A General Index Heuristic for Search with Mobile Agents

by
Thomas J. Temple
B.A., Dartmouth College (2003)
M.S., Massachusetts Institute of Technology (2006)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2011

Author .... [illegible] ....
Department of Aeronautics and Astronautics
February 4, 2011

Certified by ....
Emilio Frazzoli
Associate Professor of Aeronautics and Astronautics, MIT
Thesis Supervisor

Certified by ....
Brian Williams
Professor of Aeronautics and Astronautics, MIT
Thesis Supervisor

Certified by ....
Anouck Girard
Assistant Professor of Aerospace Engineering, Univ. of Michigan
Thesis Supervisor

Accepted by ....
Prof. Eytan Modiano
Graduate Chair

Abstract

This dissertation considers a suite of *search problems* in which agents are trying to find goals in minimum expected time. Unlike search in data structures in which time is measured by a number operations, search in metric spaces measures time by units of distance and has received much less attention. In particular, search strategies that attempt to minimize expected search time are only available for a handful of relatively simple cases.

Nonetheless many relevant search problems take place in metric spaces. This dissertation includes several concrete examples from navigation and surveillance that would have previously only been approachable by much more *ad hoc* methods. We visit these examples along the way to establishing relevance to a much larger set of problems.

We present a policy that is an extension of Whittle's index heuristic and is applicable under the following assumptions.

- The location of goals are independent random variables.
- The agents and goals are in a length space, *i.e.*, a metric space with continuous paths.
- The agents move along continuous paths with bounded speed.
- The agents' sensing is noiseless.

We demonstrate the performance of our policy by applying it to a diverse set of problems for which solutions are available in the literature. We treat each of the following problems as a special case of a more general search problem:

- search in one-dimensional spaces such as the Line Search Problem (LSP) and Cow Path Problem (CPP),
- search in two-dimensional spaces such as the Lost in a Forest Problem (LFP) and problems of coverage,
- problems in networks such as the Graph Search Problem (GSP) and Minimum Latency Tour Problem (MLTP), and
- dynamic problems such as the Persistent Patrol Problem (PPP) and Dynamic Traveling Repairperson Problem (DTRP).

On each of these we find that our policy performs comparably to, and occasionally better than, the accepted solutions developed specifically for these problems. As a result, we believe that this dissertation contributes a significant inroad into a large space of search problems that meets our assumptions, but that remains unaddressed.

Acknowledgements

This dissertation would not exist but for the support and guidance of my adviser, Prof. Emilio Frazzoli, and the input from the brilliant group of researchers he has brought together. It is indebted to the groundbreaking, cited work of John Enright and Marco Pavone, as well countless, formative, uncited discussions with Ketan Savla, Sertac Karaman, and Kyle Treleaven.

I would like to thank Derek Kingston, Corey Schumacher, Siva Banda, and the many other researchers at AFRL/RBCA at Wright Patterson AFB for their support and advocacy of this work. I would also like to extend my gratitude to our collaborators at the University of Michigan. In particular, Prof. Anouck Girard, along with Prof. Brian Williams, provided key guidance without which this dissertation would be greatly inferior.

More deeply, I am indebted to the educators who have taught me. And more so, I am indebted to my parents who not only taught me, but also positioned me to receive and appreciate this teaching.

Finally, I cannot express sufficient gratitude to my wife Courtney, who has supported me throughout this work, not least by supporting the little boy who wants nothing more than to keep me from it. Despite that fact, it is to Peak that this dissertation is dedicated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It is almost tautological to say that mobile agents attempt to move toward goals and one can nearly take for granted that they would like to reach their goals in a timely fashion. This dissertation considers *search problems* in which either the locations of the agents or those of the goals are uncertain and presents a novel control policy for getting mobile agents to their goals in minimum expected time.

We make the following assumptions throughout.

- The location of goals are independent random variables.
- The agents and goals are in a length space, *i.e.*, a metric space with continuous paths.
- The agents move as single-integrators with bounded speed, typically noiselessly.
- The agents' sensing is noiseless.

