k=1}^{m} q(1-q)^{k-1}(kt) + \frac{1}{2}\sum_{k=1}^{m} q(1-q)^{k-1}(mt+d+kt) + (2mt+d+\mathbb{E}[T])(1-q)^m \\
&= \left(\frac{t}{q} + \frac{3mt+d}{2}\right)(1-(1-q)^m) - mt + (2mt+d+\mathbb{E}[T])(1-q)^m,
\end{aligned}$$

because with probability $(1-q)^m$, the searcher does not find the object in the first m searches at location 1 or the first m searches at location 2, in which case the state of the search becomes the same as that at the very beginning of the search. Solving for $\mathbb{E}[T]$ gives

$$\mathbb{E}[T] = \frac{t}{q} + \frac{3mt+d}{2} + \frac{(2mt+d)(1-q)^m - mt}{1-(1-q)^m}.$$

which implies

$$
\begin{aligned}
m^* &= \operatorname*{argmin}_{m\in\mathbb{N}} \left(\frac{3mt+d}{2} + \frac{(2mt+d)(1-q)^m - mt}{1-(1-q)^m} \right) \\
&= \operatorname*{argmin}_{m\in\mathbb{N}} \left((mt+d)\frac{1+(1-q)^m}{1-(1-q)^m} \right) \\
&= \operatorname*{argmin}_{m\in\mathbb{N}} f(m). \qquad (3.2)
\end{aligned}
$$

It can be shown using differentiation that the function $f(m)$ has a unique minimum in $\mathbb{R}$ and so there exists an optimal search sequence of the form described earlier. This solution will usually be unique; however, since m is restricted to the natural numbers, there exist some cases where two consecutive values of m both correspond to optimal policies. In these cases, we use the convention that m^* is defined as the larger of the two values.

3.2.3 Value Approximation via Finite-stage Dynamic Programming

In this subsection, we introduce a finite-stage dynamic programming (DP) approach to approximate the optimal ETD for a given initial state of the SMDP in Section 3.2.1. Consider a variant of our search problem in which, for each location $i \in [n]$, there is a maximum number of searches b_i that the searcher is allowed to carry out at that location. If the searcher has not found the object after carrying out the maximum number of searches at all n locations, then the object is revealed and the search immediately comes to an end. By making the b_i values sufficiently large, we can use the optimal time to discovery in this problem variant as an approximation for the optimal ETD in the original problem.

Our DP approach is based on bounding the state space by choosing suitable values for the maxima b_i. To determine these values, we first choose a small probability $\varepsilon > 0$ and then, for each location $i \in [n]$, compute the minimum value of b_i such that the probability that the object is in location i and the searcher does not find it after b_i searches in location i is smaller than ε. Thus, we set $b_i = \lceil \log_{(1-q_i)}(\varepsilon/p_i) \rceil$ for $i \in [n]$.

Suppose that we want to approximate the optimal ETD for a given initial state $(1, (p_1, \dots, p_n))$. At an arbitrary decision epoch, write $H = (i, (s_1, \dots, s_n))$ as the *search history*, where i is the searcher's current location and s_j is the number of times that location j has been searched so far, for $j \in [n]$. Given the initial state $(1, (p_1, \dots, p_n))$ and the search history $(i, (s_1, \dots, s_n))$, we can calculate the current state as $(i, (p'_1, \dots, p'_n))$ where, for each $j \in [n]$,

$$p'_j = \frac{p_j(1 - q_j)^{s_j}}{\sum_{k=1}^n p_k(1 - q_k)^{s_k}}. \tag{3.3}$$

If location j is searched next, then the search history is updated to

$$H^j = (j, (s'_1, \dots, s'_n)) \tag{3.4}$$

where $s'_j = s_j + 1$ and $s'_i = s_i$ for $i \neq j$.

Given an initial state $(1, (p_1, \dots, p_n))$ and a search history $H = (i, (s_1, \dots, s_n))$, the value function $V(H)$ gives the expected remaining search time if all subsequent actions are optimal:

$$V(H) = \min_{j \in [n]} \left\{ d_{ij} + t_j + (1 - p'_j q_j) V(H^j) \right\}, \tag{3.5}$$

where p'_j and H^j are defined by (3.3) and (3.4) respectively.

We begin by setting $V(i, (b_1, \dots, b_n)) = 0$ for each $i \in [n]$, which is the point when the object is revealed free of charge. We then iterate backwards via repeated application of (3.5) until we are able to derive the value of $V(1, (0, \dots 0))$, which corresponds to the optimal ETD in our variant of the problem. Computationally, we carry out the following steps:

1. Initialize M as an empty $(n+1)$-dimensional matrix of size $n \times b_1 \times \dots \times b_n$. For each $i \in [n]$, set $M(i, b_1, \dots, b_n) = 0$. Also set $k = \sum_{i \in [n]} b_i$.

2. (Backwards iteration) Consider all elements $(i, s_1, \dots, s_n)$ of the matrix M such that $\sum_{j \in [n]} s_j = k - 1$. For each of these elements, initialize an empty vector W of length n. Next, For each $j \in [n]$, set

$$W_j = \begin{cases} d_{ij} + t_j + (1 - p'_j q_j) M(j, s_i^{(j)}, \dots, s_n^{(j)}), & s_j < b_j, \\ \infty, & s_j = b_j, \end{cases}$$

where $s_k^{(j)} = s_k + \mathbb{I}_{k=j}$ for $k \in [n]$. Then set $M(i, s_1, \dots, s_n) = \min_j W_j$.

3. If $k = 1$, stop and output $M(1, 0, 0, \ldots, 0)$ as the optimal ETD. Otherwise, reduce k by 1 and return to step 2.

This approach produces an approximation to the optimal value of the original search problem. We also note that, having populated the matrix M with approximations for all of the $V(H)$ values, we can quickly generate an approximation for the optimal search sequence (in a truncated form) by starting at $(1, (0, 0, \ldots, 0))$, moving forwards in time and identifying the minimizing action in (3.5) at each decision epoch. However, this method still suffers from the curse of dimensionality as the number of locations increases, so other approaches are needed for larger problems.

3.3 A Heuristic Method Based on Indices

In this section, we develop an index heuristic for the search problem formulated in Section 3.2.1. Since the Gittins' index policy is optimal for the version of our search problem with no travel time, it is natural to explore an index policy when travel time is present. The class of restless bandits problems (RBPs), first introduced by Whittle (1988), is a generalization of the multi-armed bandit problem that allows bandits that are not active to change state and earn rewards according to a particular set of rules. As with most classes of multi-armed bandit problems, if the problem is of a reasonable size then finding an optimal policy directly is computationally difficult. However, Whittle (1988) developed an index heuristic using Lagrangian relaxation which has been shown empirically to find near-optimal solutions in many RBPs.

3.3.1 Development of an Index Heuristic

Consider the search problem as described in Section 3.2.1. First, consider the problem of choosing a search policy π to maximize the expected discounted reward $\mathbb{E}_\pi[\beta^T]$ where $\beta \in (0, 1)$ and T is the total search time. Under any deterministic, Markov search policy π, write $\tau_\pi(i, k)$ for the time until completion of the k^{th} search of location i under policy

π. We can then write

$$\mathbb{E}_\pi[\beta^T] = \sum_{i=1}^{n}\sum_{k=1}^{\infty} p_i q_i (1-q_i)^{k-1}\beta^{\tau_\pi(i,k)}. \tag{3.6}$$

Now focus on the contribution to the objective $\mathbb{E}_\pi[\beta^T]$ accruing from visits to one particular location. We drop the location identifier in this discussion and use location parameters p, q and t for ease of presentation. While the searcher is away from this location, no contribution is made to the objective function in (3.6), so we say that the location is under the *passive* action, denoted b. If the location is passive and has been searched k times to date, we say that it is in state (b, k). After a period of passivity, when the searcher chooses to search the location, we say the location is *active*, denoted a. However, before search can resume, there will be some travel time, denoted by σ, to expend. In this account we will treat σ as a positive-valued random variable independent of all else and for which $\mathbb{E}[\beta^\sigma] < \infty$. If the location is active and has been searched k times to date, we say that it is in state (a, k). Based on this description, we develop a semi-Markov decision process model of a single location which takes the form of a restless bandit. We call this model the search location restless bandit (SLRB). The key elements of the SLRB model are as follows:

1. The state space is $\Omega = \cup_{k=0}^{\infty}\{(a, k), (b, k)\}$.

2. Actions $\{a, b\} \equiv \{\text{active, passive}\}$ are available in each state.

3. State transitions are as follows:

    (a) Under action $a : (a, k) \to (a, k+1)$ and $(b, k) \to (a, k+1)$. These transitions take times t and $t + \sigma$ respectively. They both earn a (discounted) return of $p(1-q)^k q$, to be received once the transition is complete.

    (b) Under action $b : (a, k) \to (b, k)$ and $(b, k) \to (b, k)$. These transitions both take one unit of time to complete. Both transitions earn a (discounted) return of W (referred to as a *subsidy for passivity* in Whittle (1988)) once the transition is complete. Here W may be assumed to be positive. The discount rate is $\beta \in (0, 1)$.

4. Decision epochs occur at time zero and at the conclusions of all transitions.

5. The goal is to choose actions to maximize the expected discounted reward earned over an infinite horizon. We use $V(\omega, W)$ for the value (optimal expected discounted reward) when in state $\omega \in \Omega$ at time zero. Bellman's optimality equations are as follows for all $k \in \mathbb{N}$:

$$V((a,k),W) = \max\left\{\beta^t p(1-q)^k q + \beta^t V((a,k+1),W); \beta W + \beta V((b,k),W)\right\}, \tag{3.7}$$

and

$$V((b,k),W) = \max\left\{\beta^t \mathbb{E}[\beta^\sigma] p(1-q)^k q + \beta^t \mathbb{E}[\beta^\sigma] V((a,k+1),W); \frac{\beta W}{1-\beta}\right\}. \tag{3.8}$$

In the preceding equations, the active action is optimal whenever the maximum is achieved by the first term and the passive action is optimal whenever the maximum is achieved by the second term. The simple form of the second term in the second equation arises from that fact that if the passive action is optimal in (b,k) then it will continue to be so.

Whittle's indices are only defined for bandits that meet an indexability condition. This condition is only met if, as the level of passive subsidy increases, so does the collection of states for which the passive action is optimal. We now write $\Pi_\beta(W)$ for the set of states for which the passive action is optimal. Thus, if $\Pi_\beta(W)$ is non-decreasing in W for all β then we say that the restless bandit is *indexable* with a *Whittle Index* $W(\omega,\beta) : \Omega \to \mathbb{R}$ given by $W(\omega,\beta) = \inf\{W; \omega \in \Pi_\beta(W)\}$. For this restless bandit problem, indexability can be verified by examining the value equations. In particular, if for some $W^* \in \mathbb{R}^+$, the maximum is achieved by the second term (which represents the passive action) then for any $W > W^*$ the maximum will also be achieved by the second term. Moreover, the Whittle index is the smallest value of W for which the two terms are equal.

Suitable development of the analysis in Glazebrook et al. (2006) yields the following conclusion:

Proposition 3.2. *The SLRB model is indexable, with Whittle indices given by*

$$W((a,k),\beta) = (1-\beta)\frac{\beta^t p(1-q)^k q}{(1-\beta^t)}, \quad k \in \mathbb{N},$$

and

$$W((b,k),\beta) = (1-\beta)\beta^t \mathbb{E}[\beta^\sigma] p(1-q)^k q \max_{r\geq 1} \frac{\sum_{s=0}^{r-1} (1-q)^s \beta^{st}}{1-\beta^{rt} E\left(\beta^\sigma\right)}, \quad k \in \mathbb{N}.$$

With these indices, an optimal policy for the SLRB with discount rate β is as follows: In state ω, choose the passive action whenever $W \geq W(\omega,\beta)$ and choose the active action otherwise. This policy provides a natural calibration for the search locations, both when being searched (and hence in some state (a,k)) and when not being searched (and hence in some state (b,k)). However, our search model is not discounted and so we are especially interested in the above indices in the limit $\beta \to 1$, which we develop as follows:

$$W(a,k) = \lim_{\beta\to 1} W((a,k),\beta) = \frac{p(1-q)^k q}{t}, \quad k \in \mathbb{N}$$

and

$$\begin{aligned} W(b,k) = \lim_{\beta\to 1} W((b,k),\beta) &= p(1-q)^k q \max_{r\in\mathbb{N}} \frac{\sum_{s=0}^{r-1}(1-q)^s}{\mathbb{E}[\sigma]+rt} \\ &= p(1-q)^k \max_{r\in\mathbb{N}} \frac{1-(1-q)^r}{\mathbb{E}[\sigma]+rt}, \quad k \in \mathbb{N}. \end{aligned}$$

To understand how the index heuristic works, consider the SMDP formulated in Section 3.2.1. Suppose the current location is i and, for each $j \in [n]$, location j has previously been searched s_j times. Our analysis suggests the index

$$W_{ii}(s_i) = \frac{p_i q_i (1-q_i)^{s_i}}{t_i} \tag{3.9}$$

for the current location i and the index

$$W_{ij}(s_j) = p_j(1-q_j)^{s_j} \sup_{k\in\mathbb{N}} \left\{ \frac{1-(1-q_j)^k}{d_{ij}+kt_j} \right\} \tag{3.10}$$

for location $j \neq i$.

At each epoch, the corresponding index heuristic mandates that if (3.9) is greater than or equal to (3.10) for all other locations then the searcher should carry out one additional search at her current location. Otherwise, if (3.10) is larger (3.9) for at least one location then the searcher should move to and search the location having the largest index. If more than one of these locations have indices with the same largest value then the searcher should choose one from these locations arbitrarily.

In the remainder of this paper, any mention of the index heuristic refers to the heuristic defined by (3.9)–(3.10) and not the Gittins' index policy discussed in Section 3.1.

3.3.2 Properties of the Index Heuristic

To gain insight into the index heuristic proposed in Section 3.3.1, we now compare it with the optimal search policy in two special cases.

First, consider the special case in which all travel times are zero. In this case, the indices (3.9)–(3.10) reduce to the Gittins' indices in (3.1), which are known to be optimal in this special case.

Second, consider the special case discussed in Section 3.2.2, in which the search problem consists of $n = 2$ locations having identical parameters, namely, $p_1 = p_2 = 0.5$, $q_1 = q_2 = q$, $t_1 = t_2 = t$ and $d_{12} = d_{21} = d > 0$. Direct inspection of (3.9) and (3.10) makes it clear that the index of the searcher's current location is initially larger, so the index heuristic mandates that the search should start there. However, the form of (3.9) is such that this index decreases after each unsuccessful search, while the index of the other location remains unchanged. Therefore, there exists a minimum value $\hat{m} \in \mathbb{N}$ such that the index of the initial location falls below that of the other location after $\hat{m}$ unsuccessful searches have taken place. Inspection of (3.9) and (3.10) yields that

$$\hat{m} = \min \left\{ m \geq 1 \ : \ \frac{q(1-q)^m}{t} < \max_{k \in \mathbb{N}} \left[\frac{1-(1-q)^k}{d+kt} \right] \right\}.$$

Following $\hat{m}$ unsuccessful searches at the first location, the index heuristic mandates $\hat{m}$ searches at the new location. If these searches are also unsuccessful, then the indices of the two locations are in the same ratio that they were at time zero, but with the roles of the locations reversed. It then follows that a further $\hat{m}$ searches are planned for the new location (making $2\hat{m}$ in total since the first switch) before a second switch takes place and the searcher returns to the starting location. We conclude that the index heuristic has a similar structure to the optimal policy, but with the parameter m set to $\hat{m}$ rather than the optimal value m^* defined in (3.2).

We next compare $\hat{m}$ with m^*. The index heuristic explicitly takes into account the travel time to the other location, but does not account for the time spent traveling back to the current location in the future. It is intuitive that the index heuristic tends to underestimate the long-term cost of switching to the other location, which suggests that the index heuristic may advise the searcher to leave the current location before it is optimal to do so. Our next result confirms this intuition by showing that $\hat{m} \le m^*$.

Proposition 3.3. *In the two-location symmetric problem, $\hat{m} \le m^*$.*

Proof. Set $t = 1$ without loss of generality. From (3.9) and (3.10) it follows that $\hat{m}$ is the smallest $m \in \mathbb{N}$ such that

$$q(1-q)^m < \max_{k \in \mathbb{N}} \frac{1-(1-q)^k}{d+k}.$$

It is sufficient to show that this inequality holds when $m = k = m^*$, or equivalently that

$$d + m^* \le \frac{1-(1-q)^{m^*}}{q(1-q)^{m^*}}. \tag{3.11}$$

From the definition of m^* in (3.2) it follows that, for all $k \in \mathbb{N}$, we have

$$(d+m^*)\frac{1+(1-q)^{m^*}}{1-(1-q)^{m^*}} \le (d+k)\frac{1+(1-q)^k}{1-(1-q)^k}.$$

In particular, this inequality holds when $k = m^* + 1$. With this choice of k we obtain

$$d + m^* \le \frac{1-(1-q)^{m^*}(q+(1-q)^{m^*+1})}{2q(1-q)^{m^*}}. \tag{3.12}$$

Combining (3.11) and (3.12) implies that it is sufficient to show

$$\frac{1-(1-q)^{m^*}(q+(1-q)^{m^*+1})}{2q(1-q)^{m^*}} \le \frac{1-(1-q)^{m^*}}{q(1-q)^{m^*}}.$$

Rearranging this inequality yields:

$$(2-q)(1-q)^{m^*} - (1-q)^{2m^*+1} \le 1.$$

This inequality holds for $m^* = 1$ and, since the left-hand side is decreasing with m^*, we conclude that it holds for all m^*. □

It follows from Proposition 3.3 that, in the two-location symmetric problem, the only suboptimal actions given by the index heuristic are those that instruct the searcher to move away from her current location. It is natural to ask whether a similar principle holds in more general cases. Consider a problem with n locations in which all travel times are identical; that is, $d_{ij} = d > 0$ for all $i, j \in [n]$ with $i \neq j$. We refer to this as the *symmetric travel case*. We can show that in this case, it is always optimal to conduct an additional search of the current location if the index heuristic recommends this option. This result is a natural extension of Theorem 3.1, which established the same property for the Gittins' indices, albeit without assuming symmetric travel times.

Theorem 3.4. *Consider the symmetric travel case, where any two locations can be moved between in $d > 0$ units of time. Suppose that the searcher is at location i and, for each $j \in [n]$, location j has been searched s_j times. If for each $j \neq i$ we have*

$$W_{ii}(s_i) = \frac{p_i q_i (1-q_i)^{s_i}}{t_i} > W_{ij}(s_j) = p_j (1-q_j)^{s_j} \sup_k \left\{ \frac{1-(1-q_j)^k}{d + k t_j} \right\}$$

then it is optimal to search location i next.

The proof follows similar arguments to that of Theorem 3.1 and involves showing that, given any search sequence that begins by moving to a location $j \neq i$, we can perform a series of maneuvers in order to obtain a modified sequence that searches i immediately and yields a smaller value for the ETD. Please refer to Section 3.7.2 for full details.

3.4 Search Sequence Generation and Improvement

As discussed in Section 3.3, search sequences generated by the index heuristic defined in (3.9) and (3.10) need not be optimal. A plausible explanation is that these indices do not take into account the distances of candidate locations to other locations and the future travel times. This section offers a few methods to address the shortcomings of the index heuristic.

To evaluate the performance of a heuristic, apply it to an initial state to obtain a search sequence and let T denote the random amount of time needed to find the object with the search sequence. Compute the ETD by

$$\mathbb{E}[T] = \int_0^\infty \mathbb{P}(T > t)dt = \sum_{k=0}^{\infty}(\tau_{k+1} - \tau_k)\mathbb{P}(T > \tau_k), \tag{3.13}$$

where τ_k denotes the total time elapsed following completion of the k^{th} unsuccessful search, with $\tau_0 \equiv 0$. If the search sequence generated by the heuristic is truncated after l searches, then a lower bound for the ETD is given by

$$\mathbb{E}_{LB}[T] = \sum_{i=0}^{l}(\tau_{i+1} - \tau_i)\mathbb{P}(T > \tau_i).$$

This lower bound can be made arbitrarily tight by extending the point of truncation. Ideally, we would also like to have a tight upper bound for the ETD, but this is more difficult to obtain.

We next present a method to approximate the ETD for a given search sequence. Choose some small value $\varepsilon > 0$ and truncate the search sequence if s_i searches have been carried out at location i, $i \in [n]$, such that

$$\sum_{i\in[n]} p_i(1 - q_i)^{s_i} < \varepsilon. \tag{3.14}$$

In other words, the probability of not finding the object after exhausting the searches in the truncated search sequence is no more than ε. After truncating a search sequence, append the truncated search sequence with a Hamiltonian cycle and repeat the same Hamiltonian cycle indefinitely. The performance of this new sequence with repeated Hamiltonian cycles can be computed exactly because the tail sum in (3.13) conforms to a geometric pattern. For convenience, we simply use the ordered list of locations 1 to n as the Hamiltonian cycle. For example, if the last search before truncation was at location n and s_i is the number of searches at location $i \in [n]$ prior to truncation, then this method produces an approximation of the ETD as

$$\begin{aligned}\mathbb{E}_{PES}[T] &= \mathbb{E}_{LB}[T] + \sum_{r=0}^{\infty}\sum_{i=1}^{n}(d_{i-1,i} + t_i)\left(\sum_{a=1}^{i-1} p_a(1 - q_a)^{s_a+r+1} + \sum_{a=i}^{n} p_a(1 - q_a)^{s_a+r}\right)\\ &= \mathbb{E}_{LB}[T] + \sum_{i=1}^{n}(d_{i-1,i} + t_i)\left(\sum_{a=1}^{i-1} \frac{p_a(1 - q_a)^{s_a+1}}{q_a} + \sum_{a=i}^{n} \frac{p_a(1 - q_a)^{s_a}}{q_a}\right), \end{aligned} \tag{3.15}$$

where $d_{01} := d_{n1}$. The preceding approximation obtained with this method is somewhat *pessimistic* because a Hamiltonian cycle does not take into account any information that can be used to shorten the ETD.

Section 3.4.1 introduces two new heuristics that instruct the searcher to conduct one additional search at her current location if $W_{ii}(s_i) > W_{ij}(s_j)$ for all $j \neq i$ according to (3.9) and (3.10), and otherwise use alternative approaches to determine where to search next. Section 3.4.2 describes two methods to improve a given search sequence.

3.4.1 Other Index-based Search Heuristics

We now introduce two alternative approaches for constructing search sequences that use the indices given by (3.9) and (3.10). Both heuristics make use of the observation in Section 3.3.2 that when following the index heuristic the searcher tends to move away from her location too soon and so these new heuristics only deviate from the index heuristic when it instructs the searcher to move. The first heuristic creates a planned sequence using the index heuristic and uses it to construct a more detailed index corresponding to the option of moving away that explicitly takes into account all planned searches and travel times before the searcher returns to her current location. The second heuristic instead creates a mini-problem only using the current location and the m other locations with the highest indices and then uses a dynamic programming approach to approximate an optimal solution to that problem.

Round-Trip Index Heuristic

This heuristic is based on the idea that the index heuristic is too short-sighted in its approach for deciding whether or not the searcher should move away from her current location. In particular, the index heuristic fails to take into account the search times and travel times that would be involved in visiting a sequence of other locations before eventually returning to the current location. Therefore, we introduce a *round-trip* heuristic in which the indices for moving between locations are informed by longer-term considerations. This round-trip index heuristic forms a natural analogue of the restless bandit index in (3.10) when the nature of the absence from location i is the round trip mandated by the index heuristic rather than just the searches at one other location.

Firstly, we create a planned sequence using the index heuristic, by iterating over decision epochs and choosing the location of each new search according to the indices (3.9)–(3.10). The planned sequence is truncated at the point where the criterion (3.14) is satisfied and each location has been searched at least once thereafter.