Despite these restrictive assumptions, the space of such problems remains vast. It is difficult to make strong statements, either theoretical or otherwise, about the performance of a policy across any non-negligible portion of this problem space. As a result, the main goal of this thesis will be to present a policy that is *general* rather than one that is *cordon bleu.*

We begin with a particular search problem, the Cow Path Problem (CPP), and develop an "index policy" which is heuristic, but nonetheless stands on solid theoretical ground by leveraging the body of work on Whittle Indexability. Then we generalize this policy by analogy into a metric, which we will refer to as an "index function," by which we can compare *partial plans.* Nearly half of this dissertation can be summarized as a collection of recipes demonstrating how such a function can be adapted to the above problem space and demonstrating how to do so in an efficient manner.

We will use this metric in the same way as a receding horizon planner: find the best partial plan, execute its first control action, then recompute. What fundamentally differentiates our algorithm from a receding horizon strategy is the index function allows the comparison of plans with differing and arbitrary length. By considering both long and short plans we are able to avoid many of the issues associated with local methods. With this metric in hand, the primary obstacle is determining how to restrict, parametrize, and efficiently search the space of partial plans.

We demonstrate the performance of our policy by applying it to a diverse set of search problems for which answers are available in the relevant literature. The underlying hypothesis is that if the same policy performs well on all of them, it can reasonably be expected to perform well on other problems in the space for which answers are not available. Since these problems are only anecdotes, we will choose them carefully. We will focus on the most generic cases and the most general algorithms, while attempting to show our policy in the *worst* possible light, comparatively.

We will treat each of the following problems as a special case of the more general search problem to which our policy is applicable. We examine

- search in one-dimensional spaces such as the Line Search Problem (LSP) and Cow Path Problem (CPP),

- search in two-dimensional spaces such as the Lost in a Forest Problem (LFP) and problems of coverage,

- problems in networks such as the Graph Search Problem (GSP) and Minimum Latency Tour Problem (MLTP), and

- dynamic problems such as the Persistent Patrol Problem (PPP) and Dynamic Traveling Repairperson Problem (DTRP).

On each of these, our policy performs comparably well, and occasionally better, than the available solutions. By demonstrating the effectiveness of a single heuristic on such a diverse set of problem, we believe that this dissertation contributes a significant inroad into the large space of yet unaddressed search problems.

1.1 Scope and Terminology

Let us take a moment to clarify the title, and the motivation, of this dissertation, while introducing some essential terminology.

In the context of mobile agents, a *search problem* is a problem in which one or more agents must move through a metric space to find one or more goals. The goals might be points or sets, static or time-varying, and the agents might or might not know their own position or those of the goals. The metric we will consider will always be a notion of *expected latency*, which is the expected time at which a randomly selected goal will be reached.

In abstract terms, let optimization problem P be a "search problem"

$$z_p^* \equiv \min_{x \in X(P)} z_P(x)$$

in which the feasible set, $X(P)$, is a set of *control policies* and the objective, z_p, is expected latency.

In the context of this dissertation, an *algorithm* for search is a function A from a space of search problems, $\mathcal{Q}$, to the space of all control policies, such that $A(P) \in X(P)$. In abstract terms, statements about the *generality* of algorithm A refer to the size of its domain, $\mathcal{Q}$, while statements about its *quality* describe the closeness between $z_p(A(P))$ and z_P^* across this domain.

Understandably, research is almost exclusively focused on the quality of algorithms. The common metric for quality is the worst-case ratio of objectives.

$$\mathrm{Cf}(A) \equiv \max_{P \in \mathcal{Q}} \frac{z_p(A(P))}{z_p^*}. \tag{1.1}$$

Definition 1.1.1 (Constant-Factor Approximation). *An algorithm A for which $\mathrm{Cf}(A) < \infty$ is said to provide a Constant-Factor Approximation to $\mathcal{Q}$.*

An algorithm for which this maximum is one is definitively a "solution" to the set of problems $\mathcal{Q}$. However if $\mathrm{Cf}(A) > 1$, there might be metrics showing a different preference between algorithms, for instance if we were to replace the maximization in Equation 1.1 with an expectation with respect to some measure over $\mathcal{Q}$.