A new, improved sequence is then created by carrying out the following steps:

1. Let the new sequence initially be empty. Begin at decision epoch $m = 0$.

2. Check the location to be searched at epoch m under the planned sequence. If this location matches the searcher's current location, then append this search to the new sequence and increase m by 1. Otherwise (if the next location in the planned sequence is different from the current location), let the index of the current location be defined by (3.9) and let $W(i \to i)$ denote a new index that indicates the value of moving to a new location and following the route proposed by the planned sequence until a return to i is mandated. The index $W(i \to i)$ is calculated by

$$W(i \to i) = \frac{\sum_{j=1}^{K} p_{\alpha_j}(1-q_{\alpha_j})^{s_{\alpha_j}} \left\{ \sum_{r=1}^{\nu_{\alpha_j}} q_{\alpha_j}(1-q_{\alpha_j})^{r-1} \right\}}{d(i \to i) + \sum_{j=1}^{K} \nu_{\alpha_j} t_{\alpha_j}}, \qquad (3.16)$$

   where $\alpha_1, \ldots, \alpha_K$ is an ordered list of the locations to be visited between successive visits to i (noting that some locations may be visited more than once over this period), ν_{α_j} is the total number of searches to be conducted in location α_j and $d(i \to i)$ is the total amount of time spent traveling between all locations visited (including the return trip back to i). If $W(i \to i)$ is larger than $W_{ii}(s_i)$ then append the next search in the planned sequence to the new sequence and increase m by 1. Otherwise, append an additional search in the current location to the new sequence. Note that an extra search in the current location implies that the index $W_{ii}(s_i)$ will change, but the index $W(i \to i)$ will not change.

3. If the stopping criterion (3.14) is satisfied for the new sequence, then stop. Otherwise, return to step 2.

The new sequence given by this procedure may be seen as an adjusted version of the index heuristic sequence in which each *block* of searches (i.e., each string of consecutive searches carried out at a single location) can be extended in length, but is never shortened.

Hybrid Heuristic

One shortcoming of the index heuristic is that it tends to instruct the searcher to leave her current location prematurely, as discussed in Section 3.3.2. This subsection presents a method to reevaluate the situation when $W_{ii}(s_i)$ in (3.9) is smaller than $W_{ij}(s_j)$ in (3.10) for some $j \neq i$ in order to decide whether or not the searcher should carry out an additional search at location i.

Our new hybrid heuristic works by instructing the searcher to search the current location one more time whenever the index heuristic would choose to do so, but using a different procedure to make decisions when the index heuristic instructs the searcher to move to a different location. Specifically, when the index heuristic asks the searcher to leave her current location, we identify the current location and m other locations having the highest indices (as defined by (3.10)) as candidate locations, where m is a predetermined integer between 2 and $n-1$. We then construct a new $(m+1)$-location sub-problem with the detection probabilities, search times, and travel times for these $m+1$ candidate locations. Prior probabilities for the locations are obtained by scaling the latest posterior probabilities so that they sum to one. The sub-problem is then solved using the finite-stage dynamic programming algorithm described in Section 3.2.3.

Suppose the optimal policy given by the DP algorithm states that the current location should be searched k times before moving to a different location (allowing for the possibility that $k = 0$). Then, under the hybrid heuristic policy, we search the current location k times and then move to whichever other location appears after the first move in the DP search sequence. We also carry out a single search at the new location. This process is then repeated, carrying out additional searches of the new location until the index policy suggests moving to a different location, at which point a new sub-problem is generated. The hybrid heuristic may visit the locations in a different order from that of the index policy, as it explicitly considers more than one possible alternative location as well as the current location when formulating the sub-problem.

Running the DP algorithm even once for a problem with $m + 1$ locations becomes computationally intractable as m increases. We are therefore limited to using small values of m. If m is fixed, then the heuristic scales well with the number of locations n in terms of computational burden. However, as n increases, the sub-problem becomes a less useful approximation of the actual problem, and the performance of the heuristic deteriorates as a result.

3.4.2 Methods for Improving a Search Sequence

This subsection presents two methods for improving a given search sequence. The first method is based upon the well-established technique of policy improvement in stochastic dynamic programming. The second method takes an arbitrary base sequence and tests the effects of making incremental, insertion-based changes to the sequence.

Policy Improvement

Policy improvement (PI) is a technique in stochastic dynamic programming that involves repeated application of Bellman's optimality principle in order to generate a sequence of policies, with each policy improving upon its predecessor. First, a convenient base policy is used to approximate the optimal value function for each state. In each iteration, an improved policy is obtained by choosing the best action in each state with respect to the approximated value function, which also yields a better approximation for the optimal value function. Under certain conditions, PI converges to an optimal policy after finitely many iterations (Puterman, 1994). Unfortunately, carrying out PI to optimality can be computationally intensive. In many cases, the biggest improvement is usually made in the first step of the process (Tijms, 2003).

Recall from Section 3.2.3 that for a given initial state $(1, (p_1, \ldots, p_n))$ and a search history $H = (i, (s_1, \ldots, s_n))$—the searcher is currently at location i, and location j has been searched for s_j times, for $j \in [n]$—we write $V(H)$ for the expected additional search time needed to find the object. The heuristic proposed here applies a single step of PI to a suitably-chosen base policy. Let V_{BASE} denote an approximation for the value function calculated by applying the chosen base policy to a given state. An approximation for the optimal value function V can then be derived as

$$V(i,(s_i,\ldots,s_n)) = \min_{j\in N}\left\{d_{ij}+t_j+\left(1-\frac{p_jq_j(1-q_j)^{s_j}}{\sum_{k\in N}p_k(1-q_k)^{s_k}}\right)V_{\text{BASE}}(j,(s_i^{(j)},\ldots,s_n^{(j)}))\right\} \tag{3.17}$$

where $V_{\text{BASE}}(j,(s_i^{(j)},\ldots,s_n^{(j)}))$ is an approximation for the expected remaining time to discovery starting from $(j,(s_i^{(j)},\ldots,s_n^{(j)}))$ if the base policy is followed for the remainder of the search.

Approximating the value of each possible state is not necessary, and would in fact be more computationally expensive than finding an exact DP solution. Instead, at each decision epoch, the searcher approximates the value of each state that could be reached following an additional unsuccessful search at one of the n locations. Since the base policy must be derived n times at each epoch it is necessary for the base policy to be very fast to compute.

Greedy Insertion

The ETD under a particular search sequence can be computed quickly, which makes it possible to look for improvement opportunities by evaluating small incremental changes. Suppose we are given a truncated search sequence for an initial state, such as a search sequence generated according to the index heuristic of section 3.3 or the hybrid heuristic of 3.4.1. This search sequence, referred to as the *base sequence*, is then partitioned into b blocks, where each block consists of a number of consecutive searches at the same location and adjacent blocks correspond to different locations. Next, generate a population of b distinct candidate search sequences, where the k^{th} candidate sequence extends the length of the k^{th} block by one search, for $k = 1, 2, \ldots, b$. Use (3.13) to calculate a lower bound for the ETD for each of these candidate search sequences. If none of these lower bounds are lower than that of the base sequence then we conclude that no further improvements can be made with this method. If one or more of the candidate search sequences produces a lower bound smaller than that of the base sequence then we update the base sequence to be the candidate search sequence that produces the lowest such lower bound. The process is then repeated until no further improvements can be found.

3.5 Numerical Study

In this section we evaluate the performances of the heuristics presented in Sections 3.3 and 3.4 by testing them over a range of randomly-generated problem instances.

We designate three different policies as *base heuristics*. The first of these is the index policy obtained from the restless bandit analysis given in Section 3.3. The second is the round-trip index heuristic introduced in Section 3.4.1, and the third is the hybrid heuristic described in Section 3.4.1.For the hybrid heuristic, we set $m = 2$ as the number of alternative locations; that is, it considers three candidate locations (including the current location) every time the DP procedure is called upon.

We evaluate these base heuristics both without any modifications and also after applying the methods for further improvement described in Section 3.4.2. For all three base heuristics, we test the effect of applying the greedy insertion method in Section 3.4.2. Also, for the index and round-trip heuristics we test the effect of applying the policy improvement (PI) method in Section 3.4.2, followed by greedy insertion as a final step. PI is applied first because it can change the order of location visited and its runtime is very dependent on the base policy. We do not apply PI to the hybrid heuristic as this proves to be too computationally expensive due to the hybrid's use of DP.

Given that we have proposed several different base heuristics, it is also insightful to test the performance of PI with more than one base heuristic. This method is recommended by Bertsekas (2013) as a way to enhance the performance of PI in general. Essentially the method works as follows: at each decision epoch we consider each of the base heuristics separately and, for each of these heuristics, carry out the procedure described in Section 3.4.2, with the function V_{BASE} corresponding to the base heuristic under consideration. In this way we obtain several different approximations for

$V(i(s_i, \ldots, s_n))$, defined in Section 3.4.2 as an approximation for the optimal expected remaining time to discovery. We choose the action that yields the smallest approximation for $V(i(s_i, \ldots, s_n))$. In our experiments we carry out this method with only two base heuristics (index and round-trip), as it becomes too slow computationally when the hybrid heuristic is included.

3.5.1 Scenario Generation

In order to test the relative performances of our heuristics, we generated 400 scenarios with 4 locations ($n = 4$) and 400 scenarios with 8 locations ($n = 8$). In the 4-location scenarios, it is computationally feasible to approximate the optimal ETD using DP, as described in Section 3.2.3. Thus, we can assess how close our heuristics are to optimality. In the 8-location scenarios, however, DP becomes too computationally cumbersome and we can only judge the performances of the heuristics by comparing them with each other.

To generate the scenarios for $n = 4$ in our study, we began by randomly sampling 100 sets of parameter values as follows:

- Each prior probability p_i was sampled independently from a U(0,1) distribution. These probabilities were then re-scaled to satisfy $\sum_{i\in[n]} p_i = 1$.
- Each detection probability q_i was sampled independently from a U(0.2,0.9) distribution.
- Each search time t_i was sampled independently from a U(0.1,1) distribution.
- To represent the geography of the problem, the travel times were calculated by first randomly assigning each location a point in the unit square. That is, each location was given a pair of coordinates (x, y) with x and y sampled independently from a $U(0, 1)$ distribution. The travel times d_{ij} for $i, j \in [n]$ were then set in proportion to the corresponding Euclidean distances, so that

$$d_{ij} = \alpha\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \qquad i, j \in [n],$$

where $\alpha > 0$ is the constant of proportionality. The travel time between any two locations generated in this way is bounded between 0 and $\alpha\sqrt{2}$. We refer to α as a *dispersion parameter*, as it has the effect of scaling the distances between locations.

Each of these 100 scenarios was then tested under four different cases for the value of α, giving 400 distinct scenarios in total. It is clear that as $\alpha \to 0$, travel times become negligible and thus the index heuristic of Section 3.3 tends to the Gittins' index policy, which is optimal in the limit. Thus, when α is close to zero, we can expect all of our heuristics to achieve near-optimal performances. We therefore limited our numerical study to values of α that are sufficiently large to present a challenge to the performance of the heuristics. Specifically, we considered the cases $\alpha = 1$, $\alpha = 2$, $\alpha = 5$ and $\alpha = 10$. As α increases, the incentive for the searcher to change location becomes smaller, so that optimal search sequences are more likely to consist of longer blocks of consecutive searches in the same location.

To generate the scenarios for $n = 8$, we used exactly the same method as for $n = 4$ and obtained another set of 100 scenarios which were then replicated across the same four values of α.

3.5.2 Results for 4 Locations

First we assess the effectiveness of our heuristics in problems with $n = 4$. As discussed earlier, the optimal ETDs and the ETDs given by the heuristics in our experiments can only be approximated, although the approximation errors can be made arbitrarily small. To approximate the optimal ETD in a particular scenario we use finite-stage dynamic programming as described in Section 3.2.3, with the value $\varepsilon = 10^{-7}$ used to determine the values b_i for $i \in [4]$. For each of the heuristics that we consider, we obtain a pessimistic estimate of the ETD of the corresponding search sequence by using (3.15). We again use $\varepsilon = 10^{-7}$ to determine the point of truncation when calculating $\mathbb{E}_{LB}[T]$. We then estimate the suboptimalities of our heuristics by calculating their percentage deviations from the approximate optimal value. Given the pessimistic nature of the estimate in (3.15), these suboptimality percentages can also be seen as pessimistic, which implies that the performances of our heuristics are at least as good as indicated by the reported results.

The results from the 400 scenarios with $n = 4$ are reported in Table 3.1. For each value of α we have reported the average (approximate) percentage suboptimalities of the various heuristics, as well as the 75^{th}, 95^{th} and 99^{th} percentiles of these suboptimalities and the average runtime (in seconds) over all scenarios. The three columns under "Base" show the results of testing the three base heuristics without the use of either greedy insertion or PI. The three columns under "Insertion" show the results of applying greedy insertion to each of the base policies. The first two columns under "PI and Insertion" show the results of applying PI (first) and greedy insertion (second) to the index and round-trip heuristics, while the final column (entitled "Both") shows the results obtained by using both the index and the round-trip as base heuristics within PI as described at the start of Section 3.5, with greedy insertion also used as a final step.

As shown in Table 3.1, prior to the application of any further improvement method (insertion or PI), the percentage suboptimalities of the round-trip and hybrid heuristics are significantly lower than that of the index heuristic across all values of the dispersion parameter α. The hybrid heuristic is the best-performing of the three base heuristics at all but the highest percentile level (99%), but it also has a significantly higher mean running time. The performances of the heuristics are fairly consistent across all values of α, but the runtime of the hybrid heuristic notably decreases as α increases. This reduction in runtime is probably due to the generated sequences changing location less frequently as travel times increase, implying that fewer DP sub-problems need to be constructed.

We can see that, across all base heuristics, greedy insertion provides major improvement at a relatively small extra computational cost. The index heuristic benefits from the largest reduction in percentage suboptimality after applying greedy insertion, since its major weakness is a tendency to leave the current location too soon. For the base heuristics we can see that the mean running time is lower for the index heuristic

	Base			Insertion			PI and Insertion		
Scenario/Metric	Index	R-Trip	Hybr	Index	R-Trip	Hybr	Index	R-Trip	Both
$\alpha = 1$									
Mean	6.109	2.371	2.009	2.781	2.201	1.767	0.584	0.357	0.322
75th Percentile	8.069	3.34	1.492	3.827	2.818	1.196	0.759	0.301	0.301
95th Percentile	13.551	7.672	6.674	9.826	7.672	6.674	2.385	1.824	1.789
Runtime (s)	0.032	0.044	30.308	1.12	0.165	30.37	12.053	14.257	24.556
$\alpha = 2$									
Mean	6.459	1.865	1.555	2.124	1.715	1.258	0.626	0.314	0.329
75th Percentile	8.686	2.37	1.635	2.847	2.148	1.119	0.799	0.337	0.372
95th Percentile	11.936	5.409	7.419	6.724	5.323	5.109	2.348	1.473	1.57
Runtime (s)	0.033	0.086	24.474	0.811	0.187	24.527	12.037	14.904	24.697
$\alpha = 5$									
Mean	7.131	2.502	2.148	3.016	2.42	1.861	0.693	0.34	0.34
75th Percentile	9.211	3.079	2.352	3.946	3.075	1.35	0.621	0.256	0.256
95th Percentile	15.577	12.365	11.185	12.475	12.12	11.158	4.005	1.867	1.867
Runtime (s)	0.033	0.046	20.097	0.54	0.103	20.138	12.411	15.623	24.806
$\alpha = 10$									
Mean	6.74	2.625	2.184	3.046	2.594	1.844	0.549	0.372	0.372
75th Percentile	9.225	2.903	1.56	4.412	2.902	0.856	0.406	0.293	0.293
95th Percentile	16.339	13.065	11.031	13.625	13.034	10.581	2.679	1.953	1.953
Runtime (s)	0.033	0.048	16.715	0.41	0.075	16.759	12.569	16.02	26.456

Table 3.1: Comparisons between heuristics with respect to percentage suboptimalities and runtimes with $n = 4$.

than for the round-trip heuristic. This observation makes sense, since the first step of the round-trip heuristic is to generate a search sequence using the index heuristic. However, after insertion, the mean running time is higher for the index heuristic than for the round-trip heuristic. This change is due to the number of insertion steps that are needed. The base round-trip heuristic is much closer to optimal and so requires significantly fewer additional searches to be added to the sequence before reaching the point where no further improvements can be found.

Further significant improvements can be seen when PI is included. When comparing the two versions of PI with a single base heuristic it appears that the round-trip heuristic is a more effective base heuristic than the index heuristic across all metrics. As noted in Section 3.4.2, the use of PI enables the round-trip heuristic to visit the locations in a different order from that of the (base) index heuristic, and it seems to become much stronger as a result. However, we should also note that PI in general is computationally expensive, as indicated by the runtimes in the table.

When both the index and round-trip heuristics are used as base heuristics within PI, we obtain further marginal improvements in the $\alpha = 1$ case, although the results for other α values are inconclusive. However, the runtime increases significantly in this situation. It is similar to the total time required to run both single-policy versions sequentially, which makes sense as PI largely consists of repeated application of the chosen base heuristic.

3.5.3 Results for 8 Locations

Next, we evaluate the heuristics in larger problem instances with $n = 8$. For problems of this size, approximating the optimal value of the ETD using finite-stage DP becomes computationally intractable. Instead, in each scenario we use the performance of the best-performing heuristic as a benchmark and calculate the percentage deviations of the other heuristics from this benchmark. We refer to these differences as *percentage deficits*. Within each scenario, the ETDs for the various heuristics are approximated in exactly the same way as in the $n = 4$ case.

Results for the 400 experiments with $n = 8$ are presented in Table 3.2. The results are labeled in the same way as those in Table 3.1, except that percentage deficits (with respect to the best-performing heuristic) are shown rather than percentage suboptimalities.

	Base			Insertion			PI and Insertion		
Scenario/Metric	Index	R-Trip	Hybr	Index	R-Trip	Hybr	Index	R-Trip	Both
$\alpha = 1$									
Mean	8.203	4.25	2.736	4.057	4.087	2.282	0.905	0.201	0.2
75th Percentile	10.078	5.068	4.185	4.813	4.851	3.674	1.584	0.118	0.086
95th Percentile	13.424	7.67	6.892	7.645	7.556	6.292	3.185	1.166	1.166
Runtime (s)	0.159	0.225	97.01	17.115	2.04	98.772	240.881	284.697	459.857
$\alpha = 2$									
Mean	10.108	4.562	3.653	4.58	4.347	2.798	0.911	0.368	0.368
75th Percentile	12.903	6.044	5.219	6.641	5.621	4.128	1.367	0.208	0.208
95th Percentile	15.486	9.059	8.122	10.223	8.9	7.41	3.607	1.796	1.796
Runtime (s)	0.158	0.222	71.729	14.972	1.789	73.636	240.682	290.814	459.48
$\alpha = 5$									
Mean	11.723	5.143	4.305	5.469	5.052	3.362	0.949	0.369	0.369
75th Percentile	15.615	6.907	6.151	8.153	6.907	4.912	1.145	0.227	0.227
95th Percentile	18.544	12.062	10.215	12.518	12.062	9.001	4.406	1.85	1.85
Runtime (s)	0.161	0.463	55.683	8.747	1.148	57.219	240.696	306.997	467.694
$\alpha = 10$									
Mean	12.019	5.539	4.979	5.998	5.511	3.835	0.99	0.671	0.671
75th Percentile	15.881	8.75	8.383	8.84	8.75	6.746	0.999	0.215	0.215
95th Percentile	20.949	13.543	12.239	14.161	13.543	11.11	6.037	5.111	5.111
Runtime (s)	0.164	0.235	48.78	9.84	0.622	50.257	243.637	309.013	481.897

Table 3.2: Comparisons between heuristics with respect to percentage deficits and runtimes with $n = 8$

As in the 4-location problems, we find that when the base heuristics are used without any further subsequent improvements, the hybrid heuristic achieves the best performance, followed by the round-trip heuristic. Notably, the mean percentage deficits for all three base heuristics are significantly higher than the mean percentage suboptimalities found in the 4-location problems. This observation indicates that for larger problems, the gap between the base heuristics and the optimal solution is larger, and our subsequent improvement methods can improve the base heuristics more. As one would expect, the mean running times for all three heuristics are higher than in the

4-location case. For the index and round-trip heuristics the runtime is approximately five times longer, whereas for the hybrid heuristic it is about three times longer.

The greedy insertion method improves the performances of all three base heuristics. For the index heuristic, the application of insertion roughly halves the average percentage deficit at the cost of a significant increase in runtime. As in the 4-location case, the round-trip heuristic with insertion is faster than the other post-insertion heuristics, but it does not perform as well as hybrid with insertion.

The combined use of PI and insertion reduces the percentage deficit further under both base heuristics (index and round-trip). As in the 4-location case, we find that PI is particularly effective when applied to the round-trip heuristic, as it enables the order of visiting locations to be changed from the default order given by the base index heuristic. When both base policies are combined within PI we do not observe significant improvements compared to the case where the round-trip heuristic is used as the only base heuristic. We also note that the runtimes with PI are generally about 20 times longer than that in the 4-location case, indicating that PI does not scale to larger problem sizes as easily as the base heuristics themselves.

3.6 Conclusions

In this paper we have considered a search problem with similar assumptions to those studied in the literature and extended it in order to include travel times between locations. When the number of locations is small, a close approximation to an optimal policy can be found using finite-stage dynamic programming. For larger problems, we must consider heuristic approaches.

We have developed an index heuristic based on restless bandit theory. While the index heuristic possesses a few desired properties, its major weakness is that it tends to instruct the searcher to leave her current location too soon. To remedy this weakness we have developed a few methods to improve the index heuristic, and use numerical experiments to demonstrate their effectiveness.

There are several possible directions for further research. The assumption that the object must be in one of the n locations ($\sum_i p_i = 1$) could be relaxed, and alternative objectives could also be considered, such as maximizing the probability of finding the object by a certain deadline. A game theoretical dimension could also be incorporated by allowing an intelligent hider to choose a location for the object in order to make the searcher's task as difficult as possible.

3.7 Appendix

3.7.1 Proof of Theorem 3.1

Suppose the search state is $(i, p'_1, \ldots, p'_n)$. We need to show that if

$$\frac{p'_i q_i}{t_i} > \frac{p'_j q_j}{t_j} \qquad \forall\, j \in [n] \setminus \{i\}$$

then it is optimal to search location i.

Suppose the prerequisite conditions hold and consider an arbitrary search policy. Let π be the search sequence obtained by applying that policy to the current state and suppose that it starts by searching a location other than i. Let m denote the position of the first appearance of i in π. To show that π is not optimal we consider a modified search sequence $\hat{\pi}$ in which the first search of i is moved to the beginning of the sequence. To ease the comparison we assume that, when following $\hat{\pi}$, the searcher visits location i after carrying out the first m searches even if no searches take place at i on that particular visit. This assumption ensures that both sequences are identical after the first m searches, since both truncated search sequences visit each location the same number of times, have the same duration and end at location i. It is therefore sufficient to consider only the first m searches of both sequences.

To demonstrate that $\hat{\pi}$ is superior to π we consider the expected discounted rewards of both truncated sequences. Write $\mathbb{E}_1(a^T)$ and $\mathbb{E}_2(a^T)$ to denote the contribution of the first m searches to the expected discounted reward for π and $\hat{\pi}$ respectively. (Recall that the expected discounted reward is defined in (3.6).) We will show that there exists an $a^* > 0$ for which

$$1 > a > a^* \implies \mathbb{E}_1(a^T) < \mathbb{E}_2(a^T) \implies \frac{1 - \mathbb{E}_1(a^T)}{1-a} > \frac{1 - \mathbb{E}_2(a^T)}{1-a}$$

and hence, taking the limit $a \to 1$, we obtain $\mathbb{E}_1(T) \geq \mathbb{E}_2(T)$. To show that $\mathbb{E}_1(a^T) < \mathbb{E}_2(a^T)$ for all $a \in (a^*, 1)$ with a suitably chosen a^*, we make a series of changes to the expression for $\mathbb{E}_1(a^T)$ and show that each change results in an increased value for the expected discounted reward. The intermediate expressions that we obtain by making these changes do not necessarily correspond to the expected discounted rewards of admissible search sequences, but the final expression is equal to $\mathbb{E}_2(a^T)$, ensuring that the proof is valid.

Let $\tau_\pi(k)$ denote the time at which the kth search is completed when following π. In the argument that follows, the first change we make is to allow the first search of location i to begin immediately after the completion of the previous search. At each subsequent step, we bring the search of location i forwards so that it starts immediately after the search that took place two searches earlier. Additionally, the search that would have taken place immediately before the search of i is delayed by t_i time units as a result.