We wish to draw some attention to the tacit trade-off between quality and generality. Given a focus on quality rather than generality, it is natural to see a fracturing of research between even very similar problems. There are situations in which Equation 1.1 can only be improved by constraining $\mathcal{Q}$, rather than by changing A.

Consider, for example, the divergences between algorithms for the many combinatorial optimization problems in the set of problems that are NP-complete. Even though these problems are in some sense equivalent, the Constant-Factor Approximation algorithms proposed for them are generally quite different.

One might say, "Of course this is the case. The quality of approximation doesn't necessarily survive the conversion between problems." In response, we point to the Nearest Neighbor Algorithm as a counter-example.

The Nearest Neighbor Algorithm algorithm is *general* in the sense that it can be applied to a large set of problems. More importantly, we assert that it also has significant *quality*, even if it lacks quality in the sense of Equation 1.1. To support this assertion we point to the large set of problems on which it performs well. For example if we let $\mathcal{Q}$ be instances of the Traveling Salesman Problem (TSP) in the

Euclidean plane we have, from [54],

$$\mathop{\mathbb{E}}_{P\in\mathcal{Q}}\left[\frac{z_P(\mathrm{NN}(P))}{z_P^*}\right] \approx 5/4.$$

For instances of the MLTP with cities chosen uniformly in the Euclidean unit square, our experiments in Section 3.3.1 show that Nearest Neighbor Algorithm puts this notion of *expected quality* closer to 1.03. For the DTRP in this same space, the Nearest Neighbor Algorithm is comparable to the best algorithms in the literature under a range of conditions.[21, 23, 72, 76]

In this dissertation, we will attempt to present a maximally general algorithm for search problems. Of course we want this algorithm to also be of high quality, but that will not be our primary emphasis. Specifically, we make no attempt to minimize Equation 1.1. Instead we will evaluate our algorithm on specific problems of interest that allow comparison to existing work.

This is not to say that we are seeking the most general algorithm possible. We are not trying to construct an algorithm for arbitrary optimization or path planning problems. In particular, we use the word "algorithm" in such a way as to exclude most such approaches. For example, when using a meta-heuristic, or Approximate Dynamic Programming (ADP), there are a number of significant *design decisions* that must be made before one has an implementable procedure. Even k-Nearest Neighbors or, more aptly, Receding-Horizon, without a fixed or algorithmic way of setting k or the horizon, fail to satisfy this definition in the strictest sense.

Although there will remain potentially significant *implementation decisions*, for instance how to represent continuous measures on a discrete machine, we will do our best to present a singular algorithm that does not omit any essential portion of its design. There will nonetheless be situations in which we must circumvent complexity by more-or-less *ad hoc* means. While these complexities introduce what may rightly be considered "design decisions," they center on questions of "how" to approximate rather than "what" to approximate. As a result, we expect that their ramifications are less profound than those associated with more general-purpose methods such as meta-heuristics or ADP.

We do not mean to suggest that focusing on quality, specifically in the form of Equation 1.1, is futile or fallacious. This dissertation simply presents an alternative in which we sacrifice much of our ability to make statements about quality in order to make stronger statements about generality. To the contrary, it is our opinion that the approximation results for NP-complete are impressively comprehensive. That said, the continuous problems that will be our focus are generally much harder and the available guarantees are correspondingly weaker. As a result, quality-focused research has only nibbled around the edges of the problem space into which we are about to bite.

1.2 Preliminaries

Having described the scope and philosophy of this dissertation, as well as establishing some basic terminology, this section introduces a few fundamental formulations that we will be relevant throughout.

1.2.1 Markov Decision Problems

Problem 1.2.1 (Markov Decision Problem (MDP)). *A* Markov Decision Process *is a tuple $\langle X, A, T, R\rangle$. Let X be a state space and A be the set of available actions and let $T : X \times A \to \Omega(X)$ wherein $\Omega(X)$ is the set of probability measures over set X. We also have a reward model $R : X \times A \times X \to \mathbb{R}$ with $R(x_0, a, x_1)$ giving the reward gained on transitioning from state x_0 to state x_1 under action a. The Markov Decision Problem (MDP) is the problem of determining a control policy $\Pi : X \to A$ that maximizes some notion of cumulative reward.*

Bellman, in [15], famously applied dynamic programming to the Markov Decision Problem (MDP) showing that the optimal control policy is the argument-maximizer of a value equation that bears his name, shown in Equation 1.2 for the case in which cumulative reward is measured by an infinite sum with discount factor $\beta < 1$.

$$V^*(x) = \max_{a \in A} \sum_{(x' \in X)} T(x'|x, a)(\beta V^*(x') + R(x, a, x')). \tag{1.2}$$

In particular we will make use of the following well-known theorem.

Theorem 1.2.1 (*e.g.*, [42]). *Every MDP has a stationary optimum in the expected reward case.*

A well-cited result from [71], is that MDPs in general are NP-hard if discrete and PSPACE-hard if continuous. We do not go into detail on these complexity classes beyond pointing out that problems that are NP-hard require time that is exponential in size of their description (the state space, in this case) to solve exactly. It is most likely the case that problems that are PSPACE-hard cannot be solved exactly in finite time and require time exponential in the size of their description to approximate.

A Partially-Observable Markov Decision Problem (POMDP) is the extension of the MDP in which the state is not directly observed.

Problem 1.2.2 (Partially-Observable Markov Decision Problem (POMDP)). *A* Partially-Observable Markov Decision Process *has two additional components, in addition to the elements of the MDP: a set of observations Y and an observation model $O : X \times A \times X \to \Omega(Y)$ with $O(x_0, a_0, x_1)$ measuring the probability of any observation when undergoing a transition from state x_0 to state x_1 under action a_0. The Partially-Observable Markov Decision Problem (POMDP) is the problem of finding a control policy $\Pi : \Omega(X) \to A$ that maximizes some notion of cumulative reward.*

There is an exact equivalence between a POMDP and a so-called "belief state MDP" in which the state space includes a probability distribution over the unobserved states (and the transition model is consistent with the observation model). Since belief over even a pair of discrete states is a continuous variable, POMDPs are PSPACE-hard in general.

A simple example of a POMDP is that from [55], shown in Figure 1-1. The state space is a discrete set of four locations x_i, the actions are the directions $\{\text{Left}, \text{Right}\}$.

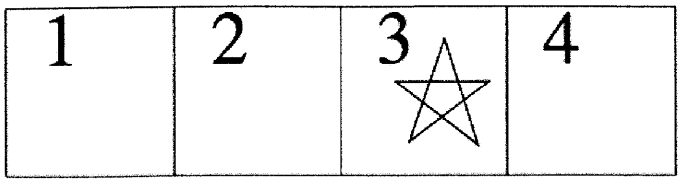


Figure 1-1: Grid world

The motion model is that an action moves the agent in the specified direction, assuming such a state exists, with probability $p = 0.9$. Otherwise the agent stays in its current state. The observation model is that the agent observes the goal if and only if it is in the goal (state x_3) and makes no observation otherwise. Assume that the reward $R(x, a, x') = -1$ and that the goal state is capturing, *i.e.*, the process is terminated when the goal is reached.