More formally, the contribution of the first search of i to $\mathbb{E}_1(a^T)$ is $p'_i q_i a^{\tau_\pi(m)}$. Let the position of this search be moved so that it starts immediately after the completion of the previous search. Then its contribution becomes $p'_i q_i a^{\tau_\pi(m-1)+t_i}$. Since all other terms are unchanged and $\tau_\pi(m) > \tau_\pi(m-1)+t_i$ it is clear that the expected discounted reward increases as a result. Next, let j denote the location immediately preceding the first search of i and write g_j for the number of times that location j is searched before this point. The contribution of this search to $\mathbb{E}_1(a^T)$ is $p'_j q_j (1-q_j)^{g_j} a^{\tau_\pi(m-1)}$. We now make adjustments to these two searches. Let the search of location i conclude at time $\tau_\pi(m-2)+t_i$ and let the search of location j conclude at time $\tau_\pi(m-1)+t_i$. The contributions of these two searches are now $p'_i q_i a^{\tau_\pi(m-2)+t_i}$ and $p'_j q_j (1-q_j)^{g_j} a^{\tau_\pi(m-1)+t_i}$ respectively. The contributions of all other searches remain unchanged. The increase in the contribution of the search at i is $p'_i q_i a^{t_i}(a^{\tau_\pi(m-1)} - a^{\tau_\pi(m-2)})$. The decrease in the contribution of the search at j is

$$p'_j q_j (1-q_j)^{g_j} a^{\tau_\pi(m-1)}(a^{t_i} - 1) \leq p'_j q_j (1-q_j)^{g_i} a^{t_j}(a^{\tau_\pi(m-2)+t_i} - a^{\tau_\pi(m-2)}).$$

However by hypothesis we have

$$\frac{p_i'q_i}{t_i} > \frac{p_j'q_j}{t_j}$$

and hence

$$\frac{p_i'q_i}{t_i} > \frac{p_j'q_j}{t_j}(1-q_j)^{g_j}$$

for any $g_j \geq 0$. From this we conclude that there exists an $a_1 < 1$ for which $1 > a > a_1$ implies that

$$\frac{p_i'q_i a^{t_i}}{a^{\tau_\pi(m-2)+t_i} - a^{\tau_\pi(m-2)}} \geq \frac{p_j'q_j(1-q_j)^{g_j}a^{t_j}}{a^{\tau_\pi(m-1)} - a^{\tau_\pi(m-2)}}$$

and hence that

$$p_i'q_i a^{t_i}(a^{\tau_\pi(m-1)} - a^{\tau_\pi(m-2)}) \geq p_j'q_j(1-q_j)^{g_j}a^{t_j}(a^{\tau_\pi(m-2)+t_i} - a^{\tau_\pi(m-2)}).$$

This confirms that the manoeuvre proposed yields an increase in the expected discounted reward for the range $a \in (a_1, 1)$.

The next maneuver similarly yields an increase for some range $a \in (a_2, 1)$, say. All desired manoeuvres (of which there are a finite number) therefore yield an increase for some range $a \in (a^*, 1)$, from which we can conclude that

$$1 > a > a^* \implies \mathbb{E}_1(a^T) < \mathbb{E}_2(a^T) \implies \frac{1-\mathbb{E}_1(a^T)}{1-a} > \frac{1-\mathbb{E}_2(a^T)}{1-a},$$

as required. $\square$

3.7.2 Proof of Theorem 3.4

Consider the symmetric travel case, where any two locations can be moved between in d units of time. Suppose the searcher is at location i and, for each $k \in [n]$, location k has been searched s_k times. We need to show that if

$$W_{ii}(s_i) = \frac{p_i q_i (1-q_i)^{s_i}}{t_i} > W_{ij}(s_j) = p_j(1-q_j)^{s_j} \sup_k \left\{ \frac{1-(1-q_j)^k}{d+kt_j} \right\} \qquad \forall\, j \in [n] \setminus \{i\}$$

then it is optimal to search location i next.

Suppose the prerequisite conditions hold and consider an arbitrary policy. Let π be the search sequence corresponding to that policy and suppose it starts by searching a location other than i. This search sequence can be uniquely partitioned into blocks, where each block consists of consecutive searches at the same location and adjacent blocks correspond to different locations. Let the blocks be numbered according to the order that they appear in the sequence π. For each block $k \in \mathbb{N}$, let b_k and c_k denote the location being searched and number of searches in the block respectively.

Suppose that the first block of searches at location i is block $m \geq 2$. To show that π is not optimal we define a modified sequence $\hat{\pi}$ that follows π but with the first search of i moved to the front. If $c_m = 1$ then, under the new sequence $\hat{\pi}$, we assume that the searcher travels to location i after the first $(m-1)$ blocks but does not search there. This assumption ensures that all subsequent searches happen at exactly the same time under both policies. Consequently, it is sufficient to focus entirely on the truncated search sequences involving only the first m blocks.

To demonstrate that $\hat{\pi}$ is superior to π we consider the expected discounted rewards for both truncated sequences. Write $\mathbb{E}_1(a^T)$ and $\mathbb{E}_2(a^T)$ to denote the contributions of the searches in the truncated sequences for π and $\hat{\pi}$ respectively to their expected discounted rewards. We show that there exists an a^* for which

$$1 > a > a^* \implies \mathbb{E}_1(a^T) < \mathbb{E}_2(a^T) \implies \frac{1-\mathbb{E}_1(a^T)}{1-a} > \frac{1-\mathbb{E}_2(a^T)}{1-a}$$

and hence, taking the limit $a \to 1$, we have $\mathbb{E}_1(T) \geq \mathbb{E}_2(T)$. To show that $\mathbb{E}_1(a^T) < \mathbb{E}_2(a^T)$, $a \in (a^*, 1)$, for some a^*, we make a series of changes to the expression for $\mathbb{E}_1(a^T)$, each of which increases its value. The intermediate expressions do not necessarily correspond to valid search sequences, but importantly the final expression corresponds to $\mathbb{E}_2(a^T)$.

The first change is to allow the first search of location i to begin immediately after the completion of the previous search. The timings of all subsequent searched remain unchanged. This new sequence does not correspond to an admissible search sequence for our problem, as it allows the searcher to search location i without moving to it. Since the searcher gains the reward for searching i earlier, this maneuver clearly increases the expected discounted reward.

To be more explicit, let $\tau_\pi(k)$ denote the time at which the final search of the k^{th}

block is completed under policy π. Thus,

$$\tau_\pi(k) = \sum_{i=1}^{k}(d + t_{b_i} c_{b_i}).$$

The contribution of the first search of i to $\mathbb{E}_\pi[a^T]$ is $p_i q_i (1-q_i)^{s_i} a^{\tau_\pi(m-1)+d+t_i}$. If the search of i is started immediately after the completion of the previous search, this contribution becomes $p_i q_i (1-q_i)^{s_i} a^{\tau_\pi(m-1)+t_i}$. Since all other terms remain unchanged, it is clear that this maneuver increases the expected discounted reward. At each subsequent step, we bring the search of location i forwards so that it starts before the searcher moves to the preceding block of searches. Additionally, each search in that block is delayed by t_i units of time. These intermediate steps do not correspond to valid search sequences, as they allow the searcher to move to location i and move away from it again without incurring any travel time. However, after the final maneuver is completed, the search of i appears at the front of the sequence. Since i is the starting location, the resulting sequence is valid and moreover corresponds to the modified sequence $\hat{\pi}$.

It remains to show that each of these intermediate steps increases the expected discounted reward. Each of these changes involves bringing forward the reward for the first search of i and delaying the rewards from one block of searches. More explicitly, for each $k \in \{1, \ldots, m-1\}$, the change corresponding to moving the first search of i in front of block k is defined as follows. For notational convenience, let $j = b_k$. Then the reward earned for searching i is brought forward by $d + c_j t_j$ units of time and the rewards earned for the searches in block k are each delayed by t_i.

Consider the discounted rewards earned while carrying out the searches in the k^{th} block. Let g_j be the total number of times that j appears in the search sequence prior to this block of searches. The sum of these contributions is

$$\sum_{w=1}^{c_j} p_j q_j (1-q_j)^{s_j+g_j+w-1} a^{\tau_\pi(k-1)+d+wt_j}.$$

If these rewards are delayed by t_i time units then the contribution becomes

$$\sum_{w=1}^{c_j} p_j q_j (1-q_j)^{s_j+g_j+w-1} a^{\tau_\pi(k-1)+t_i+d+wt_j}.$$

The decrease in the contribution of the searches at j can be found by taking the difference of these expressions, which is

$$\sum_{w=1}^{c_j} p_j q_j (1-q_j)^{s_j+g_j+w-1} a^{\tau_\pi(k-1)+d+it_j}(1-a^{t_i}).$$

If the reward for the first search of i is earned at time $\tau_\pi(k)+t_i$ then the contribution of that search is

$$p_i q_i (1-q_i)^{s_i} a^{\tau_\pi(k)+t_i} = p_i q_i (1-q_i)^{s_i} a^{\tau_\pi(k-1)+c_j t_j+d+t_i}.$$

On the other hand, if the reward for the first search of i is earned at time $\tau_\pi(k-1)+t_i$ then the contribution of that search becomes $p_i q_i (1-q_i)^{s_i} a^{\tau_\pi(k-1)+t_i}$. The contributions of all other searches remain unchanged. The increase in the contribution of the search at i is

$$p_i q_i (1-q_i)^{s_i} a^{\tau_\pi(k-1)+t_i}(1-a^{d+c_j t_j}).$$

Hence, it remains only to verify that

$$p_i q_i (1-q_i)^{s_i} a^{\tau_\pi(k-1)+t_i}(1-a^{d+c_j t_j}) > \sum_{w=1}^{c_j} p_j q_j (1-q_j)^{s_j+g_j+w-1} a^{\tau_\pi(k-1)+d+it_j}(1-a^{t_i}).$$

However, by hypothesis we have

$$\frac{p_i q_i (1-q_i)^{s_i}}{t_i} > p_j (1-q_j)^{s_j} \frac{1-(1-q_j)^{c_j-1}}{d+c_j t_j},$$

From this we conclude that there exists an $a_1 < 1$ for which $1 > a > a_1$ implies that

$$p_i q_i (1-q_i)^{s_i} \frac{a^{t_i}}{1-a^{t_i}} > p_j q_j (1-q_j)^{s_j} \frac{a^{d+c_j t_j}}{1-a^{d+c_j t_j}} \frac{1-(1-q_j)^{c_j-1}}{q_j}.$$

and hence that

$$p_i q_i (1-q_i)^{s_i} \frac{a^{t_i}}{1-a^{t_i}} > p_j q_j (1-q_j)^{s_j+g_j} \frac{a^{d+c_j t_j}}{1-a^{d+c_j t_j}} \sum_{w=1}^{c_j} (1-c_j q_j)^{w-1}.$$

This confirms that the manoeuvre proposed yields an increase in the expected discounted reward for the range $a \in (a_1, 1)$.

The next manoeuvre similarly yields an increase for some range $a \in (a_2, 1)$, say. All desired manoeuvres (of which there are a finite number) yield an increase for some range $a \in (a^*, 1)$ from which we can conclude that

$$1 > a > a^* \implies \mathbb{E}_1(a^T) < \mathbb{E}_2(a^T) \implies \frac{1-\mathbb{E}_1(a^T)}{1-a} > \frac{1-\mathbb{E}_2(a^T)}{1-a},$$

as required. □

Chapter 4

Patrolling Dispersed Locations

4.1 Introduction

Organisations are often required to protect their assets from sabotage. If these assets are in geographically dispersed locations, it can be prohibitively expensive to actively protect them all at the same time. For example, when defending a large military facility, it is impractical to guard all sections of the facility's perimeter simultaneously. Similarly, local law enforcement cannot be present in all parts of their jurisdiction at once. To ensure that none of the assets are left undefended for too long, it is necessary for one or more agents, usually called patrollers, to sequentially visit and search each of these locations.

The study of patrol theory arose in the 1970s. Much of the early work, such as that of Olson and Wright (1975), Chaiken and Dormont (1978) and Birge and Pollock (1989), focused on optimally allocating police resources to intercept ongoing criminal activity. In each of these papers, the authors assume that the crime rates in different locations are known and use these rates to inform their policies. While making use of known crime rates is useful, intelligent attackers may also take this information into account and try to act unpredictably to avoid detection. Strategic planning on the part of the attackers is more likely to occur for more impactful attacks such as acts of terrorism or when a single decision maker is planning multiple coordinated attacks. In the last few decades, much of the focus of the patrol literature has focused on game theoretic models such as in Auger (1991), Alpern and Gal (2006), Lin et al. (2013) and Lin et al. (2014). A more detailed review of the patrol literature can be found in Chapter 2.

In this chapter we consider a patrol problem where a patroller is tasked with minimising the cost of damage caused by attacks within an area known as the search space. An attack is broadly construed as an illicit activity undertaken by an attacker that has an ongoing negative effect such as damaging infrastructure or leaking information. The area under the patroller's protection contains n key locations, for some $n \in \mathbb{N}$, that could be subject to an attack. In contrast with many of the patrol problems considered in the recent literature, once an attack has started, it continues until the attacker is discovered by the patroller. To the best of our knowledge there is only one paper that considers an attacker that remains at a location until detected. Specifically, Blachman (1959) introduces a variant of the search problem introduced by Koopman (1956) where the target of the search is not initially present but arrives after some random amount of time that is uniformly distributed over some very long interval. As in our model, it is assumed that the target remains until detected.

The patroller moves between the locations and attempts to interrupt ongoing attacks. The time required for her to travel from location i to a different location j is d_{ij}. By definition, $d_{ij} \geq 0$ for all $i \neq j$ and $d_{ii} = 0$ for all i. We assume that a direct route from one location to another can take no longer than an indirect route since the patroller can always travel via other intermediate locations without searching them. Thus for all distinct $i, j, k \in [n]$, $d_{ik} \leq d_{ij} + d_{jk}$. However, we do not enforce that $d_{ij} = d_{ji}$ since travel times are not always dependant only on the distance travelled. For example, travelling up a hill, potentially with heavy search equipment, may take longer than making the reverse journey.

When she arrives at a location the patroller can choose to spend any amount of time searching for attackers. For each attacker present at that location, the patroller discovers them at rate λ_i. If more than one attacker is present then each attacker will be detected independently at rate λ_i. Thus, the instantaneous discovery rate is λ_i multiplied by the number of hidden attackers in that area. Any attacker present at location i will be discovered by the patroller in time $h > 0$ with probability $\lambda_i h + o(h)$ independently of all else. When an attacker is discovered the attack ends instantly and the patroller is free to continue her search of that location or to move elsewhere.

The objective of the patroller is to minimise the expected time to discover an attacker regardless of where the attack occurs, if and when an attack occurs. The rationale of this objective is that in many applications the patroller does not know where the attacker is likely to attack. If an adversary is actively choosing a location to attack to cause maximal harm, then the patroller's objective is to minimise this maximal harm that the attacker can cause. In the worst case scenario, the adversary may be aware of the patroller's policy, although we always assume they are unaware of the patroller's current location. It is therefore beneficial for the patroller to construct a robust policy that minimises the damage incurred in the worst case scenario where all attackers are directed to the least well defended location. To formulate this objective we shall assume that the attackers arrive according to a Poisson process and allow multiple attacks simultaneously. The patroller's objective is to minimise the maximal expected time to discover an attack among all locations. Formulating the problem in this way makes it a sequential move game where the patroller acts first by choosing a patrol strategy and the adversary moves second by choosing to attack the location with the longest expected time to discovery. Our goal is to minimise the expected time until discovery of an attacker regardless of where the attack occurs. Under the Poisson Process assumption on attack arrivals we have that, conditional on an attacker being present at a location, its time spent undiscovered is uniform over the appropriate time interval irrespective of the Poisson attack rate. Further, in our model, under any policy, the presence of additional attackers at a given location does not affect the expected time to discover a particular attacker. Thus, the rate of the arrival process has no effect on the performance of the chosen policy. Another consequence of detection events being independent is that it is equivalent to minimising the expected time to discover a single attacker. In this case the attacker's arrival time is determined by a uniform distribution over a very large interval.

Section 4.2 considers the special case where there are no travel times between locations and derives an optimal cyclic policy in this case. In Section 4.3 we define a simple cycle as a repeating patrol pattern that visits each location exactly once per repetition. We give some results that suggest that simple cycles perform well for a very broad class of patrol problems and derive a formula for the expected time for a patroller to discover an attacker at a particular location when following a simple cycle. We describe how to use this formula to find the best simple cycle when the detection rates are homogeneous. In Section 4.4 we define a sweep cycle that involves the patroller moving back and forth along a line of points. We use the same approach to find formulas for the expected time for a patroller to discover an attacker at a particular location when following a sweep cycle. In Section 4.5 we introduce an algorithm for finding the best patrol pattern of a particular cycle type that doesn't rely on homogeneity of detection rates. Finally, Section 4.6 concludes this chapter and gives some suggestions for further work.

4.2 The Case of No Travel Times

Consider a special case of the patrol problem presented in Section 4.1 where there are no travel times. In other words, $d_{ij} = 0$ for all $i, j \in [n]$. Since the patroller can move between any pair of locations instantaneously, we allow the patroller to allocate her effort among multiple locations at the same time. For each $i \in [n]$, if the fraction of effort the patroller allocates to location i at time $t \geq 0$ is $h_i(t) \in [0, 1]$, with $\sum_{i=1}^{n} h_i(t) = 1$, then at time t the instantaneous detection rate at location i is $\lambda_i h_i(t)$, for $i = 1, \ldots, n$.

In this section we will show that the optimal policy for this special case is for the patroller to allocate a certain proportion of her effort to each location which remains constant throughout. We will also derive what this proportion is. To prove this result we first require the following lemma.

Lemma 4.1. *Suppose $g(x)$ is a function defined on $x \in [0, c]$ for some $c \in \mathbb{R}^+$. If there exist $a, b \in \mathbb{R}^+$ with $a < b$ such that $g(x) \in [a, b]$ for $x \in [0, c]$, then*

$$\frac{\int_0^c g(x)dx}{c} \geq \frac{c}{\int_0^c \frac{1}{g(x)}dx},$$

with equality if $g(x)$ is constant for $x \in [0, c]$.

Proof. Let $g(x)$ be a function defined on $x \in [0, c]$ for some $c \in \mathbb{R}^+$. Also let $\phi(y) = 1/y$ and $Y = g(X)$ where X is a uniform random variable over $[0, c]$. Jensen's inequality states that if ϕ is convex and Y is an integrable real valued random variable then

$$\phi(E[Y]) \leq E[\phi(Y)].$$

Since $\phi(y)$ is clearly convex, it follows that

$$\frac{1}{E[g(X)]} \leq E\left(\frac{1}{g(X)}\right).$$

Expressing these expectations as integrals gives us

$$\frac{c}{\int_0^c g(x)dx} \leq \frac{\int_0^c \frac{1}{g(x)}dx}{c}.$$

Inverting both sides yields the desired inequality

$$\frac{\int_0^c g(x)dx}{c} \geq \frac{c}{\int_0^c \frac{1}{g(x)}dx}.$$

It is clear to see that if $g(x)$ is constant for $x \in [0, c]$ then equality is achieved. □

This lemma is the continuous version of the inequality that states that the arithmetic mean is greater than or equal to the harmonic mean.

Recall that targets arrive into the system according to a Poisson process and that the patroller's objective is to minimise the maximum expected time a target spends at a location before being detected across all locations.

Theorem 4.2. *Consider a particular location and write λ for the patroller's detection rate at this location. If the patroller adopts a cyclic patrol pattern where the long-run fraction of effort spent at this location is capped at $r \in (0, 1)$, then to minimise the expected time to detect a target at this location, it is optimal for the patroller to continuously allocate the same fraction of effort r at this location at all times. The minimised expected time to detection is $1/(r\lambda)$.*

Proof. An arbitrary policy can be delineated by some $c > 0$ and a function $h(t) \in [0, 1]$ defined for $t \in [0, c]$ with the interpretation that the patroller allocates $h(t) \in [0, 1]$ fraction of her effort at time $t \in [0, c]$ such that

$$\frac{\int_0^c h(t)dt}{c} = r, \tag{4.1}$$

and repeats the same pattern of length c indefinitely. It follows from renewal reward theory that the long-run fraction of effort allocated to this location is r.

Consider a target that arrives at the location at a random time. Since attackers arrive according to a Poisson process, they are equally likely to arrive at any point during the cycle. The amount of time that has passed since the beginning of the current cycle at the point of arrival is uniformly distributed over $[0, c]$. It remains to show that the expected time the target spends at the location before being detected by the patroller is minimised if $h(t) = r$ for $t \in [0, c]$.

To begin, write $g(t)$ for the expected time for the patroller to detect a target that arrives at time $t \in [0, c]$, so we seek to minimise

$$\int_0^c g(t)\,\frac{1}{c}\,dt = \frac{\int_0^c g(t)dt}{c}. \tag{4.2}$$

Consider a target arriving at time t and condition on whether the target is detected in the interval $[t, t + \delta)$ to get

$$g(t) = \delta + (1 - \lambda h(t)\delta + o(\delta))\, g(t + \delta),$$

because with probability $1 - \lambda h(t)\delta + o(\delta)$ the target is still at the location at time $t + \delta$ so the expected additional time to detection is $g(t + \delta)$. Rearrange the preceding to get

$$\frac{g(t + \delta) - g(t)}{\delta} = \left(\lambda h(t) + \frac{o(\delta)}{\delta}\right) g(t + \delta) - 1.$$

Taking $\delta \to 0$ yields

$$g'(t) = \lambda h(t)g(t) - 1.$$

Solve the preceding for $h(t)$ to get

$$h(t) = \frac{g'(t)}{\lambda g(t)} + \frac{1}{\lambda g(t)}.$$

Because the allocation function $h(t)$ needs to meet the constraint in (4.1), we have that

$$\begin{aligned} rc &= \int_0^c h(t)dt \\ &= \int_0^c \frac{g'(t)}{\lambda g(t)}dt + \int_0^c \frac{1}{\lambda g(t)}dt \\ &= \frac{1}{\lambda} \ln g(t)\Big|_0^c + \frac{1}{\lambda}\int_0^c \frac{1}{g(t)}dt \\ &= 0 + \frac{1}{\lambda}\int_0^c \frac{1}{g(t)}dt, \end{aligned}$$

where the last equality follows because $g(c) = g(0) \geq 1/\lambda$, because the patrol pattern of length c is repeated indefinitely, and that the expected time to detection is at least $1/\lambda$, which can be achieved only if the patroller allocates all her effort at this location all the time. Consequently, we have

$$\int_0^c \frac{1}{g(t)} dt = \lambda rc.$$

Using Lemma 4.1 and the preceding, we can find a lower bound for our objective function in (4.2) as follows:

$$\frac{\int_0^c g(t)dt}{c} \geq \frac{c}{\int_0^c \frac{1}{g(t)}dt} = \frac{c}{\lambda rc} = \frac{1}{\lambda r}. \tag{4.3}$$

In other words, $1/(\lambda r)$ is a lower bound for the expected time to detect a target at this location, and this lower bound is achieved if $g(t)$ is constant for $t \in [0, c]$. By taking $h(t) = r$ for $t \in [0, c]$, the instantaneous detection rate at the location stays at $r\lambda$ at all times, so $g(t) = 1/(\lambda r)$ for $t \in [0, c]$, which achieves the lower bound in (4.3). Consequently, the optimal policy is to take $h(t) = r$ for $t \in [0, c]$, which completes the proof. □

Theorem 4.3. *Consider the patrol problem with n locations with $d_{ij} = 0$, for all $i, j \in \{1, \ldots, n\}$. Let λ_i denote the instantaneous detection rate of the patroller at location i, for $i = 1, \ldots, n$. The optimal patrol policy is to allocate r_i fraction of time at location i continuously, where*

$$r_i = \frac{1/\lambda_i}{\sum_{j=1}^n 1/\lambda_j}. \tag{4.4}$$

The expected time to detect the target regardless of where the target arrives is equal to

$$\sum_{j=1}^n \frac{1}{\lambda_j}.$$

Proof. An arbitrary policy can be delineated by some $c > 0$ and a function $h_i(t) \in [0, 1]$ defined for $t \in [0, c]$ with the interpretation that the patroller allocates $h_i(t) \in [0, 1]$ fraction of her effort at location i, $i = 1, \ldots, n$, at time $t \in [0, c]$ such that

$$\sum_{i=1}^n h_i(t) = 1,$$

for $t \in [0, c]$, and repeats the same pattern of length c indefinitely.

Compute

$$s_i = \frac{\int_0^c h_i(t)dt}{c},$$

which represents the long-run fraction of effort allocated to location i, for $i = 1, \ldots, n$.

If the fraction of the patroller's effort allocated to location i is s_i, then according to Theorem 4.2, the expected time to detect a target at location i is at least $1/(\lambda_i s_i)$. Therefore, the value of the proposed policy is at least

$$\max_{i=1,\ldots,n} \frac{1}{\lambda_i s_i}.$$

By taking $s_i = r_i$ given in (4.4), for $i = 1, \ldots, n$, we minimise the preceding to $\sum_{j=1}^{n} 1/\lambda_j$, which is therefore a lower bound for the optimal value. Finally, because this lower bound can be achieved by taking $h_i(t) = r_i$ for $t \in [0, c]$ and $i = 1, \ldots, n$, as shown in Theorem 4.2, it follows that $\sum_{j=1}^{n} 1/\lambda_j$ is the optimal value and the policy just described is the optimal policy. $\square$

Theorem 4.3 shows that in the case of no travel times, it is optimal for the patroller to continuously allocate to a location the same fraction of effort that is inversely proportional to the detection rate at that location. If the travel time is nonzero, then at any given time the patroller is either moving between locations, or allocating 100% effort at one location. It then becomes much more difficult to determine the optimal patrol strategy.

4.3 A Simple Cycle

For a patrol problem with $d_{ij} > 0$ for all $i, j \in [n]$, a patrol pattern is an infinite ordered sequence of locations together with an infinite ordered sequence of search durations. We describe a patrol pattern as cyclic, or a cycle, if it repeats after a finite number of searches. A cycle can therefore be more succinctly represented by the truncated location and duration sequences prior to repetition. In this section we will consider a specific type of cycle called a simple cycle. We define a simple cycle as a cycle that visits each location exactly once. Thus, the truncated sequences for a simple cycle for an n

location patrol problem are of length n and the sequence of locations is a permutation of the set $[n]$.

For example, consider a two location patrol problem. If the patroller chooses to search the locations alternately and, for each $i \in \mathbb{N}$, she decides that her i^{th} search of each location will last for a number of minutes equal to the i^{th} digit of π then she is following a patrol pattern but not a cycle. If instead she decides to search each location for one minute on the i^{th} search if i is even and two minutes if i is odd then she is following a cycle but not a simple cycle. Finally, if she decides that every search of the first location will be for one minute and every search of the second location will be for two minutes then she is following a simple cycle.

In Section 4.3.1 we discuss some benefits of using simple cycles and provide some theoretical results to show that certain patrol patterns can be improved by making them simpler. In Section 4.3.2 we describe how to find the best possible simple cycle for a given problem.

4.3.1 Motivation for a Simple Cycle

In this section we discuss the benefits of using a simple cycle.

Aside from performance, there are several practical reasons to use a simple cycle. Firstly, by constraining ourselves to simple cycles we reduce the number of decision variables, which makes it easier to find the best cycle of the given type. Secondly, moving between locations in a fixed order and spending a specific amount of time at each location feels very natural. This makes simple cycles particularly easy for patrollers to implement and to explain to stakeholders.

While we expect simple cycles to perform well generally, they are particularly well suited to a certain class of patrol problem. An n location ring network is a set of n points arranged in a circle where it is only possible to move directly to adjacent locations. Thus for each $i, j \in [n]$ with $i < j$, the time taken to move from i to j is

$$\min\left(\sum_{k=i}^{j-1} d_{k,k+1}, \sum_{k=1}^{n-1} d_{k,k+1} + d_{n,1} - \sum_{k=i}^{j-1} d_{k,k+1}\right).$$

Ring networks can be used for patrol problems that involve patrolling a border or some other circular route. In these cases, moving directly between two locations that are not adjacent may not be practical or even possible. For example, a guard that has been instructed to protect the external entrances to a facility may not be allowed inside the facility they are protecting and therefore to reach an entrance on the far side of the facility they can either travel clockwise or anticlockwise around the perimeter. It would then seem wasteful not to check any other entrances that they pass.

We will now present a few results that support the idea of using a simple cycle. Consider a patrol problem with n locations, for some $n \in \mathbb{N}$. Recall that the patroller's objective is to minimise the maximum expected time to discovery across all locations. If a patrol pattern omits one or more locations then it is impossible to discover any attackers at these locations. Even if a patrol pattern visits a location for some finite amount of time then never visits there again there is a possibility that an attacker there will not be discovered and so the expected time to discovery for that location is infinite. It is therefore clear that an optimal patrol pattern must visit every location infinitely many times.

Now, consider a cycle that visits a location $i \in [n]$ twice for different lengths of time. Since a location is being visited more than once, this is not a simple cycle. Our first result demonstrates that the expected time to discover an attacker at location i can be reduced by replacing the duration of both searches by their average.

Proposition 4.4. *Any cycle that visits a location twice for different durations can be improved by replacing the duration of each search by the average of both.*

Proof. Consider a patrol problem and a corresponding cycle that visits location $i \in [n]$ twice for different durations. Suppose that the total duration of the cycle is c and that the durations of the first and second searches of location i are x_1 and x_2 respectively. Let $s = x_1 + x_2$. Also let y_1 be the amount of time the patroller spends away from location i after the first search there but before she returns. Similarly let y_2 be the amount of time she spends away after the second search of location i. These durations comprise travel times as well as some time spent searching other locations.

Consider the expected number of attackers at location i throughout the cycle at steady state. This scenario can be thought of as a queuing system with:

- A constant arrival rate of θ which represents attackers arriving at location i.
- Infinitely many servers, each with a constant service rate $\lambda = \lambda_i$, that are only active while the searcher is in location i. The identifier i for the detection/service rate is dropped for ease of notation.

The expected time a single attacker (or customer) spends in the system comprises the expected service time ($1/\lambda$) and the expected idle time.

Let b denote the number of customers in the system at the start of the cycle in steady state. Figure 4.1 shows the expected number of customers in the system throughout the cycle. The white areas show when the servers are active and the grey areas show when they are idle.

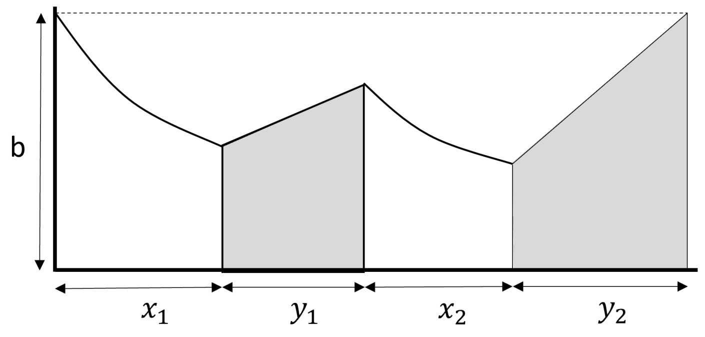


Figure 4.1: Number of customers over time in steady state

During steady state the number of customers in the system is the same at the start and end of the cycle. Thus

$$b = \left(be^{-\lambda x_1} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_1}) + \theta y_1 \right) e^{-\lambda x_2} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_2}) + \theta y_2$$

Here the first term gives the number of customers that are either in the system at the start of the current cycle or arrive before the start of the second search and are still present at the end of the cycle. The second and third terms give the number of customers that arrive during and after the second search and are still present at the end of the cycle. Collecting all b terms on the left hand side we get

$$(1 - e^{-\lambda s})b = \left(\frac{\theta}{\lambda}(1 - e^{-\lambda x_1}) + \theta y_1 \right) e^{-\lambda x_2} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_2}) + \theta y_2$$

An expression for b can then by recovered by dividing both sides by $1 - e^{-\lambda s}$.

$$\begin{aligned} b &= \theta \cdot \left(\frac{\frac{1}{\lambda}e^{-\lambda x_2} - \frac{1}{\lambda}e^{-\lambda s} + y_1 e^{-\lambda x_2} + \frac{1}{\lambda} - \frac{1}{\lambda}e^{-\lambda x_2} + y_2}{1 - e^{-\lambda s}} \right) \\ &= \theta \cdot \left(\frac{-\frac{1}{\lambda}e^{-\lambda s} + y_1 e^{-\lambda x_2} + \frac{1}{\lambda} + y_2}{1 - e^{-\lambda s}} \right) \\ &= \theta \cdot \left(\frac{1}{\lambda} + \frac{y_1 e^{-\lambda x_2} + y_2}{1 - e^{-\lambda s}} \right) \end{aligned}$$

Write L for the long-run average number of customers in the system when the patroller is on site (x_1 and x_2). The rate at which customers depart when the patroller is on site is $L\lambda$, so the average number of customers that depart the system in each cycle is $sL\lambda$. The long-run average number of customers that arrive in each cycle is $c\theta$. Setting $sL\lambda = c\theta$ yields $L = c\theta/s\lambda$. The area of the two white segments combined is therefore

$$Ls = \frac{c\theta}{\lambda}$$

The area of the first shaded region is

$$\left(be^{-\lambda x_1} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_1}) + \frac{\theta y_1}{2} \right) y_1$$

The area of the second shaded region is

$$\left(\left(be^{-\lambda x_1} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_1}) + \theta y_1 \right) e^{-\lambda x_2} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_2}) + \frac{\theta y_2}{2} \right) y_2$$

Adding these terms and dividing by c gives us the long-run average number of customers in the system. We start by adding them together and rearranging.

$$\begin{aligned} &\frac{c\theta}{\lambda} + b(y_1 e^{-\lambda x_1} + y_2 e^{-\lambda s}) + \theta \left[\frac{1}{\lambda} \left[y_1(1 - e^{-\lambda x_1}) + y_2(1 - e^{-\lambda s}) \right] + y_1 y_2 e^{-\lambda x_2} + \frac{y_1^2 + y_2^2}{2} \right] \\ &= \theta \left[\frac{c}{\lambda} + \frac{y_1 + y_2}{\lambda} + \frac{(y_1 e^{-\lambda x_2} + y_2)(y_1 e^{-\lambda x_1} + y_2 e^{-\lambda s})}{1 - e^{-\lambda s}} + y_1 y_2 e^{-\lambda x_2} + \frac{y_1^2 + y_2^2}{2} \right] \\ &= \theta \left[\frac{c}{\lambda} + \frac{y_1 + y_2}{\lambda} + \frac{y_1^2 e^{-\lambda s} + y_1 y_2 (e^{-\lambda(x_2 + s)} + e^{-\lambda x_1}) + y_2^2 e^{-\lambda s}}{1 - e^{-\lambda s}} + y_1 y_2 e^{-\lambda x_2} + \frac{y_1^2 + y_2^2}{2} \right] \\ &= \theta \left[\frac{c}{\lambda} + \frac{y_1 + y_2}{\lambda} + \frac{y_1^2 + y_2^2 + y_1 y_2 (e^{-\lambda(x_2 + s)} + e^{-\lambda x_1})}{1 - e^{-\lambda s}} + y_1 y_2 e^{-\lambda x_2} - \frac{y_1^2 + y_2^2}{2} \right] \\ &= \theta \left[\frac{c}{\lambda} + \frac{y_1 + y_2}{\lambda} + \frac{y_1^2 + y_2^2 + y_1 y_2 (e^{-\lambda x_2} + e^{-\lambda x_1})}{1 - e^{-\lambda s}} - \frac{y_1^2 + y_2^2}{2} \right] \end{aligned}$$

Thus, the long-run average number of customers in the system is:

$$\frac{\theta}{\lambda}+\frac{\theta}{c}\left[\frac{y_1+y_2}{\lambda}+\frac{y_1^2+y_2^2+y_1y_2(e^{-\lambda x_2}+e^{-\lambda x_1})}{1-e^{-\lambda s}}-\frac{y_1^2+y_2^2}{2}\right] \tag{4.5}$$

To find the values of x_1 and x_2 that minimise this expression we need only consider terms that contain x_1 and x_2 and so it is equivalent to minimise

$$\theta\left[\frac{y_1y_2(e^{-\lambda x_2}+e^{-\lambda x_1})}{1-e^{-\lambda s}}\right].$$

Thus the long-run average number of customers in the system is minimised when

$$e^{-\lambda x_1}+e^{-\lambda x_2}$$

is minimised.

We can then conclude that the expected discovery time for an attacker at location i is minimised by setting $x_1 = x_2 = s/2$. Thus, for each location it is optimal to set $x_1 = x_2$ and $y_1 = y_2$, which can be achieved if every location has $x_1 = x_2$. □

It is intuitive that this result should hold more generally. The following conjecture considers a cycle that visits each location an arbitrary number of times.

Conjecture 4.5. *Any cycle that visits a location more than once for different durations can be improved by replacing the duration of each search by the average of them all.*

In Appendix 4.7.1 we seek to generalise the proof of Proposition 4.4 to a cycle which visits location i, m times for some $m \in \mathbb{N}$ with $m \geq 2$, in order to prove the above conjecture. For this more general case we show that the solution $x_1 = \cdots = x_m = s/m$, $y_1 = \cdots = y_m = (c-s)/m$ is a stationary point but have thus far been unable to show that this corresponds to a unique minimum.

4.3.2 Optimising a Simple Cycle

In this section we consider how to find the best simple cycle for a given problem. Suppose we are faced with an n location patrol problem of the type described in Section 4.1 and that we wish to follow a simple cycle. Let Σ_n denote the set of permutations of the set $[n]$. A simple cycle is determined by an ordering $\sigma \in \Sigma_n$ of the n locations together with a vector $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in (\mathbb{R}^+)^n$ of search durations for each location, where for each $i \in [n]$, x_i is the duration of the search at location i.

Additionally, for each $i \in [n]$, write $f_i(\sigma, \mathbf{x})$ for the expected time for a patroller to discover an attacker at location i under the simple cycle determined by $(\sigma, \mathbf{x})$. The patroller's challenge is to choose a simple cycle $(\sigma, \mathbf{x})$ to minimise $f(\sigma, \mathbf{x})$ where

$$f(\sigma, \mathbf{x}) = \max_{i \in [n]} \{f_i(\sigma, \mathbf{x})\}.$$

Hence, the patroller's problem is given by

$$\min_{\sigma, \mathbf{x} \in \Sigma_n \times (\mathbb{R}^+)^n} f(\sigma, \mathbf{x}) = \min_{\sigma, \mathbf{x} \in \Sigma_n \times (\mathbb{R}^+)^n} \max_{i \in [n]} \{f_i(\sigma, \mathbf{x})\}.$$

To address this problem we must first derive a formula for $f_i(\sigma, \mathbf{x})$ by evaluating the expected time to discovery for a particular attacker at location i for some $i \in [n]$ when the patroller is following simple cycle $(\sigma, \mathbf{x})$. Write $c(\sigma, \mathbf{x})$ for the total length of the cycle. Thus

$$c(\sigma, \mathbf{x}) = \sum_{j=1}^{n} (x_j + d_{\sigma(j),\sigma(j+1)}),$$

using the convention that $d_{\sigma(n),\sigma(n+1)} = d_{\sigma(n),\sigma(1)}$. In the following discussion we shall sometimes write

$$c(\sigma, \mathbf{x}) = C(\mathbf{x}) + D(\sigma)$$

where

$$C(\mathbf{x}) = \sum_{j=1}^{n} x_j \quad \text{and} \quad D(\sigma) = \sum_{j=1}^{n} \left(d_{\sigma(j),\sigma(j+1)}\right)$$

to distinguish between the search time and travel time components of the total cycle time.

Suppose that the attacker is sent to attack location i for some $i \in [n]$. The expected time to discover this attacker depends on where the patroller is at the time of arrival. Since attackers arrive according to a Poisson process, attackers are equally likely to arrive at any point during the patrol pattern. Let Z be a uniform random variable over $(0, c(\sigma, \mathbf{x}))$ and write $\mathbb{E}[T|Z = z]$ for the expected search time conditional on the attacker arriving z units of time into the patroller's current cycle. Thus,

$$f_i(\sigma, \mathbf{x}) = \int_0^{c(\sigma,\mathbf{x})} \frac{\mathbb{E}[T|Z=z]}{c(\sigma,\mathbf{x})}\, \mathrm{d}z = \frac{1}{c(\sigma,\mathbf{x})} \int_0^{c(\sigma,\mathbf{x})} \mathbb{E}[T|Z=z]\, \mathrm{d}z \tag{4.6}$$

To find an expression for $\mathbb{E}[T|Z=z]$ we consider two scenarios. If the patroller is at location i when the attacker arrives then we have $z \in [0, x_i)$. Otherwise, if the patroller is travelling or is searching another location, then $z \in [x_i, c(\sigma, \mathbf{x}))$.

Firstly, consider $z \in [0, x_i)$. The contribution of the remainder of the current search is:

$$\int_0^{x_i - z} \lambda_i y e^{-\lambda_i y} dy = \frac{1}{\lambda_i}\left(1 - e^{-\lambda(x_i - z)}(\lambda_i(x_i - z) + 1)\right)$$

No contribution is made while the patroller is away. The contribution of the first z time units of the next search of location i is given by

$$\begin{aligned}\int_{x_i - z}^{x_i} (c(\sigma,\mathbf{x}) - x_i + y)\lambda_i e^{-\lambda_i y}\, \mathrm{d}y = \frac{1}{\lambda_i}\big(&e^{-\lambda_i(x_i - z)}(\lambda_i(c(\sigma,\mathbf{x}) - z) + 1) \\ &- e^{-\lambda_i x_i}(\lambda_i c(\sigma,\mathbf{x}) + 1)\big)\end{aligned}$$

Thus the contribution of the first $c(\sigma, \mathbf{x})$ time units is

$$\frac{1}{\lambda_i}\left(1 - e^{-\lambda_i x}(\lambda_i c(\sigma,\mathbf{x}) + 1)\right) + e^{-\lambda_i(x_i - z)}(c(\sigma,\mathbf{x}) - x_i).$$

Using recursion we have that

$$\begin{aligned}\mathbb{E}[T|Z=z] &= \frac{1}{\lambda_i}\left(1 - e^{-\lambda_i x_i}(\lambda_i c(\sigma,\mathbf{x}) + 1)\right) \\ &\quad + e^{-\lambda_i(x_i - z)}(c(\sigma,\mathbf{x}) - x_i) + (c(\sigma,\mathbf{x}) + \mathbb{E}[T|Z=z])e^{-\lambda_i x_i} \\ &= \frac{1}{\lambda_i}\left(1 - e^{-\lambda_i x_i}\right) + e^{-\lambda_i(x_i - z)}(c(\sigma,\mathbf{x}) - x_i) + \mathbb{E}[T|Z=z]e^{-\lambda_i x_i}\end{aligned}$$

Thus

$$\mathbb{E}[T|Z=z](1 - e^{-\lambda_i x_i}) = \frac{1}{\lambda_i}\left(1 - e^{-\lambda_i x_i}\right) + (c(\sigma,\mathbf{x}) - x_i)e^{-\lambda_i(x_i - z)}$$

So we have that

$$\mathbb{E}[T|Z=z] = \frac{1}{\lambda_i} + \frac{(c(\sigma,\mathbf{x}) - x_i)e^{-\lambda_i(x_i - z)}}{1 - e^{-\lambda_i x_i}} \tag{4.7}$$

The total amount of time between the attacker's arrival and detection consists of two components: (a) the amount of time the patroller spends at the attacker's location, and (b) the amount of time the patroller spends away from the attacker's location. In (4.7), the first term $1/\lambda_i$ is the expected value of an exponential random variable which corresponds to (a), and the second term corresponds to (b). Component (b) would be zero if the patroller detects the attacker in the first $x_i - z$ time units. If the patroller does not detect the attacker in the first $x_i - z$ time units, which occurs with probability $e^{-\lambda_1(x_i-z)}$, then the number of times the patroller needs to go through an "off cycle" $c(\sigma, \mathbf{x}) - x_i$ before detecting the attacker follows a geometric distribution with success probability $1 - e^{-\lambda_i x_i}$, whose expected value is $1/(1 - e^{-\lambda_i x_i})$. Putting these together produces component (b) in (4.7).

Integrating over the values of z between 0 and x_i gives us:

$$\begin{aligned}\int_0^{x_i} \left(\frac{1}{\lambda_i} + \frac{(c(\sigma,\mathbf{x}) - x_i)e^{-\lambda_i(x_i-z)}}{1-e^{-\lambda_i x_i}}\right) \mathrm{d}z &= \frac{x_i}{\lambda_i} + \frac{e^{-\lambda_i x_i}(c(\sigma,\mathbf{x}) - x_i)}{1-e^{-\lambda_i x_i}} \int_0^{x_i} e^{\lambda z} dz \\ &= \frac{x_i}{\lambda_i} + \frac{e^{-\lambda_i x_i}(c(\sigma,\mathbf{x}) - x_i)}{1-e^{-\lambda_i x_i}} \left(\frac{e^{\lambda_i x_i} - 1}{\lambda_i}\right)\end{aligned}$$

Thus,

$$\int_0^{x_i} \mathbb{E}[T|Z = z]\mathrm{d}z = \frac{c(\sigma,\mathbf{x})}{\lambda_i} \tag{4.8}$$

Next we consider $z \in [x_i, c(\sigma, \mathbf{x}))$. Since the attacker is in location i there is no chance the patroller will discover them until she returns to location i. Thus,

$$\mathbb{E}[T|Z = z] = c(\sigma,\mathbf{x}) - z + \mathbb{E}[T|Z = 0].$$

Setting $z = 0$ in (4.7) we have

$$\mathbb{E}[T|Z = 0] = \frac{1}{\lambda_i} + \frac{e^{-\lambda_i x_i}(c(\sigma,\mathbf{x}) - x_i)}{1-e^{-\lambda_i x_i}}$$

Thus

$$\mathbb{E}[T|Z = z] = c(\sigma,\mathbf{x}) - z + \frac{1}{\lambda_i} + \frac{e^{-\lambda_i x_i}(c(\sigma,\mathbf{x}) - x_i)}{1-e^{-\lambda_i x_i}}$$

Integrating over the values of z between x_i and $c(\sigma, \mathbf{x})$ gives us:

$$\int_{x_i}^{c(\sigma,\mathbf{x})} \left(c(\sigma,\mathbf{x}) - z + \frac{1}{\lambda_i} + \frac{e^{-\lambda_i x_i}(c(\sigma,\mathbf{x}) - x_i)}{1-e^{-\lambda_i x_i}}\right) dz$$

$$= \left(c(\sigma,\mathbf{x}) + \frac{1}{\lambda_i} + \frac{e^{-\lambda_i x_i}(c(\sigma,\mathbf{x}) - x_i)}{1-e^{-\lambda_i x_i}}\right)(c(\sigma,\mathbf{x}) - x_i) - \left[\frac{z^2}{2}\right]_{x_i}^{c(\sigma,\mathbf{x})}$$

Thus,

$$\int_{x_i}^{c(\sigma,\mathbf{x})} \mathbb{E}[T|Z=z]dz = (c(\sigma,\mathbf{x})-x_i)\left(\frac{c(\sigma,\mathbf{x})-x_i}{2}+\frac{1}{\lambda_i}+\frac{e^{-\lambda_i x_i}(c(\sigma,\mathbf{x})-x_i)}{1-e^{-\lambda_i x_i}}\right) \quad (4.9)$$

Therefore, by adding (4.8) and (4.9) and dividing by $c(\sigma,\mathbf{x})$ we have

$$f_i(\sigma,\mathbf{x}) = \frac{1}{\lambda_i}+\frac{c(\sigma,\mathbf{x})-x_i}{c(\sigma,\mathbf{x})}\left(\frac{c(\sigma,\mathbf{x})-x_i}{2}+\frac{1}{\lambda_i}+\frac{e^{-\lambda_i x_i}(c(\sigma,\mathbf{x})-x_i)}{1-e^{-\lambda x_i}}\right)$$

or equivalently

$$f_i(\sigma,\mathbf{x}) = \frac{1}{\lambda_i}+\frac{c(\sigma,\mathbf{x})-x_i}{c(\sigma,\mathbf{x})}\left(\frac{1}{\lambda_i}-\frac{c(\sigma,\mathbf{x})-x_i}{2}+\frac{c(\sigma,\mathbf{x})-x_i}{1-e^{-\lambda_i x_i}}\right) \quad (4.10)$$

This equation gives the expected time to discover an attacker at location i while following a simple cycle of length $c(\sigma,\mathbf{x})$ that searches location i for x_i time units on each visit and can be used to compute the expected time to discovery for attackers at each location. Note that the total cycle time $c(\sigma,\mathbf{x})$ depends on the choice of x_i for each location as well as the total travel time of the chosen ordering σ. The patroller wishes to minimise the maximum expected time to discovery across all locations.

The following result demonstrates that for any duration vector $\mathbf{x}$, the best choice of ordering σ is a minimum Hamiltonian cycle.