For this POMDP each (stationary) policy is a function from the probability simplex $\Omega(\{x_1, x_2, x_4\})$ to the actions $\{\text{left}, \text{right}\}$. The striking feature of this problem is that despite its simplicity, the answer is not trivial, nor is it immediately obvious if we make motion deterministic ($p = 1.0$). One might assume that the optimal policy is to go in the direction for in which the goal is most likely. However it is easy to show that this cannot be optimal for the probability distribution $\{.5 + \epsilon, 0, .5 - \epsilon\}$ for small ϵ.

1.2.2 Path Planning

The example in Figure 1-1 illustrates how the POMDP is a natural model for our problem of interest. However the problem of path planning with uncertainty over a continuous definition of position involves belief over *continuous* variables, which potentially introduces a state space that is not even finite-dimensional.

Despite its difficulty, using the POMDP formulation for path planning has been extensively studied, widely used, and as a result, has had some practical successes in the form of heuristics.

For instance, one approach is referred to as the Augmented MDP (AMDP) which is an example of the strategy of "feature selection" in Approximate Dynamic Programming (ADP). Rather than solving Equation 1.2 over the entire state–belief space, AMDP instead does so over a finite or reduced-dimension projection Φ of the space.

$$V_\Pi(\xi) = \max_{a \in A} \int_{\xi'} \sum_{x,x' \text{ s.t. } \Phi(x)=q, \Phi(x')=q'} T(x'|x,a)(\beta V_\Pi(\xi') + R(x,a,x')). \qquad (1.3)$$

For example, in [81] the authors used features corresponding to the mean and variance of the state distribution. This allowed them to exploit landmarks in order to reliably reach a goal location.

The primary difficulty with feature selection is that there is no algorithmic way of selecting features, and ultimately the success of the approach depends on this selection. For instance, mean and variance alone would not be sufficient for the example in Figure 1-1.

1.2.3 Bandit Problems

In the Multi-Armed Bandit Problem (MABP), a single server must choose from between n processes exactly one on which to work. Each process is an observable Markov chain that, when worked on, or *activated*, evolves and gives reward. When a process is not activated its state does not evolve. The problem is to determine an activation policy that, based on the state of each process, decides which process to activate. This problem is referred to as the "Multi-Armed Bandit" problem because of its relevance to a gambler in front of a group of slot machines.

Gittins famously showed in [43] that there exists an "index function," depending only on the state of a single process, which can be used to greedily solve the problem to optimality. Let X_i denote the state space of process i, let $x_i \in X_i$ denote its current state, and let $\Pi_i \subseteq X_i$ be a set of "stopping states." A "trajectory" τ from x_i to Π_i is a sequence of states starting with x_i and ending with $x_i' \in \Pi_i$, such that the final state x_i' is the first and only such state, *i.e.* $\tau \cap \Pi_i = \{x_i'\}$. Let $R(\tau_j)$ denote the reward received on transitioning from the $j-1$st to the jth state along τ and let

$\beta < 1$ be a discount factor. The Gittins index of process i is given by

$$\gamma_i^*(x_i) = \sup_{\Pi_i} \mathop{\mathbb{E}}_{\tau} \left[\frac{\sum_j \beta^j R(\tau_j)}{\sum_j \beta^j} \right] \tag{1.4}$$

which maximizes expected *reward rate* over the choice of stopping states. The optimal policy is activate the process with the largest index.

The MABP has some relevant limitations. First, the state of unplayed "arms" (*i.e.*, processes) only changes when they are played. Secondly, there is no cost associated with switching between arms. If either of these features are present, the problem is called a Restless Bandit Problem (RBP) and Gittins' indexing policy is no longer optimal[10].

Nonetheless, Whittle in [92] used a a linear programming relaxation to derive an index *heuristic* that is available if the problem has a property that has been dubbed "Whittle Indexability." This heuristic reduces to the Gittins index in the MABP. There is a large and growing body of research that suggests that RBPs with this property are apparently *much easier* than a general MDP of similar size. Specifically, this property gives rise to a relatively simple policy which has a small optimality gap in practice which we describe in Section 1.2.4.