Proposition 4.6. *The patroller's problem can always be solved by taking optimising permutation σ^* to be any Hamiltonian cycle that satisfies*

$$D(\sigma^*) = \min_{\sigma\in\Sigma_n} D(\sigma)$$

Proof. Consider the quantity $f_i(\sigma,\mathbf{x})$ for some fixed $\mathbf{x}$. From (4.10) we may write

$$\mathbb{E}[T] = \frac{1}{\lambda_i}+\frac{C(\mathbf{x})+D(\sigma)-x_i}{C(\mathbf{x})+D(\sigma)}\left(\frac{1}{\lambda_i}+(C(\mathbf{x})+D(\sigma)-x_i)A(\mathbf{x})\right)$$

for positive constant $A(\mathbf{x})$. It is straightforward to show that for any $\sigma_1,\sigma_2\in\Sigma_n$,

$$D(\sigma_1)\le D(\sigma_2) \implies f_{\sigma_1 i}(\mathbf{x})\le f_{\sigma_2 i}(\mathbf{x}), \forall i,\mathbf{x}.$$

The result now follows easily. □

It remains to find the optimal choice for the duration vector, $\mathbf{x}$. This choice will be discussed further in subsequent sections.

It is useful to briefly examine the behaviour of (4.10) in some limiting cases. Firstly, consider the behaviour of (4.10) as we vary the detection rates. It is clear that as a particular detection rate tends to zero the expected time for the patroller to discover an attacker hidden in the corresponding location diverges to infinity. This observation makes sense since lowering the detection rate at a location increases the expected time to discover an attacker hidden there. Next, consider what happens as the detection rate at a particular location i diverges to infinity. Here we would expect the expected search time to fall and for the optimal value of x_i to tend to zero. For a fixed positive value of x_i, $e^{-\lambda_i x_i} \to 0$ as $\lambda_i \to \infty$. Therefore the expected search time would tend to

$$\frac{(c(\sigma,\mathbf{x}) - x_i)^2}{2c(\sigma,\mathbf{x})}.$$

Thus, as the detection rate for a particular location increases the patroller should spend less time searching there. As x_i tends to 0, the preceding equation converges to $c(\sigma,\mathbf{x})/2$, which is the expected time an attacker spends at the location until the patroller returns to the location for the first time.

It is also clear that if the cycle time $c(\sigma,\mathbf{x})$ (or, by extension, any of the travel times involved in the cycle) tend to infinity then the expected time to find an attacker also diverges to infinity. Again this observation makes sense as the patroller would be spending an increasingly high proportion of her time travelling rather than searching.

4.3.3 The Case of Homogeneous Locations

In this section we seek to find the best simple cycle $(\sigma,\mathbf{x})$, as described in Section 4.3, for patrol problems with homogeneous locations. We describe the locations of a patrol problem as homogeneous if the detection rate is the same at all locations. Throughout this section we will use λ to denote the common detection rate.

The primary benefit of considering patrol problems with homogeneous locations, and the reason for the inclusion of this section, is that since all detection rates are the same it is intuitive that each location should be searched for the same amount of time.

Thus, the problem of finding the best duration vector $\mathbf{x}$ reduces to a single dimension. This intuition is formalised in the following result.

Proposition 4.7. *A best simple cycle for a homogeneous patrol problem must visit each location for the same duration.*

Proof. Consider a simple cycle $(\sigma, \mathbf{x})$ for a homogeneous patrol problem. Proposition 4.6 tells us that any permutation $\sigma \in \Sigma_n$ that satisfies $D(\sigma^*) = \min_{\sigma \in \Sigma_n} D(\sigma)$ can be used to obtain a best simple cycle. Additionally, it is clear that the total cycle duration is independent of the choice of permutation. We can therefore write

$$c(\sigma^*, \mathbf{x}) = D(\sigma^*) + C(\mathbf{x}).$$

By examining (4.10) we can see that for a fixed $C(\mathbf{x})$ and for each $i \in [n]$, $f_i(\sigma, \mathbf{x})$ increases as x_i decreases. Thus $f(\sigma, \mathbf{x})$ is minimised by taking $x_1 = \cdots = x_n$. $\square$

Now, consider a homogeneous patrol problem with n locations for some $n \in \mathbb{N}$ and a common detection rate λ. Propositions 4.6 and 4.7 tell us that the best simple cycle for this problem should visit the locations in an order σ^* that constitutes a minimum Hamiltonian cycle and that each location should be searched for x^* units of time for some $x^* \in \mathbb{R}^+$.

Once a minimum Hamiltonian cycle has been found it remains to find x^*. This value can be found by applying (4.10) to find the expected time to discovery when following this cycle. To find the optimal search duration we need to find the value of x that minimises (4.10) with $c(\sigma, \mathbf{x}) = D(\sigma^*) + nx$. Thus we need to find

$$x^* = \operatorname*{argmin}_{x>0} \left(\frac{D(\sigma^*) + nx - x}{D(\sigma^*) + nx} \left(\frac{1}{\lambda} - \frac{D(\sigma^*) + nx - x}{2} + \frac{D(\sigma^*) + nx - x}{1 - e^{-\lambda x}} \right) \right). \tag{4.11}$$

For $n = 2$, (4.11) simplifies to

$$x^* = \operatorname*{argmin}_{x>0} \left(\frac{D(\sigma^*) + x}{D(\sigma^*) + 2x} \left(\frac{1}{\lambda} - \frac{D(\sigma^*) + x}{2} + \frac{D(\sigma^*) + x}{1 - e^{-\lambda x}} \right) \right).$$

Finding a closed form for x^* is challenging so using a numerical method, such as the ternary search algorithm that we discuss in Section 4.5.2, is necessary.

Table 4.1 shows the best search duration x^*, obtained numerically and rounded to three decimal places, for different homogeneous two location patrol problems with a common travel time d between each distinct pair of locations.

	$d = 1$	$d = 2$	$d = 3$
$\lambda = 1$	2.709	3.323	3.731
$\lambda = 2$	1.661	2.021	2.257
$\lambda = 3$	1.243	1.505	1.674

Table 4.1: Best search duration for a 2 location patrol problem

We can see that as the common detection rate λ increases the best search duration decreases. This observation makes sense as each location can be searched for less time while still achieving the same probability of discovering an attacker that is already there. We can also see that as d increases the best search duration also increases. This observation also makes sense as if the time between consecutive visits to each location increases then the searcher is more incentivised to stay at the location for longer in order to reduce the probability that an attacker is overlooked. It is also worth noting that despite the optimal search duration increasing with travel time, the percentage of time spent searching each location actually falls. For $\lambda = 1$ the patroller spends approximately 36.5% of her time searching each location when $d = 1$, 31.2% of her time searching each location when $d = 2$ and 27.7% of her time searching each location when $d = 3$.

Figure 4.2 shows how the expected time for the patroller to discover a particular attacker varies with search duration x at each location for a 2 location homogeneous patrol problem with detection rate $\lambda = 1$ and where all travel times are set to $d = 1$.

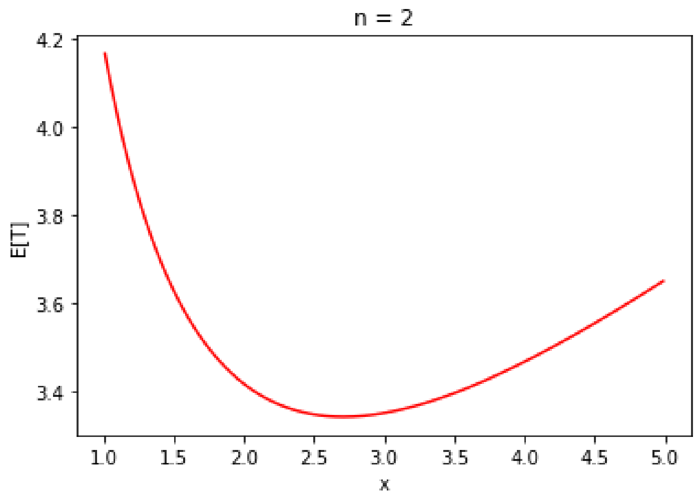


Figure 4.2: Search time vs expected time to discovery

It appears that the expected time to discover an attacker is unimodal in x, the search duration.

Figure 4.3 shows how the absolute sub-optimality of the expected time to detection varies with time spent searching each location for patrol problems with 2, 3 and 4 locations with detection rate $\lambda = 1$ and where all travel times are set to $d = 1$.

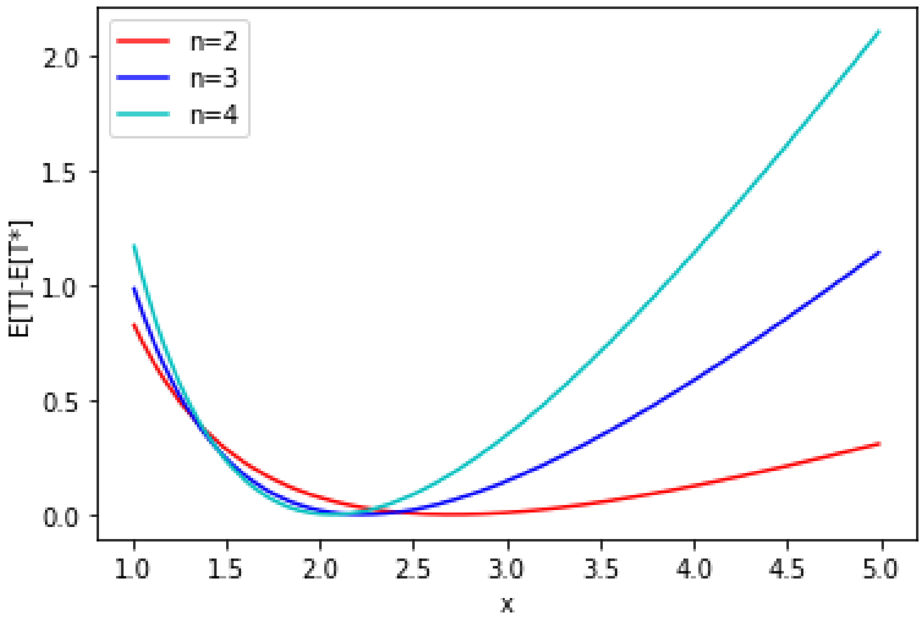


Figure 4.3: Search time vs absolute sub-optimality of the expected time to detection

We can see that as the number of locations increases the optimal amount of time to spend searching each location decreases. This observation makes sense since the patroller has more locations to visit and shortening her stay at one location reduces the amount of time she spends away from each of the others.

4.3.4 The Case of Heterogeneous Locations

The results of the previous section rely on the homogeneity of detection rates and the use of Proposition 4.7 to reduce the dimensionality of the problem. In this section we consider the broader class of heterogeneous patrol problems and derive a formula for the expected time to discover an attacker when following a given simple cycle.

For simplicity we initially consider two location heterogeneous problems. The ordering $\sigma \in \Sigma_2$ is arbitrary since both orderings simply involve alternating between the two locations. The total duration of the simple cycle $(\sigma, \mathbf{x})$ is therefore $x_1 + d_{12} + x_2 + d_{21}$ regardless of the choice of $\sigma \in \Sigma_2$. For ease of notation, write $\tau = d_{12} + d_{21}$. To determine the best values for the search durations x_1 and x_2 (denoted as x_1^* and x_2^* respectively) we evaluate the time to discover a particular attacker under this cycle. Thus,

$$(x_1^*, x_2^*) = \underset{(x_1,x_2)\in(\mathbb{R}^+)^2}{\operatorname{argmin}} \left(\max\{f_1(\sigma, \mathbf{x}), f_2(\sigma, \mathbf{x})\}\right).$$

For each $i \in \{1, 2\}$, we can find $f_i(\sigma, \mathbf{x})$ by applying (4.10) with $c(\sigma, \mathbf{x}) = x_1 + x_2 + \tau$, which yields

$$f_1(x_1, x_2) = \frac{1}{\lambda_1} + \frac{\tau + x_2}{\tau + x_1 + x_2}\left(\frac{1}{\lambda_1} - \frac{\tau + x_2}{2} + \frac{\tau + x_2}{1 - e^{-\lambda_1 x_1}}\right)$$

and

$$f_2(x_1, x_2) = \frac{1}{\lambda_2} + \frac{\tau + x_1}{\tau + x_1 + x_2}\left(\frac{1}{\lambda_2} - \frac{\tau + x_1}{2} + \frac{\tau + x_1}{1 - e^{-\lambda_2 x_2}}\right).$$

Now, we seek to generalise this result. Consider an n location heterogeneous problem for some $n \in \mathbb{N}$. Here, the ordering of the locations does matter. However, from Proposition 4.6 we know that a best simple cycle can be obtained using any minimum Hamilton cycle for the ordering. Without loss of generality, let σ be the identity ordering and assume that this permutation is a minimum Hamiltonian cycle. To determine the best optimal values for the search durations x_i (denoted as x_i^*) we evaluate the time to discover a particular attacker under this cycle. Thus,

$$(x_1^*, \cdots, x_n^*) = \underset{(x_1,\cdots,x_n)\in(\mathbb{R}^+)^n}{\operatorname{argmin}} \left(\max_{i\in[n]} \{f_i(\sigma, \mathbf{x})\} \right) \tag{4.12}$$

For each $i \in \{1, \cdots, n\}$, we can find $f_i(\sigma, \mathbf{x})$ by applying (4.10) with

$$c(\sigma, \mathbf{x}) = \sum_{j=1}^{n} \left(x_j + d_{\sigma(j),\sigma(j+1)} \right),$$

using the convention that $d_{\sigma(n),\sigma(n+1)} = d_{\sigma(n),\sigma(1)}$, which yields

$$f_i(\sigma, \mathbf{x}) = \frac{1}{\lambda_i} + \frac{c(\sigma, \mathbf{x}) - x_i}{c(\sigma, \mathbf{x})} \left(\frac{1}{\lambda_i} - \frac{c(\sigma, \mathbf{x}) - x_i}{2} + \frac{c(\sigma, \mathbf{x}) - x_i}{1 - e^{-\lambda_i x_i}} \right). \tag{4.13}$$

Like with the homogeneous case, finding a closed from for the optimal vector $\mathbf{x}$ of search durations is challenging. A numerical approach for finding this vector is discussed in Section 4.5.2.

4.4 A Sweep Cycle

In Section 4.3 we introduced the notion of a simple cycle and provided some examples where we would expect a cycle of this type to perform particularly well. In this section we consider a different cycle type, motivated by a different class of patrol problems.

4.4.1 Motivation of a Sweep Cycle

Consider a set of n locations that are positioned along some sort of road or path. These locations could be carriages in a train, service stations along a motorway or roads intersecting the border between two neighbouring counties. In each of these cases there is a natural ordering given by the geography of the problem with location 1 at one end of the path and location n at the other. In many of these cases it can be impractical to move directly between two locations that are not adjacent. Thus, if locations are labelled from one end of the path to the other, then for any distinct pair $i, j \in [n]$ with $i < j$, $d_{ij} = \sum_{k=i}^{j-1} d_{k,k+1}$. We describe such problems as line networks.

If the patroller were following a simple cycle for a patrol problem on a line network she would start at one end of the line and then move along the line to the other end, visiting each location along the way. She would then travel back in the reverse direction but this time ignoring all locations (i.e. not searching them) until she arrives back at her starting point. It is worth noting that if the patroller were to stop at any of the locations on the reverse journey there would be no additional travel time. Building on this observation, one natural change that the patroller could make to her patrol pattern is to split up the search time at each of the middle locations so that she searches for part of the planned time on her first visit and the remaining time on the way back. It is not immediately obvious how this time should be divided, or whether allowing the search to be broken up into two parts would mean that the total time spent searching each location in a cycle should change. We therefore introduce the sweeping cycle.

Consider an n location heterogeneous patrol problem for some $n \in \mathbb{N}$, along with some ordering $\sigma \in \Sigma_n$ and a vector $\mathbf{x} \in (\mathbb{R}^+)^{2n}$ of search durations. Note that the vector of search durations here is twice as long as for a simple cycle. The sweep cycle characterised by σ and $\mathbf{x}$ is denoted as $(\sigma_{\leftrightarrow}, \mathbf{x})$ and is defined as follows. For each $i \in [n]$, x_i is the duration of the first search of location i and x_{2n+1-i} is the duration of the second search of location i. The patroller initially searches each of the locations in the order given by σ, starting with location $\sigma(1)$ and finishing with a search of location $\sigma(n)$. Hitherto, this cycle is the same as a simple cycle. For a simple cycle the patroller would then travel back to location $\sigma(1)$ and repeat the process. However, for a sweep cycle, the patroller instead immediately searches location $\sigma(n)$ again and then carries out the reverse process, searching the locations in the reverse ordering (ie, location $\sigma(n)$, then location $\sigma(n-1)$ and so on). The patroller finishes the first full sweep upon completing the second search of location $\sigma(1)$.

For example, if the patroller is following the sweep cycle

$$((1,2,3)_{\leftrightarrow}, (0.5, 0.6, 0.7, 0.8, 0.9, 1)),$$

then she would start by searching location 1 for 0.5 units of time, then location 2 for 0.6 units of time, then location 3 for 0.7 units of time. She would then continue to search location 3 for a further 0.8 units of time before travelling back to location 2 to search there for a further 0.9 units of time and finally back to location 1 to search for 1 unit of time.

4.4.2 Optimising a Sweep Cycle

In this section we consider how to find the best sweep cycle.

Our first result for the sweep cycle shows us that for each $i \in [n]$ the best sweep pattern searches location i for the same duration during both visits. This result is not obvious as the time between visits may not be same on the left and right sweep.

Theorem 4.8. *Consider an n location patrol problem, for some $n \in \mathbb{N}$. For each $i \in [n]$, the duration of both searches of location i when following the best sweep cycle are of the same duration.*

Proof. A sweep cycle is a type of cycle where each location is visited twice and so Proposition 4.4 tells is that each of these searches must be of the same duration. □

This is useful as it allows us to half the number of decision variables from $2n$ to n which will make finding the vector of best search durations much easier.

Since the sweep cycle was motivated by line networks, it is useful to consider how they compare to simple cycles in this specific context.

Theorem 4.9. *In a line network, the best sweep cycle outperforms the best simple cycle.*

Proof. Consider a line network with $n \geq 3$ locations labelled in order from one end of the line to the other. Proposition 4.6 tells us that a best simple cycle can be found using the the natural ordering, $1, \ldots, n$. Let $\mathbf{x} = \{x_1, \ldots, x_n\}$ be the vector of durations for a best simple cycle using this ordering. Now consider a sweep cycle using the same ordering where for each $i \in [n]$, each search of location i is of length $y_i = x_i/2$. Since each location is searched twice, this sweep cycle has the same total length as the length of our best simple cycle. For locations 1 and n, the two searches take place one after the other so it is clear that $f_i(\sigma, \mathbf{x}) = f_i(\sigma_{\leftrightarrow}, \mathbf{y})$ for $i = 1, n$. It follows from Proposition 4.4 that $f_i(\sigma, \mathbf{x}) > f_i(\sigma_{\leftrightarrow}, \mathbf{y})$ for $i = 2, \ldots, n-1$. Since the patroller wishes to minimise the maximum time to discovery across all locations, this sweep policy performs equally to the best simple cycle. Now, arbitrarily select a location $j \in \{2, \ldots, n-1\}$ and set $y_j = (x_j - \varepsilon)/2$ for some small $\varepsilon > 0$. This change reduces the total cycle time for the sweep cycle by 2ε. Thus for all $i \neq j$, $f_i(\sigma, \mathbf{x}) > f_i(\sigma_{\leftrightarrow}, \mathbf{y})$ and as long as ε is chosen to be sufficiently small then $f_j(\sigma, \mathbf{x}) > f_j(\sigma_{\leftrightarrow}, \mathbf{y})$. Thus, the performance of this improved sweep cycle is better than that of the best simple cycle. □

4.4.3 The Case of Homogeneous Locations

Consider a patrol problem where all locations have the same detection rate. Unlike when deriving the best simple cycle, the homogeneity of detection rates does not allow us to conclude that the best duration will be the same at each location when considering a sweep cycle. The reason that the dimension of the problem cannot be reduced in this way is that the time between searches is not consistent from location to location when following a sweep cycle. In particular, after the first search of the location at the start of the sweep, the patroller must go to and visit every other location twice. She will then conduct her second search of the first location, which immediately rolls into another search of the same location at the start of the next cycle. In contrast, the two searches of a location near the middle of the sweep will be more evenly spaced throughout the cycle. It is therefore intuitive that the best search duration is longer for the first location than one in the middle. However, there are two locations that will always have the same search duration when detection rates are homogeneous. These are the locations at the start and end of the sweep cycle since both receive two consecutive searches and then are not visited again for the rest of the cycle. This observation allows us to reduce the number of decision variables by one, which makes finding the best sweep cycle easier. It is worth noting that while the detection rates should be the same for these two locations, they do not need to be the same everywhere else.

The number of decision variables can be further reduced if certain other conditions are met. Suppose that, in addition to detection rates being homogeneous for all locations, the travel time between each pair of adjacent locations along the line are also the same. In this case the number of decision variables can be reduced to $\lceil n/2 \rceil$. This condition is met when all locations are equally spread out along a line. In this somewhat more restrictive case, the best search duration for any particular location $i \leq \lfloor n/2 \rfloor$ must be the same as the best search duration for location $n+1-i$. If n is odd then there is one location where $i = n+1-i$ so this location is paired with itself. This is because the time that the patroller spends away from location i after the first search of location i is the same as the time that the patroller spends away from location $n+1-i$ after the second search of location $n+1-i$ and vice versa.

4.4.4 The Case of Heterogeneous Locations

Like with simple cycles, if the detection rates are heterogeneous, then the number of decision variables is equal to the number of locations.

We now provide an approach for computing the best sweep pattern given a particular ordering σ. Recall that we have defined a sweep so that the endpoints are visited twice in each cycle. Without loss of generality, assume that the ordering being used is the identity ordering. Thus the cycle pattern takes the form $1 \to \cdots \to n \to n \to \cdots \to 1$. Thus the experience of each location $i \in [n]$ is as follows. First it is searched for x_i, then the patroller goes away for y_{i1} units of time, then it is searched for a further x_i units of time, then finally the patroller goes away for y_{i2} units of time. Note that for the end location the time spent away can be zero.

Let c be the total cycle time. Thus

$$c = \sum_{j=1}^{n} 2x_j + \sum_{j=1}^{n-1} \left(d_{j,j+1} + d_{j+1,j}\right).$$

For each $i \in [n]$ let y_{i1} be the time taken between the first and second search of location i. Also let y_{i2} be the sum of times before the first search and after the second search of location i. Thus, $y_{11} = 0$ and $y_{12} = c - 2x_1$. For all $i \in [n] \backslash \{1\}$,

$$y_{i1} = \sum_{j=1}^{i-1} 2x_j + \sum_{j=1}^{i-1} \left(d_{j,j+1} + d_{j+1,j}\right).$$

Similarly $y_{n1} = c - 2x_n$ and $y_{n2} = 0$. For all $i \in [n-1]$,

$$y_{i2} = \sum_{j=i+1}^{n} 2x_j + \sum_{j=i}^{n-1} \left(d_{j,j+1} + d_{j+1,j}\right).$$

By examining (4.5) from the proof of Proposition 4.4 and applying Little's Law it is clear that the expected time to discover an attacker at a location $i \in [n]$ while sweeping is

$$f_i(\sigma, \mathbf{x}) = \frac{1}{\lambda_i} + \frac{1}{c}\left[\frac{y_{i1}+y_{i2}}{\lambda_i} + \frac{y_{i1}^2+y_{i2}^2+2y_{i1}y_{i2}e^{-\lambda x_i}}{1-e^{-2\lambda x_i}} - \frac{y_{i1}^2+y_{i2}^2}{2}\right]. \tag{4.14}$$

Recall that the patroller wishes to minimise the maximum expected time to discover an attacker across all locations. She therefore wishes to find

$$(x_1^*, \cdots, x_n^*) = \underset{(x_1,\cdots,x_n)\in(\mathbb{R}^+)^n}{\operatorname{argmin}} \left(\max_{i\in[n]}\{f_i(\sigma, \mathbf{x})\}\right)$$

where for each $i \in [n]$, $f_i(\sigma, \mathbf{x})$ is given by (4.14).

We can therefore use (4.14) to find the best sweep cycle using a numerical approach that we discuss in Section 4.5.2.

4.5 Numerical Demonstration

In this section we numerically demonstrate how to compute the best simple cycle and the best sweep cycle. For (4.13) and (4.14) computing exact gradients is challenging. Therefore, to use standard gradient descent methods, we would have to approximate the gradient, which can be computationally expensive in high dimensions. Gradient descent methods also do not take into account any of the structural properties of the functions they are trying to optimise. Both of the functions we are trying to optimise appear to be unimodal. We therefore present a ternary search algorithm which can be used to compute the minimum of a unimodal function without the need for gradient approximations. In Section 4.5.1 we present a ternary search algorithm capable of computing the best simple cycle for a patrol problem with homogeneous detection rates. In Section 4.5.2 we explain how nesting ternary search algorithms allows us to compute the best cycle of a given type more generally. In Section 4.5.3, we implement the nested ternary algorithm on a few examples to find the best simple cycle and, where a sweep cycle seems appropriate, also find the best sweep cycle. We also present examples to show that simple cycles and sweep cycles are not always optimal.

4.5.1 Ternary Search

The basic ternary search, where the objective is to find a point within a chosen tolerance $\varepsilon > 0$ of the minimum of a unimodal function f with one bounded decision variable, works as follows:

1. Initialise a and b as the lower and upper bounds for the decision variable.
2. Let $l = (2a + b)/3$ and $r = (a + 2b)/3$.
3. Evaluate $f(l)$ and $f(r)$.
4. If $f(l) \geq f(r)$ then it is clear that the minimum cannot be in the lower third so we update $a = l$. Similarly, if $f(l) \leq f(r)$ then it is clear that the minimum cannot be in the upper third so we update $b = r$.
5. If $(b - a)/2$ is less than ε return the approximate optimal value $(a + b)/2$. Otherwise, return to Step 2.

The number of iterations required for convergence depends on the initial difference between a and b and the chosen tolerance level required for termination.

We wish to use ternary search to find the best patrol pattern of a given cycle type for an n location heterogeneous patrol problem. In this section we focus on finding the best simple cycle for problems with homogeneous locations. Proposition 4.6 tells us that any minimum Hamiltonian cycle will allow us to find a best simple cycle. For the case of homogeneous detection rates, we can use Proposition 4.7 to reduce the problem of finding the best search duration for a simple cycle to a single dimension. To find x^*, the best search duration for a simple cycle, we need to minimise (4.11).

To be able to apply ternary search to the problem of finding the best simple cycle we need to show that (4.11) is unimodal in x.

Proposition 4.10. *The function*

$$F(x) = \frac{D(\sigma^*) + nx - x}{D(\sigma^*) + nx} \left(\frac{1}{\lambda} - \frac{D(\sigma^*) + nx - x}{2} + \frac{D(\sigma^*) + nx - x}{1 - e^{-\lambda x}} \right).$$

has a unique minimum for $x > 0$.

Proof. To show that $F(x)$ has a unique minimum for $x > 0$ we show that it is convex. A full account of this proof is given in Appendix 4.7.2. □

To use ternary search we also need to know an upper bound for the decision variable in order to begin iterations. This can be done using the following algorithm:

1. Set $\hat{x}_i = 1$ as an initial estimate of x_i^*.
2. Let $L = \hat{x}_i/2$ and $U = 2\hat{x}_i$.
3. Evaluate $f(L)$, $f(\hat{x}_i)$ and $f(U)$.
4. If $f(\hat{x}_i) < \min(f(L), f(U))$ then we can conclude that $f(L)$ is a lower bound for x_i^* and $f(U)$ is an upper bound for x_i^*.
5. Otherwise, if $f(\hat{x}_i) \geq f(L)$ replace L by $L/2$ and if $f(\hat{x}_i) \geq f(U)$ replace U by $2U$ and return to Step 3.

In Section 4.5.2 we discuss how ternary search can be adapted for problems where finding the best vector of search durations cannot be reduced to 1 dimension.

4.5.2 Nested Ternary Search

In Section 4.5.1 we introduced ternary search and described how it could be used to find the best simple cycle when detection rates are homogeneous. We now consider how this approach can be adapted for finding the best simple cycle when detection rates are not homogeneous, and also for finding the best cycle of other cycle types.

The difficulty here is that ternary search is only capable of optimising a single decision variable, but we are now attempting to find a best vector of search durations. This challenge can be addressed by nesting ternary search algorithms within each other. This approach results in an algorithm with n layers, where each layer is a ternary search algorithm aiming to optimise a single decision variable. We start with the top layer by using ternary search to optimise x_1^*. Each time we call the function f in this layer, another ternary search algorithm is run to find the best value of x_2 for that fixed value of x_1. In each subsequent layer another decision variable is fixed until at the bottom layer we are trying to find the best value of x_n for fixed values of $x_1, \cdots, x_{n-1}$. This nesting procedure solves the issue of having a multi-dimensional decision variable but results in an algorithm that scales poorly with n in terms of computational intensity.

Another consideration when applying this nested ternary search algorithm is unimodality of the objective function.

Conjecture 4.11. *For both simple cycles and sweep cycles, let $f(x_1, \ldots, x_n)$ be the expected time to discovery for the cycle where each location $i \in [n]$ is searched for x_i units of time. The function*

$$g(x_1, \ldots, x_{n-1}) = \min_{x_n \in \mathbb{R}^+} [f(x_1, \ldots, x_n)]$$

has only one local minimum.

Proving this result in general is challenging. However, for a particular problem instance, it is possible to graphically verify this property.

As an example, consider a patrol problem with 3 locations and the following parameters: $\lambda_1 = \lambda_2 = 1$, $\lambda_3 = 2$, $d_{12} = d_{21} = d_{23} = d_{32} = 1$, and $d_{13} = d_{31} = 2$. When considering the best ordering for a simple cycle, we refer to Proposition 4.6, which tells us that any minimum Hamiltonian cycle will allow us to find a best simple cycle. For a three location problem, any ordering without repetition is a minimum Hamiltonian cycle. To find $\mathbf{x}^*$, the vector of best search durations, for a simple cycle we need to minimise the maximum of $f_i(\sigma, \mathbf{x})$ over $i \in [n]$ where $f_i(\sigma, \mathbf{x})$ is given by (4.13).

Figure 4.4 depicts a surface for $g(x_1, x_2)$ as defined in Conjecture 4.11 and Figure 4.5 is a contour plot for the same function. We can see from Figures 4.4 and 4.5 that within the ranges shown $x_1, x_2 \in (1, 5)$, there exists a unique minimum for g. Applying ternary search yields a best search duration of 2.488 at locations 1 and 2, which matches what we can see in the contour plot. The best x_3 corresponding to this unique minimum is 1.408, which yields an expected time to discovery of 5.306. Consequently, Conjecture 4.11 holds for simple cycles in this case.

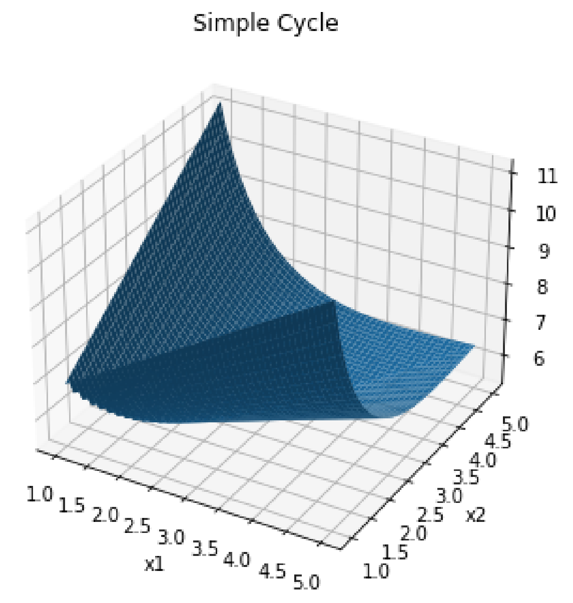


Figure 4.4: Surface showing convexity for a simple cycle

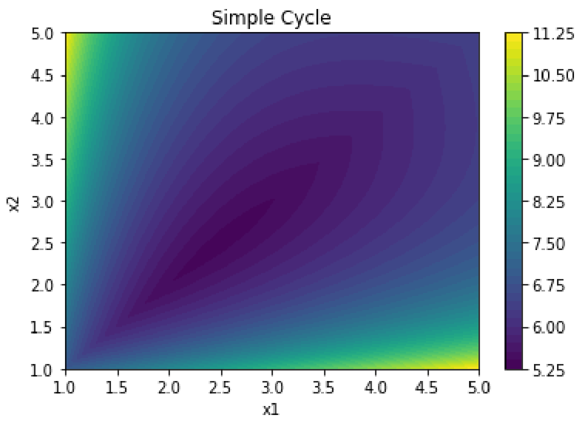


Figure 4.5: Contour plot showing convexity for a simple cycle

Finding the best ordering for a sweep cycle for a general patrol problem is also challenging. However, as mentioned earlier, we only suggest using a sweep pattern in scenarios similar to line networks where an efficient ordering is obvious. In this case the points are in a line so $1 \to 2 \to \cdots \to n$ is the obvious ordering. To find $\mathbf{x}^*$, the vector of best search durations, for a sweep cycle, we need to minimise the maximum of $f_i(\sigma_{\leftrightarrow}, \mathbf{x})$ over $i \in [n]$ where $f_i(\sigma_{\leftrightarrow}, \mathbf{x})$ is given by (4.14).

Figure 4.6 depicts a surface for $g(x_1, x_2)$ as defined in Conjecture 4.11 and Figure 4.7

depicts a contour plot for the same function. We can see from Figures 4.6 and 4.7 that within the ranges shown $x_1, x_2 \in (0.5, 3)$, there exists a unique minimum. Applying ternary search yields a best search duration of 1.139 at location 1 and 1.406 at location 2 which matches what we can see in the contour plot. The best x_3 corresponding to this unique minimum is 0.834. Consequently, Conjecture 4.11 holds for sweep cycles in this case. A sweep cycle with these parameters has an expected time to discovery of 5.048, which is better than the best simple cycle.

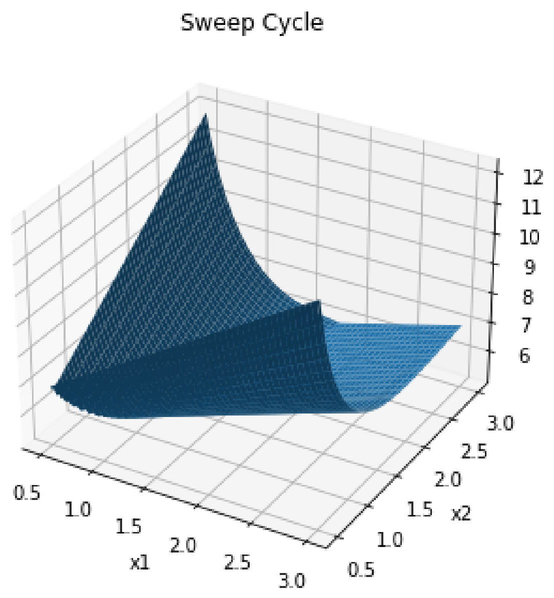


Figure 4.6: Surface showing convexity for a sweep cycle

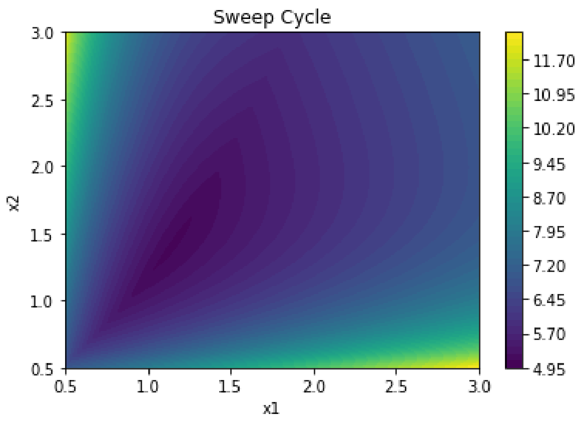


Figure 4.7: Contour showing convexity for a sweep cycle

4.5.3 Examples

In this section we examine some specific three and four location patrol problems. The three location scenarios are shown in Figures 4.8, 4.9, 4.10 and 4.11 while four location scenarios are shown in Figure 4.12 and six location scenarios are shown in Figures 4.13 and 4.14. In each of these figures, the patrol problems are depicted as graphs with each node representing a location and each arc representing a route between two locations. In each scenario, the location number is given inside the node and the detection rate for that location is noted beside the node. The time to travel between two locations is displayed next to the corresponding arc. For simplicity, in all of these scenarios the travel times between each pair of distinct locations are symmetric and so for all $i, j \in [n]$ we have $d_{ij} = d_{ji}$. Note that where there is no arc between two nodes it should be understood that there is no direct route between the corresponding locations. Thus the travel time between these locations is equal to the shortest indirect route between these locations. For example, in scenarios (d), (e) and (f), shown in Figure 4.9, there is no direct route between locations 1 and 3 so $d_{13} = d_{12} + d_{23} = 2$.

Example 4.12. Scenarios (a), (b) and (c), given in Figure 4.8 are each geographically symmetric, meaning that all travel times are set to d for some $d \in \mathbb{R}^+$. In each of these scenarios, $d = 1$. Scenarios (a) and (b) also have homogeneous detection rates with $\lambda = 1$ for all locations in scenario (a) and $\lambda = 2$ for all locations in scenario (b).

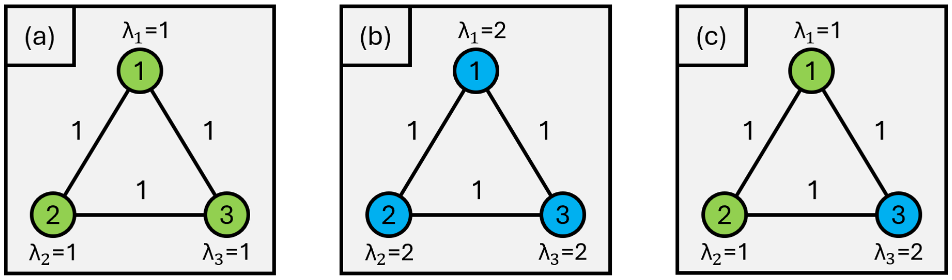


Figure 4.8: Three location examples with unit travel times

For scenarios (a) and (b), the best simple cycle can be found using the ternary search algorithm as described in Section 4.5.1. For scenario (a), the best simple cycle searches each location for $x^* = 2.230$ units of time which yields an expected time to discovery of $f(\sigma, \mathbf{x}^*) = 5.333$. For scenario (b), the best simple cycle searches each location for $x^* = 1.361$ units of time which yields an expected time to discovery of $f(\sigma, \mathbf{x}^*) = 3.540$. Since the only difference between these two scenarios is that the homogeneous detection rate is higher for scenario (b), the expected time to discover each attacker in (b), even under the same policy, must be lower. Additionally, it makes sense that this policy could be further improved by reducing the amount of time spent searching each location, thereby allowing the patroller to get back to the other locations more quickly.

Scenario (c) has heterogeneous detection rates with $\lambda_1 = \lambda_2 = 1$ and $\lambda_3 = 2$. The best simple cycle for this problem can be computed using nested ternary search as described in Section 4.5.2. The search duration for the two locations with the lower detection rate is $x_1^* = x_2^* = 2.304$ and the duration for the other location is $x_3^* = 1.284$. These values yield an expected time to discover an attacker of $f(\sigma, \mathbf{x}^*) = 4.723$. This result lies between the expected time to discover an attacker in scenarios (a) and (b), because two of the detection rates are the same as in scenario (a) and the other is the same as in scenario (b).

The best sweep cycle can also be computed for each of these scenarios using nested ternary search. These are not the sorts of problems where we would expect a sweep pattern to perform well and indeed we can see from Table 4.2 that the best sweep cycle performs worse than the best simple cycle for all three scenarios. For scenario (c), the order in which the sweep pattern should visit the locations is not immediately obvious. The results presented in Table 4.2 are for when location 3 is visited last. We would expect the same result if location 3 is visited first. If location 3 was visited second the sweep cycle performs slightly worse with an expected time to discover an attacker of $f(\sigma_{\leftrightarrow}, \mathbf{x}) = 5.056$. ▲

The best simple and sweep cycles for each of the three location scenarios are shown in Table 4.2. Each of these results were calculated using ternary search or nested ternary search. The tolerance ε in all cases was set to 10^{-8}. Throughout this section all numerical results are rounded to three decimal places.

	Best Simple Cycle				Best Sweep Cycle			
Scenario	x_1^*	x_2^*	x_3^*	$f(\sigma, \mathbf{x}^*)$	x_1^*	x_2^*	x_3^*	$f(\sigma_{\leftrightarrow}, \mathbf{x}^*)$
(a)	2.230	2.230	2.230	5.333	1.415	1.124	1.415	5.657
(b)	1.361	1.361	1.361	3.540	0.883	0.620	0.883	3.865
(c)	2.304	2.304	1.284	4.723	1.406	1.139	0.834	5.047
(d)	2.425	2.425	2.425	5.932	1.415	1.124	1.415	5.657
(e)	1.473	1.473	1.473	4.096	0.883	0.670	0.883	3.865
(f)	2.488	2.488	1.408	5.306	1.406	1.139	0.834	5.048
(g)	3.742	3.742	3.347	14.667	2.296	1.552	2.296	14.081
(h)	3.778	3.778	3.778	15.083	2.373	1.471	2.373	14.884

Table 4.2: Three location results for simple cycles and sweep cycles

Example 4.13. Scenarios (d), (e) and (f) given in Figure 4.9 each show a line of three evenly spaced locations with $d_{12} = d_{23} = 1$ and $d_{13} = 2$. Scenarios (d) and (e) have homogeneous detection rates with $\lambda = 1$ for all locations in scenario (d) and $\lambda = 2$ for all locations in scenario (e). The best simple cycle for each of these scenarios can be found using the ternary search algorithm described in Section 4.5.1. Scenario (f) has detection rates of $\lambda_1 = \lambda_2 = 1$ and $\lambda_3 = 2$. We can find the best simple cycle for scenario (f) and the best sweep cycle for scenarios (d), (e) and (f) using nested ternary search as described in Section 4.5.2.

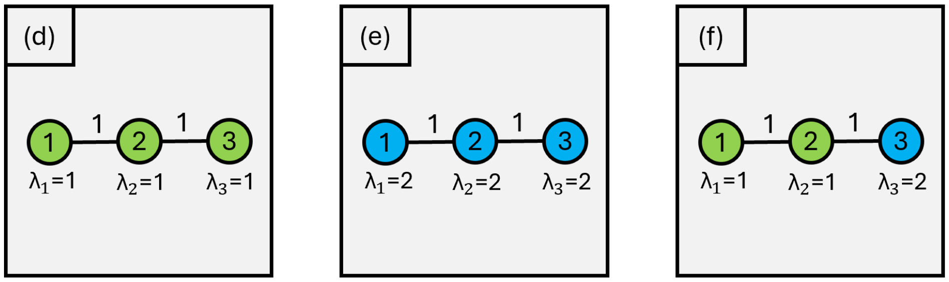


Figure 4.9: Three location examples with points equally spaced along a line

For scenario (d), the search durations for the best simple cycle are all $x^* = 2.425$, which yields an expected time to discover each attacker of $f(\sigma, \mathbf{x}^*) = 5.932$. The search durations for the best sweep are $x_1^* = x_3^* = 1.415$ for locations 1 and 3 and $x_2^* = 1.124$ for location 2. These values yield an expected time to discover each attacker of $f(\sigma_{\leftrightarrow}, \mathbf{x}^*) = 5.657$. During the sweep pattern, the patroller spends a higher proportion of her time at locations 1 and 3. This observation makes sense, since both of these locations have long gaps without visitation during the cycle. Sweeping also spends a lower proportion of time travelling between locations, leading to a lower expected time to discover each attacker.

For scenario (e), the search durations for the best simple cycle are all $x^* = 1.473$, which yields an expected time to discover each attacker of $f(\sigma, \mathbf{x}^*) = 4.096$. The search durations for the best sweep are $x_1^* = x_3^* = 0.883$ for locations 1 and 3 and $x_2^* = 0.670$ for location 2. These values yield an expected time to discover each attacker of $f(\sigma_{\leftrightarrow}, \mathbf{x}^*) = 3.865$.

For scenario (f), the search durations for the best simple cycle are all $x^* = 2.488$, which yields an expected time to discover each attacker of $f(\sigma, \mathbf{x}^*) = 5.306$. The search durations for the best sweep are $x_1^* = 1.406$, $x_2^* = 1.139$ and $x_3^* = 0.834$. These values yield an expected time to discover each attacker of $f(\sigma_{\leftrightarrow}, \mathbf{x}^*) = 5.048$. As with scenario (d), the best sweep cycle also performs better for both of these scenarios. ▲

So far, the scenarios we have discussed have been chosen because they have characteristics that make either simple cycles or sweep cycles perform well. Recall from Section 4.3.1 that an n location ring network is a set of n points arranged in a circle, where it is only possible to move directly to adjacent locations. Thus for each $i, j \in [n]$ with $i < j$, the time taken to move from i to j is

$$\min\left(\sum_{k=i}^{j-1} d_{k,k+1}, \sum_{k=1}^{n-1} d_{k,k+1} + d_{n,1} - \sum_{k=i}^{j-1} d_{k,k+1}\right).$$

Scenarios (a), (b) and (c) are examples of ring networks which we would expect a simple cycle to perform particularly well for. Scenarios (d), (e) and (f) are examples of line networks which we would expect a sweep cycle to perform particularly well for, as discussed in Section 4.4.

However, a simple cycle is not necessarily optimal on a ring network, and a sweep cycle is not necessarily optimal on a line network, as demonstrated in the next few scenarios.

Example 4.14. In general, to show that a particular cycle type is not optimal, we first need to find the best cycle of the given type and then find at least one patrol pattern that performs better. Consider scenario (g), in which location 1 is very far away from the other two locations.

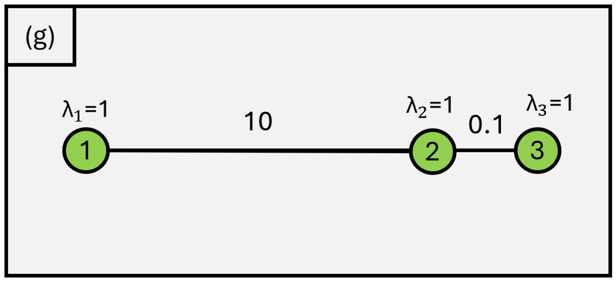


Figure 4.10: A three location example with points unevenly spaced along a line

It is intuitive that if two out of three locations are very close together while the third is further away then it may be beneficial for the patroller to move between the closer locations a few times before making the much longer journey to the distant location. If we imagine the two closer locations getting closer and closer together, they could eventually be treated as a super location where the patroller moves rapidly between them during the search. Thus, the alternative cycle type that we propose takes the form $1 \to 2 \to 3 \to 2 \to 3$. The best cycle type of this form can be found using nested ternary search, using an altered version of (4.14) where y_{i1} and y_{i2} are given as follows:

$$
\begin{aligned}
y_{11} &= d_{12} + x_2 + d_{23} + x_3 + d_{32} + x_2 + d_{23} + x_3 + d_{31},\\
y_{12} &= 0,\\
y_{21} &= d_{23} + x_3 + d_{32},\\
y_{22} &= d_{23} + x_3 + d_{31} + 2x_1 + d_{12},\\
y_{31} &= d_{32} + x_2 + d_{23},\\
y_{32} &= d_{31} + 2x_1 + d_{12} + x_2 + d_{23}.
\end{aligned}
$$

Recall that for each $i \in [n]$, y_{i1} and y_{i2} denote the time spent away from location i following the first and second searches respectively. Scenario (g), shown in Figure 4.10, is a line network with $d_{12} = 10$ and $d_{23} = 0.1$. The detection rate at all these locations is 1. The best simple cycle involves searching each location for 3.742 units of time at each location, which yields an expected time to discovery of $f(\sigma, \mathbf{x}^*) = 14.667$.

The best sweep cycle involves searching locations 1 and 3 for 2.296 time units and location 2 for 1.552 time units, which yields an expected time to discovery of $f(\sigma_{\leftrightarrow}, \mathbf{x}^*) = 14.081$. The best cycle of the type $1 \to 2 \to 3 \to 2 \to 3$ involves searching location 1 for $x_1^* = 2.481$ time units and locations 2 and 3 for $x_2^* = x_3^* = 1.949$ time units. These values result in an expected time to discover each attacker of 14.008 time units, which is slightly better than the best sweep cycle. ▲

Example 4.15. Scenario (h), shown in Figure 4.11 is similar to scenario (a) but with location 1 moved so that its distance from both other locations is 10. The distance between location 2 and 3 remains 1 and all detection rates are set to $\lambda = 1$.

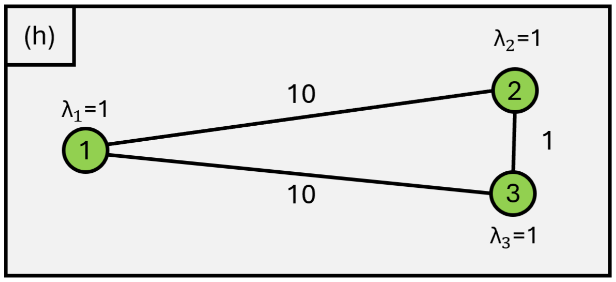


Figure 4.11: A three location example in a stretched triangle

As with Example 4.14 we can find the best simple cycle and the best sweep cycle using nested ternary search as described in Section 4.5.2. The best simple cycle involves searching each location for $x^* = 3.778$ units of time. These values result in an expected time to discovery of $f(\sigma, \mathbf{x}^*) = 15.083$. The best sweep cycle visits the locations in the order that they are labelled. Locations 1 and 3 are searched for $x_1^* = x_3^* = 2.373$ units of time and location 2 is searched for $x_2^* = 1.471$ units of time. These values result in an expected time to discovery of $f(\sigma, \mathbf{x}^*) = 14.884$. This result demonstrates that the sweep cycle can perform better than a simple cycle even when the locations do not form a line network. The best cycle of the type $1 \to 2 \to 3 \to 2 \to 3$ involves searching location 1 for $x_1^* = 2.937$ units of time and locations 2 and 3 for $x_2^* = x_3^* = 1.941$ units of time. These values yield an expected time to discover each attacker of 14.830 units of time, which is slightly better than the best sweep cycle. ▲

Example 4.16. Scenarios (i) and (j), both given in Figure 4.12, each have homogeneous detection rates with $\lambda = 1$ for each of their four locations.

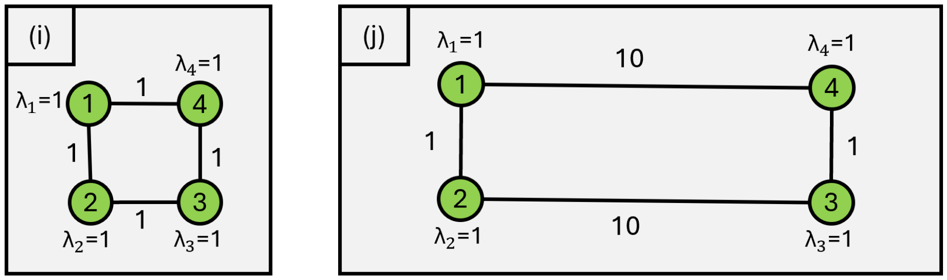


Figure 4.12: Four location examples in a ring

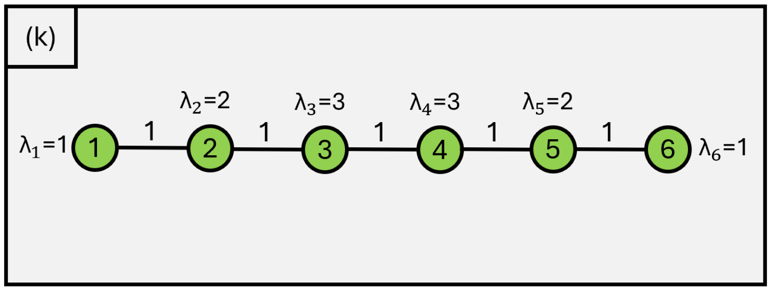
Since both scenarios have homogeneous detection rates, Proposition 4.7 tells us that the best simple cycle must have the same search duration at each location. We can therefore calculate the best simple cycle using ternary search as described in Section 4.5.1. For scenario (i) the best search duration for a simple cycle is 2.067 at all locations, which yields an expected time to discover each attacker of 8.136. For scenario (j) the best search duration for a simple cycle is 3.552 at all locations, which yields an expected time to discover each attacker of 21.366. Unsurprisingly the best search duration is longer for scenario (j) than for scenario (i), as the total travel time for each cycle is larger in scenario (j).

As with scenarios (g) and (h), scenario (j) includes some travel times that are very large. In scenario (j), locations 1 and 2 are very far from locations 3 and 4. Therefore it may be beneficial to visit the first two locations a few times before travelling to the other two. Similarly, once the patroller is on the right-hand side it may be beneficial to search location 3 and 4 a few times before returning. Thus, we propose the following cycle type: $1 \to 2 \to 1 \to 2 \to 3 \to 4 \to 3 \to 4$. A formula for the expected time to discovery can be found using an altered version of (4.14) where y_{i1} and y_{i2} are as follows:

$$\begin{aligned}
y_{11} &= d_{12} + x_2 + d_{21}, \\
y_{12} &= d_{12} + x_2 + d_{23} + x_3 + d_{34} + x_4 + d_{43} + x_3 + d_{34} + x_4 + d_{41}, \\
y_{21} &= d_{21} + x_1 + d_{12}, \\
y_{22} &= d_{23} + x_3 + d_{34} + x_4 + d_{43} + x_3 + d_{34} + x_4 + d_{41} + x_1 + d_{12}, \\
y_{31} &= d_{34} + x_4 + d_{43}, \\
y_{32} &= d_{34} + x_4 + d_{41} + x_1 + d_{12} + x_2 + d_{21} + x_1 + d_{12} + x_2 + d_{23}, \\
y_{41} &= d_{43} + x_3 + d_{34}, \\
y_{42} &= d_{41} + x_1 + d_{12} + x_2 + d_{21} + x_1 + d_{12} + x_2 + d_{23} + x_3 + d_{34}.
\end{aligned}$$

Since the locations are homogeneous and the travel times between visits are identical for each location (one long trip and one short trip) it follows that using the same search duration for each location should perform well. Because there is only one decision variable, the ternary search algorithm can be used to find the best policy of this type. For scenario (i) the best search duration for a cycle of this type with all searches the same length is 3.308 at all locations, which yields an expected time to discover each attacker of 8.567. This cycle performs worse than the best simple cycle. For scenario (j) the best search duration for a cycle of this type with all searches the same length is 2.060 at all locations, which yields an expected time to discover each attacker of 16.527. This cycle performs significantly better than the best simple cycle and demonstrates that even in ring networks a simple cycle is not always optimal. ▲

Example 4.17. Scenarios (k) and (l), both given in Figure 4.13, include 6 locations spaced at unit intervals along a line. In both scenarios, the detection rates are such that $\lambda_1 = \lambda_6$, $\lambda_2 = \lambda_5$ and $\lambda_3 = \lambda_4$. It is clear to see that this symmetry will ensure that the best possible cyclic policy involves searching location i for the same amount of time as location $7 - i$, for each $i \in \{1, 2, 3\}$.



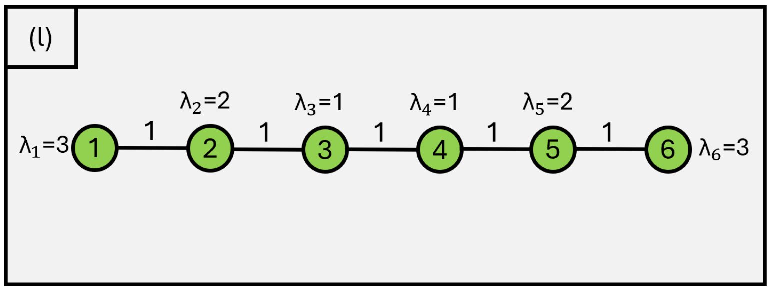


Figure 4.13: Six location examples in a line

For scenario (k), locations 1 and 6 have a detection rate of 1, locations 2 and 5 have a detection rate of 2 and locations 3 and 4 have a detection rate of 3. Since the locations in the middle have higher detection rates and also have greater time separations between successive visits during a cycle, we would expect these locations to have shorter optimal search times. For the best sweep cycle locations 1 and 6 have a search time of 1.396, locations 2 and 5 have a search time of 0.0749 and locations 3 and 4 have a search time of 0.00493. Recall that we have defined a sweep cycle such that the two end locations are visited twice in immediate succession so the total search durations at these locations is 2.792.

For scenario (l), locations 1 and 6 have a detection rate of 3, locations 2 and 5 have a detection rate of 2 and locations 3 and 4 have a detection rate of 1. The two factors referred to in scenario (k) (for justifying the shorter search times for locations in the middle) are therefore in opposition and it is not immediately obvious whether the inner or outer locations should be searched for longer. For the best sweep cycle locations 1 and 6 have a search time of 0.622, locations 2 and 5 have a search time of 0.137 and location 3 and 4 have a search time of 0.0282. ▲

Example 4.18. Scenario (m), given in Figure 4.14, is a ring of six locations. Locations

1 and 2 have a detection rate of 1, locations 3 and 4 have a detection rate of 2 and locations 5 and 6 have a detection rate of 3. This scenario may be particularly applicable when the locations in the cycle can be classified into distinct types. This may be based on different environments such as ‘open grassland’ or ‘forested’ which could reasonably affect the detection rates.

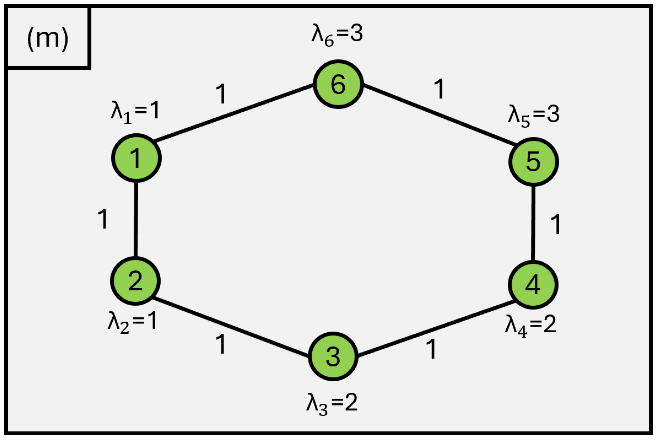


Figure 4.14: Six location example in a ring

The best simple cycle can be calculated using nested ternary search, noting that since three pairs of locations share detection rates we only require three decision variables. The best search durations are 2.175 for the two locations with a detection rate of 1, 1.162 for the two locations with a detection rate of 2 and 0.798 for the two locations with a detection rate of 3. ▲

4.6 Conclusions

In this chapter we introduced a patrol problem where the patroller is tasked with protecting a set of dispersed locations from attacks. The time needed for the patroller to travel between these locations is fixed and depends on the locations that she is moving between. While at a location, the patroller can spend any amount of time searching it for attackers. Her search has a conditional detection rate which is known and can vary between locations. Her objective is to minimise the expected duration of an attack regardless of where that attack occurs.

We first examined the special case where there is no cost of moving between any pair of locations. We showed that for this case it is optimal to continuously allocate a constant fraction of effort to each location and that at each location the fraction of effort that should be allocated is inversely proportional to the detection rate.

Motivated by ring and line networks we presented two cycle types: simple cycles and sweep cycles. For each cycle type, we derived a formula for the expected time to discover an attacker at a particular location. We then described how to find the best patrol pattern of each cycle type using a nested ternary search algorithm. The discussion of each cycle type also included some special cases where the number of decision variables that need to be optimised can be reduced.

We gave several examples of ring and line networks and show how to compute the best simple cycles and the best sweep cycles. In each of these cases we used the nested ternary search algorithm to compute the best parameters of a given cycle type. We also gave examples to demonstrate that the optimal policy on a ring network is not necessarily a simple cycle, and the optimal policy on a line network is not necessarily a sweep cycle. Since we have no way of deciding the best cycle type the best we can do is make an informed guess of what cycle type will perform well and then find the best parameters of that cycle type.

A possible area of further work is to develop heuristics for a general network. Further work is also needed to prove Conjecture 4.5.

In this chapter, we have assumed that the cost of an attack is the same at each location. A natural extension to this problem is to weigh attacks at different locations differently. These weights could be used to indicate higher value assets that could be damaged or stolen, more dangerous machinery that is more likely to cause harm if broken, or more sensitive sources of information that would be more harmful or costly if leaked. This information could be included in the model via an additional parameter c_i where, for each $i \in [n]$, c_i indicates the cost per unit time of an attacker being present at location i. The patroller's new objective would be to minimise the maximum expected cost of an attack across all locations.

4.7 Appendix

4.7.1 Supporting Evidence of Conjecture 4.5

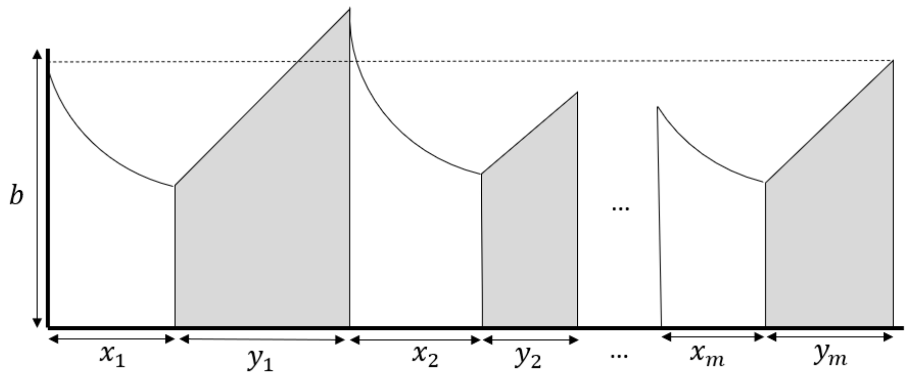
Consider a two location patrol problem and suppose there exists an optimal cycle of length c in which each location is visited m times and where the total time spent searching location 1 is s. Let one of the arrivals at location 1 be the start of the cycle and for each $i \in [m]$, let x_i denote the length of time that the patroller searches location 1 for during the i^{th} visit. Thus $\sum_{i=1}^{m} x_i = s$. Also let y_i denote the time spent away from location 1 immediately after the i^{th} search. This value includes the time to travel to and from location 2 as well as some time spent searching there. Thus $\sum_{i=1}^{m} y_i = c - s$. We claim that the optimal values for the durations of these searches are $x_1 = \cdots = x_m = s/m$ and the optimal durations for the gaps between the searches are $y_1 = \cdots = y_m = (c-s)/m$.

Consider the expected number of attackers at location 1 throughout the cycle at steady state. We can think of this scenario as a queuing system with:

- A constant arrival rate of θ which represents attackers arriving at location 1.
- Infinitely many servers, each with a constant service rate λ, that are only active while the patroller is in location 1.

The expected time a single attacker (or customer) spends in the system is comprised of the expected service time $(1/\lambda)$ and the expected idle time.

Let b denote the number of customers in the system at the start of the cycle during steady state. The figure below shows the expected number of customers in the system throughout the cycle. The white areas show when the servers are active and the grey areas show when they are idle.



Thus

$$b = \left(\cdots \left(\left(be^{-\lambda x_1} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_1}) + \theta y_1 \right) e^{-\lambda x_2} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_2}) + \theta y_2 \right) e^{-\lambda x_3} + \cdots + \theta y_m \right).$$

Here the dots how the m terms nested within each other. Collecting all b terms on the left hand side gives us

$$b(1 - e^{-\lambda s}) = \left(\cdots \left(\left(\frac{\theta}{\lambda}(1 - e^{-\lambda x_1}) + \theta y_1 \right) e^{-\lambda x_2} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_2}) + \theta y_2 \right) e^{-\lambda x_3} + \cdots + \theta y_m \right).$$

Dividing by $(1 - e^{-\lambda s})$ and rearranging yields

$$b = \theta \left[\frac{1}{\lambda} + \frac{y_m + \sum_{i=1}^{m-1} y_i e^{-\lambda \sum_{j=i+1}^{m} x_j}}{1 - e^{-\lambda s}} \right].$$

If we define $x_{m+1} = 0$, then we can rewrite this equation as

$$b = \theta \left[\frac{1}{\lambda} + \frac{\sum_{i=1}^{m} y_i e^{-\lambda \sum_{j=i+1}^{m+1} x_j}}{1 - e^{-\lambda s}} \right].$$

Write L for the long-run average number of customers in the system when the patroller is on site (x_1 and x_2). The rate at which customers depart when the patroller is on site is $L\lambda$, so the average number of customers that depart the system in each cycle is $sL\lambda$. The long-run average number of customers that arrive in each cycle is $c\theta$. Setting $sL\lambda = c\theta$ yields $L = c\theta/s\lambda$.

The area of all white segments combined is therefore

$$Ls = \frac{c\theta}{\lambda}.$$

The area of the first shaded region is

$$\left(be^{-\lambda x_1} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_1}) + \frac{\theta y_1}{2}\right) y_1$$

$$= \theta y_1 \left(\frac{1}{\lambda} + \frac{y_1}{2} + \sum_{i=1}^{m} \frac{y_i e^{-\lambda(x_1 + \sum_{j=i+1}^{m+1} x_j)}}{1 - e^{-\lambda s}}\right)$$

The area of the second shaded region is

$$\left(\left(be^{-\lambda x_1} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_1}) + \theta y_1\right) e^{-\lambda x_2} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_2}) + \frac{\theta y_2}{2}\right) y_2$$

$$= \theta y_2 \left(\frac{1}{\lambda} + y_1 e^{-\lambda x_2} + \frac{y_2}{2} + \frac{e^{-\lambda(x_1+x_2)} \left(\sum_{i=1}^{m} y_i e^{-\lambda \sum_{j=i+1}^{m+1} x_j}\right)}{1 - e^{-\lambda s}}\right)$$

For each $k \in [m]$ the area of the k^{th} shaded region is y_k multiplied by:

$$\left(\cdots\left(\left(be^{-\lambda x_1} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_1}) + \theta y_1\right) e^{-\lambda x_2} + \frac{\theta}{\lambda}(1 - e^{-\lambda x_2}) + \theta y_2\right) e^{-\lambda x_3} + \cdots + \frac{\theta y_k}{2}\right)$$

This can be rewritten as

$$\theta y_k \left(\frac{1}{\lambda} - \frac{y_k}{2} + \frac{\sum_{i=k+1}^{m} y_i e^{-\lambda(\sum_{j=i+1}^{m+1} x_j + \sum_{j=1}^{k} x_j)} + \sum_{i=1}^{k} y_i e^{-\lambda \sum_{j=i+1}^{k} x_j}}{1 - e^{-\lambda s}}\right).$$

Adding these terms and dividing by c gives us the long-run average number of customers in the system.

We want to find the values of x_1, x_2, y_1 and y_2 that minimise the long-run average number of customers in the system. The area of the white segments has no dependence on these parameters so it is sufficient to only consider the area of the shaded segments, so we wish to minimise:

$$\sum_{k=1}^{m} y_k \left(\frac{1}{\lambda} - \frac{y_k}{2} + \frac{\sum_{i=k+1}^{m} y_i e^{-\lambda(\sum_{j=i+1}^{m+1} x_j + \sum_{j=1}^{k} x_j)} + \sum_{i=1}^{k} y_i e^{-\lambda \sum_{j=i+1}^{k} x_j}}{1 - e^{-\lambda s}}\right)$$

$$= \frac{y_1 + \cdots + y_m}{\lambda} - \frac{y_1^2 + \cdots + y_m^2}{2}$$

$$+\frac{1}{1-e^{-\lambda s}}\sum_{k=1}^{m}\left(y_k\left(\sum_{i=k+1}^{m} y_i e^{-\lambda(\sum_{j=i+1}^{m+1} x_j+\sum_{j=1}^{k} x_j)}+\sum_{i=1}^{k} y_i e^{-\lambda\sum_{j=i+1}^{k} x_j}\right)\right)$$

$$=\frac{y_1+\cdots+y_m}{\lambda}-\frac{y_1^2+\cdots+y_m^2}{2}+\frac{1}{1-e^{-\lambda s}}$$

$$\left(\sum_{k=1}^{m}\sum_{i=k+1}^{m} y_k y_i e^{-\lambda(\sum_{j=i+1}^{m+1} x_j+\sum_{j=1}^{k} x_j)}+\sum_{k=1}^{m}\sum_{i=1}^{k} y_k y_i e^{-\lambda\sum_{j=i+1}^{k} x_j}\right)$$

In the first set of sums $i > k$ whereas in the second set of sums $i \leq k$.

$$=\frac{y_1+\cdots+y_m}{\lambda}-\frac{y_1^2+\cdots+y_m^2}{2}+\frac{1}{1-e^{-\lambda s}}$$

$$\left(y_1^2+\cdots+y_m^2+\sum_{k=1}^{m}\sum_{i=1}^{k-1} y_k y_i\left(e^{-\lambda(\sum_{j=1}^{i} x_j+\sum_{j=k+1}^{m} x_j)}+e^{-\lambda\sum_{j=i+1}^{k} x_j}\right)\right)$$

To find the values of $x_1,\ldots,x_m$ that minimise this expression we need only consider terms that contain $x_1,\ldots,x_m$. Thus we seek to minimise:

$$\sum_{k=1}^{m}\sum_{i=1}^{k-1} y_k y_i\left(e^{-\lambda(\sum_{j=1}^{i} x_j+\sum_{j=k+1}^{m} x_j)}+e^{-\lambda\sum_{j=i+1}^{k} x_j}\right) \tag{4.15}$$

subject to

$$\sum_{j=1}^{m} x_j - s = 0 \qquad \text{and} \qquad \sum_{j=1}^{m} y_j - c + s = 0.$$

We will attempt to minimise (4.15) using the method of Lagrangian multipliers. Define the Lagrangian function $\mathcal{L}$ as

$$\mathcal{L}(x_1,,\cdots,x_m,y_1,\cdots,y_m,Z_1,Z_2)$$

$$=\sum_{i=1}^{m-1}\sum_{k=i+1}^{m} y_k y_i\left(e^{-\lambda(\sum_{j=1}^{i} x_j+\sum_{j=k+1}^{m} x_j)}+e^{-\lambda\sum_{j=i+1}^{k} x_j}\right)$$

$$+Z_1(x_1+x_2+x_3-s)+Z_2(y_1+y_2+y_3-c+s)$$

Now, we need to find the critical points by taking partial derivatives with respect to each variable and setting them equal to zero. Finding the partial derivatives with respect to x_1, y_1, Z_1 and Z_2 is relatively straight forward.

$$\frac{\delta\mathcal{L}}{\delta x_1}=-\lambda\left(\sum_{i=1}^{m-1}\sum_{k=i+1}^{m} y_i y_k e^{-\lambda(\sum_{j=1}^{i} x_j+\sum_{k+1}^{m})x_j}\right)+Z_1=0$$

$$\frac{\delta \mathcal{L}}{\delta y_1} = \sum_{k=2}^{m} y_k \left(e^{-\lambda(x_1 + \sum_{j=k+1}^{m} x_j)} + e^{-\lambda \sum_{j=2}^{k} x_j} \right) + Z_2 = 0$$

$$\frac{\delta \mathcal{L}}{\delta Z_1} = \sum_{i=1}^{m} x_m - s = 0$$

$$\frac{\delta \mathcal{L}}{\delta Z_2} = \sum_{i=1}^{m} y_m - c + s = 0$$

The other partial derivatives are a bit more challenging to compute. Recall that we arbitrarily chose one of m visits of location 1 to be the start of the cycle and then labelled the visits of m accordingly. By redefining the start of the cycle we can find an expression that is equivalent to 4.15 but where the subscripts are shifted. To make use of this observation it is necessary to define $x_{m+t} = x_m$ and $y_{m+t} = y_m$ for each $t \in [m]$. We can then write

$$\frac{\delta \mathcal{L}}{\delta x_t} = -\lambda \left(\sum_{i=1}^{m-1} \sum_{k=i+1}^{m} y_{i+t-1} y_{k+t-1} e^{-\lambda(\sum_{j=1}^{i} x_{j+t-1} + \sum_{k+1}^{m} x_{j+t-1})} \right) + Z_1 = 0$$

$$\frac{\delta \mathcal{L}}{\delta y_t} = \sum_{k=2}^{m} y_{k+t-1} \left(e^{-\lambda(x_k + \sum_{j=k+1}^{m} x_{j+k-1})} + e^{-\lambda \sum_{j=2}^{k} x_{j+k-1}} \right) + Z_2 = 0$$

This system of equations can be solved by taking $x = x_1 = \cdots = x_m = s/m$, $y = y_1 = \cdots = y_m = (c-s)/m$,

$$Z_1 = \lambda y \left(\sum_{i=1}^{m-1} i e^{-\lambda i x} \right)$$

and

$$Z_2 = -2y \left(\sum_{i=1}^{m-1} e^{-\lambda i x} \right).$$

We can therefore conclude that setting $x = x_1 = \cdots = x_m = s/m$ and $y = y_1 = \cdots = y_m = (c-s)/m$ yields a stationary point for (4.15).

It remains to show that this stationary point is unique and corresponds to the global minimum. The fact that these equations are not linear makes this problem very challenging.

4.7.2 Proof of Proposition 4.10

Let

$$F(x) = \frac{D(\sigma^*) + nx - x}{D(\sigma^*) + nx}\left(\frac{1}{\lambda} - \frac{D(\sigma^*) + nx - x}{2} + \frac{D(\sigma^*) + nx - x}{1 - e^{-\lambda x}}\right). \tag{4.16}$$

We want to show that (4.16) has a unique minimum for $x > 0$. This can be done by showing that the second derivative is positive, i.e. proving convexity. Since we have fixed σ^* we will, for ease of notation, write $d = D(\sigma^*)$ throughout this discussion. The second derivative can then be written as

$$F''(x) = \frac{G(x)}{\lambda(d + nx)^3(1 - e^{-\lambda x})^3}, \tag{4.17}$$

where

$$\begin{aligned}
G(x) =& \lambda^3(d + nx)^2(d + (n-1)x)^2 e^{-\lambda x}(1 + e^{-\lambda x}) && (4.18)\\
&- 2\lambda^2(d + nx)(d + (n-1)x)((n-2)d + n(n-1)x)e^{-\lambda x}(1 - e^{-\lambda x}) && (4.19)\\
&+ 2nd(1 - e^{-3\lambda x} - 3e^{-\lambda x} + 3e^{-2\lambda x}) && (4.20)\\
&+ d^2\lambda(1 + e^{-3\lambda x} - e^{-\lambda x} - e^{-2\lambda x}). && (4.21)
\end{aligned}$$

Recall that $n \in \mathbb{N}\backslash\{1\}$ and that $d, \lambda > 0$. It is therefore clear that the denominator in (4.17) is positive. Since we can rewrite line (4.20) as $2nd(1 - e^{-\lambda x})^3$ it is also clear that this term is positive. The final term given by line (4.21) is also positive because $g(y) = e^{-\lambda xy}$ is convex in y, so $g(0) + g(3) > g(1) + g(2)$.

Thus, to prove that (4.17) is positive it is sufficient to show that the sum of (4.18) and (4.19) is positive. These terms have a common factor of

$$\lambda^2(d + nx)(d + (n-1)x)e^{-\lambda x}$$

which is clearly positive. We now proceed by induction on n. Define

$$\begin{aligned}
f_n(x) =& \lambda(d + nx)(d + (n-1)x)(1 + e^{-\lambda x}) && (4.22)\\
&- 2((n-2)d + n(n-1)x)(1 - e^{-\lambda x}) && (4.23)
\end{aligned}$$

First, we consider

$$f_2(x) = \lambda(d + 2x)(d + x)(1 + e^{-\lambda x}) - 4x(1 - e^{-\lambda x}).$$

This expression is clearly increasing in d so it is sufficient to show that $f_2(x) > 0$ when $d = 0$. In this case

$$f_2(x) = 2\lambda x^2(1 + e^{-\lambda x}) - 4x(1 - e^{-\lambda x}).$$

Differentiating this with respect to x gives us

$$f_2'(x) = 4\lambda x + 4e^{-\lambda x} - 2\lambda^2 x^2 e^{-\lambda x} - 4$$

Now define $p = \lambda x$ and consider

$$g(p) = 4p + 4e^{-p} - 2p^2 e^{-p} - 4.$$

Its first, second and third derivatives are

$$\begin{aligned} g'(p) &= 4 - 4e^{-p} - 4pe^{-p} + 2p^2 e^{-p}, \\ g''(p) &= 2pe^{-p}(4 - p), \\ g'''(p) &= 2e^{-p}(p^2 - 6p + 4). \end{aligned}$$

By examining $g''(p)$, we can see that $g'(p)$ has a stationary point when $p = 4$ and since $g'''(4) < 0$, this stationary point is a maximum. Given that $g'(0) = 0$ and $g'(p) \to 4$ as $p \to \infty$ and the stationary point is a maximum, $g'(p)$ must always be positive. Hence $f_2'(x) \geq 0$ and $f_2(x) \geq 0$ for all $x > 0$.

Moving on to the inductive step, we have

$$\begin{aligned} &f_{n+1}(x) - f_n(x) \\ &\qquad = -2d + 2\lambda nxe^{-\lambda x} + 2\lambda dxe^{-\lambda x} - 4nx + 2de^{-\lambda x} + 2\lambda nx^2 + 2\lambda dx + 4nxe^{-\lambda x} \\ &\qquad = 2d(\lambda x + \lambda xe^{-\lambda x} + e^{-\lambda x} - 1) + 2nx(2e^{-\lambda x} - 2 + \lambda x + \lambda xe^{-\lambda x}). \end{aligned}$$

Define $p = \lambda x$ and consider

$$g(p) = p + pe^{-p} + e^{-p} - 1$$

and

$$h(p) = 2e^{-p} - 2 + p + pe^{-p}.$$

It is sufficient to show that $g(p) \geq 0$ and $h(p) \geq 0$. We have

$$g'(p) = 2d(1 + e^{-p} - pe^{-p} - e^{-p}) = 2d(1 - pe^{-p}).$$

The function $1 - pe^{-p}$ is non-negative with a minimum when $p = 1$ and thus $g'(p) \geq 0$. Since $g(0) = 0$ this shows that $g(p) \geq 0$.

It now only remains to show that $h(p) \geq 0$. We have

$$h'(p) = 1 - (1 + p)e^{-p} \tag{4.24}$$

$$h''(p) = pe^{-p} \tag{4.25}$$

Is is clear that $h''(p) \geq 0$ so since $h'(0) = 0$ and $h(0) = 0$ it follows that $h'(p) \geq 0$ and $h(p) \geq 0$ for $p \geq 0$. This confirms that $f_{n+1}(x) - f_n(x) \geq 0$.

Given that $f_2(x) \geq 0$ and $f_{n+1}(x) - f_n(x) \geq 0$, we conclude that for all $n \in \mathbb{N}\backslash\{1\}$, $f_n(x) \geq 0$. This completes the proof that the function $F(x)$ in (4.16) is convex.

Chapter 5

Conclusions and Further Work

In this chapter we conclude the thesis by summarising our contributions and proposing some potential avenues for further work.

5.1 Summary

In this thesis, we have studied a search problem and a patrol problem. A key feature shared by these two problems is that the searcher and the patroller both need to travel between dispersed locations to look for attackers, whereas most of the works in the literature assume travel times are negligible.

In the search problem presented in Chapter 3, a target is randomly hidden in one of n locations (for some $n \in \mathbb{N}$) according to a known probability distribution. A searcher dynamically visits and searches these locations as directed by a search policy. We initially applied the Gittins' index policy, which is known to be optimal for the problem without travel times, to the more general problem (with travel times). We showed that whenever the index for the searcher's current location is largest, this index policy will always mandate an optimal action. Motivated by this discovery, we developed an index-based heuristic policy using restless bandit theory that takes into account the cost of moving to the candidate location. To gain some insight into how this index heuristic performed, we compared it to the optimal policy in the case of two location symmetric problems which can be solved analytically. We showed that in this case the structures of both the index policy and the optimal policy are the same, but the index heuristic will sometimes search the current location fewer times than specified under the optimal policy. More generally, we were able to prove that, when the travel times are all the same, the index heuristic only instructs the searcher to remain at her current location when it is optimal to do so.

We also developed two other heuristics that make use of these indices. The first heuristic uses the standard index to plan a route and then uses this plan to construct a more informed index. The second heuristic uses the indices to identify some candidate locations and then uses dynamic programming to approximate an optimal policy on the sub-problem only containing the candidate locations. For all three of these base heuristics we proposed two methods for augmenting their performance. The first is an iterative insertion method and the second is a single step policy improvement algorithm. We then provided extensive numerical results to evaluate the performances of these heuristics with respect to the approximated optimal values.

The second problem studied in this thesis, presented in Chapter 4, concerns a patroller who patrols a set of dispersed locations to detect attackers. As with the search problem, there is a travel time between each pair of locations. The patroller can spend any amount of time searching each location on each visit, during which time they detect any attacker at a constant rate. We initially explored the special case in which all travel times are set to zero and showed that the optimal policy involves the patroller dividing her effort between different locations. This is possible since the locations can be moved between instantly and as frequently as required. We showed that at each location the fraction of effort that should be allocated is inversely proportional to the detection rate.

For the case where travel times are non-zero we introduced two types of patrol cycles—simple cycle and sweep cycle—that intuitively should perform well in some broadly applicable classes of patrol problems. Simple cycles involve visiting each of the locations in the same order. Moreover, each time a given location is visited, the patroller spends the same amount of time searching that location. Intuitively, a simple cycle is a strong policy for patrolling a perimeter.

As part of our work on simple cycles, we showed that:

- If a cycle visits a location twice then both of these visits should be of the same duration.
- The order in which locations are visited in the best simple cycle must correspond to a minimum Hamiltonian cycle.
- In the special case where all detection rates are the same, all locations should be visited for the same amount of time.

The second cycle type—the sweep cycle—is motivated by patrolling a border. In the sweep cycle, the patroller searches all locations in some order during a forward trip and then searches all locations in the reverse order in a return trip. As part of our work on sweep cycles, we showed that:

- The best sweep cycle will spend the same amount of time searching each location on both visits in a cycle.
- When the locations are arranged in a line, the best sweep cycle will always perform better than the best simple cycle.

For each of these cycle types we derived a formula for the expected time to discover an attacker across all locations. We then numerically computed the best simple cycle and the best sweep cycle for a variety of examples. While a simple cycle and a sweep cycle are strong candidate policies, we presented examples to show that the simple cycle need not be optimal for a ring network, and the sweep cycle need not be optimal for a line network.

5.2 Further Work

We now propose some possible directions for future research.

5.2.1 Extensions of the Search Model

There are many ways in which the distributed discrete search problem presented in Chapter 3 can be extended. Since our problem is itself an extension of a well studied search problem, it is intuitive to initially consider extensions and variations of that problem. Many of these extensions and variations are discussed in Chapter 2. We can then consider including travel times within each of these models.

For example, Chew (1973) considers the possibility that the target is not in one of the n locations (i.e., that $\sum_{i=1}^{n} p_i < 1$) and shows that at each epoch an optimal policy either searches the location with the largest Gittins' index or stops searching. An intuitive first attempt to generate a policy for a problem with travel times would be to generate a search sequence using one of the approaches described in Chapter 3 and then truncate it using an appropriate variation of the stopping rule from Chew (1973).

Alternative objectives could also be considered, such as maximising the probability of finding the object by a certain deadline. For the classical problem without travel times, Chew et al. (1967) shows that under certain conditions it is optimal to follow the Gittins index policy until the next proposed search exceeds the deadline. However, with this objective, the order in which locations are searched does not matter. It therefore follows that each location should be visited at most once to avoid unnecessary travel times. If the budget is sufficiently high and we can assume that all locations will be visited at least once then we can simply remove the duration of the minimum Hamiltonian cycle and solve the problem without travel times and a reduced budget. If the budget is small, then for any subset of the locations we can find a candidate solution that visits only those locations. This can be done by first subtracting the duration of the minimum Hamiltonian cycle of the desired subset of locations from the budget and then solving the problem without travel times. It makes more sense to omit locations that have a high search cost, a low detection probability, a low initial hiding probability or that are remote. If there are too many locations to check all possible subsets then a heuristic could be devised to select suitable subsets.

5.2.2 Extensions of the Patrol Model

There are also a number of extensions that may be considered for the patrol model presented in Chapter 4.

In our model, we assumed that the cost of an attack is the same at each location. A natural extension to this problem is to weight attacks at different locations differently. These weights could be used to indicate higher value assets that could be damaged or stolen, more dangerous machinery that is more likely to cause harm if broken or more sensitive sources of information that would be more harmful or costly if leaked. This information could be included in the model via an additional parameter c_i where, for each $i \in [n]$, c_i indicates the cost per unit time of an attacker being present at location i. The patroller's new objective would be to minimise the maximum expected cost of an attack across all locations.

5.2.3 Extensions of Other Models

We now propose another model that includes travel times. Search operations often take place over large areas and can last days, weeks or even years. Conditions affecting the search can change drastically, both among locations and over time. For example, adverse weather can make things harder to spot or interfere with sensors. Even in shorter duration searches conditions may change. During a bomb scare, a searcher's ability to search an area improves after people have been evacuated. In this case, having access to an alternative search mode such as sniffer dogs may be useful. Clarkson et al. (2020) considers problems with multiple search modes but assumes that both modes are available to the searcher at all times.

We can include this variability into the model as follows. Consider an object hidden in one of n geographically dispersed locations according to a known probability distribution. At any given time the conditions at each location may be 'good' or 'bad'. Searches that are carried out while conditions are 'good' are more likely to be successful. Each day, the searcher may pay to search one of these locations or wait for more favourable conditions. Initially, the probability that the object is at location i is p_i, where $p_1 + \ldots + p_n = 1$. A search at location i has a cost of c_i and the probability that the search is successful is g_i if the conditions are good and b_i if the conditions are bad. If the searcher chooses to wait, she will incur a cost of c_0. This search model could have many potential real-world applications.

Bibliography

Agmon, N., Kraus, S., and Kaminka, G. A. (2008a). Multi-robot perimeter patrol in adversarial settings. In *2008 IEEE International Conference on Robotics and Automation*, pages 2339–2345. IEEE.

Agmon, N., Sadov, V., Kaminka, G. A., and Kraus, S. (2008b). The impact of adversarial knowledge on adversarial planning in perimeter patrol. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, pages 55–62.

Alpern, S., Baston, V., and Gal, S. (2009). Searching symmetric networks with utilitarian-postman paths. *Networks: An International Journal*, 53(4):392–402.

Alpern, S., Bui, T., Lidbetter, T., and Papadaki, K. (2022a). Continuous patrolling games. *Operations Research*, 70(6):3076–3089.

Alpern, S., Chleboun, P., Katsikas, S., and Lin, K. Y. (2022b). Adversarial patrolling in a uniform. *Operations Research*, 70(1):129–140.

Alpern, S. and Gal, S. (2002). Searching for an agent who may or may not want to be found. *Operations Research*, 50(2):311–323.

Alpern, S. and Gal, S. (2006). *The theory of search games and rendezvous*, volume 55. Springer Science & Business Media.

Alpern, S. and Lidbetter, T. (2014). Searching a variable speed network. *Mathematics of Operations Research*, 39(3):697–711.

Alpern, S., Lidbetter, T., Morton, A., and Papadaki, K. (2016). Patrolling a pipeline. In *Decision and Game Theory for Security: 7th International Conference, GameSec 2016, New York, NY, USA, November 2-4, 2016, Proceedings 7*, pages 129–138. Springer.

Alpern, S., Lidbetter, T., and Papadaki, K. (2019). Optimizing periodic patrols against short attacks on the line and other networks. *European Journal of Operational Research*, 273(3):1065–1073.

Alpern, S., Morton, A., and Papadaki, K. (2011). Patrolling games. *Operations research*, 59(5):1246–1257.

Assaf, D. and Zamir, S. (1985). Optimal sequential search: a bayesian approach. *The Annals of Statistics*, 13(3):1213–1221.

Assaf, D. and Zamir, S. (1987). Continuous and discrete search for one of many objects. *Operations Research Letters*, 6(5):205–209.

Auger, J. M. (1991). An infiltration game on k arcs. *Naval Research Logistics (NRL)*, 38(4):511–529.

Basilico, N., Gatti, N., and Amigoni, F. (2012). Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial intelligence*, 184:78–123.

Basilico, N., Gatti, N., Amigoni, F., et al. (2009). Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 57–64.

Baston, V. and Kikuta, K. (2013). Search games on networks with travelling and search costs and with arbitrary searcher starting points. *Networks*, 62(1):72–79.

Bertsekas, D. P. (2013). *Rollout Algorithms for Discrete Optimization: A Survey.* In Handbook of Combinatorial Optimization, edited by Panos M. Pardalos, Ding Zhu Du and Ronald L. Graham, 2989-3013. New York: Springer.

Birge, J. and Pollock, S. (1989). Modelling rural police patrol. *Journal of the Operational Research Society*, 40(1):41–54.

Blachman, N. M. (1959). Prolegomena to optimum discrete search procedures. *Naval Research Logistics Quarterly*, 6(4):273–281.

Black, W. L. (1965). Discrete sequential search. *Information and control*, 8(2):159–162.

Blackwell, D. (1962). Notes on dynamic programming. *Unpublished, Dept. of Stat., University of California.*

Bui, T. and Lidbetter, T. (2023). Optimal patrolling strategies for trees and complete networks. *European Journal of Operational Research*, 311(2):769–776.

Bui, T., Lidbetter, T., and Lin, K. Y. (2024). Optimal pure strategies for a discrete search game. *European Journal of Operational Research*, 313(2):767–775.

Chaiken, J. M. and Dormont, P. (1978). A patrol car allocation model: Capabilities and algorithms. *Management Science*, 24(12):1291–1300.

Chelst, K. (1978). An algorithm for deploying a crime directed (tactical) patrol force. *Management Science*, 24(12):1314–1327.

Chew, M. C. (1973). Optimal stopping in a discrete search problem. *Operations Research*, 21(3):741–747.

Chew, M. C. et al. (1967). A sequential search procedure. *The Annals of Mathematical Statistics*, 38(2):494–502.

Clarkson, J., Glazebrook, K. D., and Lin, K. Y. (2020). Fast or Slow: Search in Discrete Locations with Two Search Modes. *Operations Research*, 68(2):552–571.

Clarkson, J. and Lin, K. Y. (2024). Computing optimal strategies for a search game in discrete locations. *INFORMS Journal on Computing.*

Clarkson, J., Lin, K. Y., and Glazebrook, K. D. (2023). A classical search game in discrete locations. *Mathematics of Operations Research*, 48(2):687–707.

Dagan, A. and Gal, S. (2008). Network search games, with arbitrary searcher starting point. *Networks: An International Journal*, 52(3):156–161.

De Guenin, J. (1961). Optimum distribution of effort: an extension of the koopman basic theory. *Operations Research*, 9(1):1–7.

Dobbie, J. M. (1968). A survey of search theory. *Operations Research*, 16(3):525–537.

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

Enslow Jr, P. H. (1966). A bibliography of search theory and reconnaissance theory literature. *Naval Research Logistics Quarterly*, 13(2):177–202.

Garrec, T. (2019). Continuous patrolling and hiding games. *European Journal of Operational Research*, 277(1):42–51.

Gittins, J. (1974). A dynamic allocation index for the sequential design of experiments. *Progress in statistics*, pages 241–266.

Gittins, J. (1989). *Multi-armed bandit allocation indices*. John Wiley & Sons.

Gittins, J., Glazebrook, K., and Weber, R. (2011). *Multi-armed bandit allocation indices*. John Wiley & Sons.

Gittins, J. C. (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148–164.

Glazebrook, K. D., Ruiz-Hernandez, D., and Kirkbride, C. (2006). Some indexable families of restless bandit problems. *Advances in Applied Probability*, 38(3):643–672.

Hohzaki, R. (2016). Search games: Literature and survey. *Journal of the Operations Research Society of Japan*, 59(1):1–34.

Jotshi, A. and Batta, R. (2008). Search for an immobile entity on a network. *European Journal of Operational Research*, 191(2):347–359.

Kadane, J. B. (1968). Discrete search and the Neyman-Pearson lemma. *Journal of Mathematical Analysis and Applications*, 22(1):156–171.

Kadane, J. B. (1971). Optimal whereabouts search. *Operations Research*, 19(4):894–904.

Kimeldorf, G. and Smith, F. H. (1979). Binomial searching for a random number of multinomially hidden objects. *Management Science*, 25(11):1115–1126.

Koopman, B. O. (1946). Search and Screening. *OEG Rep.*

Koopman, B. O. (1956). The theory of search. i. kinematic bases. *Operations research*, 4(3):324–346.

Koopman, B. O. (1957). The theory of search: Iii. the optimum distribution of searching effort. *Operations research*, 5(5):613–626.

Kress, M., Lin, K. Y., and Szechtman, R. (2008). Optimal discrete search with imperfect specificity. *Mathematical methods of operations research*, 68(3):539–549.

Larson, R. C. (1972). *Urban police patrol analysis*, volume 28. MIT Press Cambridge, MA.

Lau, H., Huang, S., and Dissanayake, G. (2008). Discounted mean bound for the optimal searcher path problem with non-uniform travel times. *European journal of operational research*, 190(2):383–397.

Lee, S. M., Franz, L. S., and Wynne, A. J. (1979). Optimizing state patrol manpower allocation. *Journal of the Operational Research Society*, 30(10):885–896.

Lidbetter, T. (2013). Search games with multiple hidden objects. *SIAM Journal on Control and Optimization*, 51(4):3056–3074.

Lidbetter, T. (2020). Search and rescue in the face of uncertain threats. *European Journal of Operational Research*, 285(3):1153–1160.

Lidbetter, T. and Lin, K. Y. (2019). Searching for multiple objects in multiple locations. *European Journal of Operational Research*, 278(2):709–720.

Lin, K. Y. (2022). Optimal patrol of a perimeter. *Operations Research*, 70(5):2860–2866.

Lin, K. Y., Atkinson, M. P., Chung, T. H., and Glazebrook, K. D. (2013). A graph patrol problem with random attack times. *Operations Research*, 61(3):694–710.

Lin, K. Y., Atkinson, M. P., and Glazebrook, K. D. (2014). Optimal patrol to uncover threats in time when detection is imperfect. *Naval Research Logistics (NRL)*, 61(8):557–576.

Lin, K. Y. and Singham, D. I. (2016). Finding a hider by an unknown deadline. *Operations Research Letters*, 44(1):25–32.

Matula, D. (1964). A periodic optimal search. *The American Mathematical Monthly*, 71(1):15–21.

McGrath, R. G. and Lin, K. Y. (2017). Robust patrol strategies against attacks at dispersed heterogeneous locations. *International Journal of Operational Research*, 30(3):340–359.

Mela, D. F. (1961). Information theory and search theory as special cases of decision theory. *Operations Research*, 9(6):907–909.

New Straits Times (2017). Final report on flight MH370 expected to be completed this year - liow. `https://www.nst.com.my/news/2017/02/216228/final-report-flight-mh370-expected-be-completed-year-liow`. Accessed: 2024-02-07.

Olson, D. G. and Wright, G. P. (1975). Models for allocating police preventive patrol effort. *Journal of the Operational Research Society*, 26(4):703–715.

Papadaki, K., Alpern, S., Lidbetter, T., and Morton, A. (2016). Patrolling a border. *Operations Research*, 64(6):1256–1269.

Portugal, D. and Rocha, R. P. (2013). Distributed multi-robot patrol: A scalable and fault-tolerant framework. *Robotics and Autonomous Systems*, 61(12):1572–1587.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and sons.

Richardson, H. R., Stone, L. D., et al. (1971). Operations analysis during the underwater search for Scorpion. *Naval Research Logistics Quarterly*, 18(2):141–157.

Ross, S. M. (1969). A problem in optimal search and stop. *Operations Research*, 17(6):984–992.

Ross, S. M. (1983). Introduction to stochastic dynamic programming: Probability and mathematical.

Shechter, S. M., Ghassemi, F., Gocgun, Y., and Puterman, M. L. (2015). Trading off quick versus slow actions in optimal search. *Operations Research*, 63(2):353–362.

Smith, F. H. and Kimeldorf, G. (1975). Discrete sequential search for one of many objects. *The Annals of Statistics*, pages 906–915.

Stone, L. D. (2004). *Theory of Optimal Search*. Informs.

Stone, L. D., Royset, J. O., Washburn, A. R., et al. (2016). *Optimal Search for Moving Targets*. Springer.

Subelman, E. J. (1981). A hide–search game. *Journal of Applied Probability*, 18(3):628–640.

Sweat, C. W. (1970). Sequential search with discounted income, the discount a function of the cell searched. *The Annals of Mathematical Statistics*, 41(5):1446–1455.

Szechtman, R., Kress, M., Lin, K., and Cfir, D. (2008). Models of sensor operations for border surveillance. *Naval Research Logistics (NRL)*, 55(1):27–41.

Taylor III, B. W., Moore, L. J., Clayton, E. R., Davis, K. R., and Rakes, T. R. (1985). An integer nonlinear goal programming model for the deployment of state highway patrol units. *Management Science*, 31(11):1335–1347.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294.

Tijms, H. C. (2003). *A first course in stochastic models*. John Wiley and sons.

Tognetti, K. P. (1968). An optimal strategy for a whereabouts search. *Operations Research*, 16(1):209–211.

Washburn, A. R. (2014). *Search and Detect*. CreateSpace Independent Publishing Platform.

Wegener, I. (1982). The discrete search problem and the construction of optimal allocations. *Naval Research Logistics Quarterly*, 29(2):203–212.

Whittle, P. (1988). Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A):287–298.

Yanovski, V., Wagner, I. A., and Bruckstein, A. M. (2003). A distributed ant algorithm for\protect efficiently patrolling a network. *Algorithmica*, 37:165–186.