1.2.4 Whittle's Index Heuristic

Problem 1.2.3 (subsidy-γ problem). *Given a single process of a RBP as a two-action MDP, the* subsidy-γ problem *is the otherwise equivalent MDP in which a subsidy γ is added to the reward for the "active action," i.e., activating the process.*

Definition 1.2.2 (Whittle Indexability). *Let $\Pi_i(\gamma)$ denote the set of states of the process for which the active action is optimal in the subsidy-γ problem. Process i is* indexable *if $\Pi_i(\gamma)$ increases monotonically from the empty set to the entire state space as subsidy, γ, increases from $-\infty$ to ∞. An RBP is said to be* indexable *if each process is indexable.*

The Whittle index, γ_i^* *of an indexable process* i *in state* x *is given by*

$$\gamma_i^*(x) = \inf_{x \in \Pi_i(\gamma_i)} \gamma_i. \tag{1.5}$$

Whittle's *index policy* is to always pursue the processes for which γ_i^* is minimum. While non-optimal in general, this heuristic has been extensively examined and has been shown to perform very well empirically, *i.e.*, within a few percent of optimal (see *e.g.*, [5, 18, 44]). Furthermore, the heuristic is asymptotically optimal as the number of arms n goes to infinity and the fraction of playable arms remains constant[90]. As a result, there has been much recent effort into establishing the indexability of classes of problems.

1.3 Organization

This dissertation is organized as follows.

In Chapter 2, we will cover search on lines and rays. Section 2.2 presents our main theoretical result establishing the Whittle Indexability of the CPP and developing the index function that forms the basis of the remainder of the dissertation. Section 2.3 illustrates the power of this result by examining a subtle path planning problem.

Chapter 3 extends the index policy to search in networks in which the edges represent continua. In Section 3.3 we examine combinatorial problems such as the MLTP that can be considered special cases of search in which the locations of the goals are directly observed.

Chapter 4 extends the index policy to search in length spaces. Section 4.2 covers the LFP for which only a few cases have been considered and we compare the index policy to these. Section 4.3 presents a fundamentally novel approach to problems of coverage. In Section 4.4 we consider the ramifications of relaxing our assumptions on the agent's dynamics and consider the problem of *optimal control* for systems that satisfy our other assumptions.

Chapter 5 considers problems in which goals are added to the environment by a spatio-temporal random process. Section 5.3 considers the dynamic version of the coverage problem which has only recently been broached. Sections 5.5–5.6 extend the DTRP to length spaces and Section 5.7 defines the Dynamic Search and Repair Problem (DSRP) which adds the element of search to the DTRP. In Section 5.7 we examine the performance of the index policy on the DTRP and find it to outperform the Nearest Neighbor Algorithm on a range of instances for which better results have not been presented. In Section 5.8 we consider a practical application which we believe underscores the contribution of this dissertation. Specifically we consider an instance of the DSRP surveilling a network of roads using Unmanned Arial Vehicles (UAVs). Finally, we describe a complete implementation which has been fully integrated into a control architecture in use by the Air Force Research Lab (AFRL).

Chapter 6 summarizes the contribution of this dissertation and enumerates a few of the ways in which it could be extended and improved. The most obvious of these would be to extend the theoretical characterization of the algorithm we present and more accurately demarcate the set of problems on which it performs well or poorly. In particular, this includes proving some of the conjectures with which this dissertation is seeded. At the same time we are pessimistic about the extent to which this is possible without deep and fundamental breakthroughs. We are much more optimistic about applications in which this work can be applied, validated, and extended. In particular, we discuss the possibility of incorporating the search policy we develop into a more general ADP approach.

Chapter 2

Search on Rays

This chapter considers problems in which agents move in a one-dimensional space, which we call the "region," specifically a set of rays with a common origin: $\mathcal{A} \equiv \{1, \ldots, n\} \times \mathbb{R}_+$. The problems we consider will all be static, *i.e.*, the set of goals does not change over time.

In this chapter we will develop the policy that is the main contribution of this dissertation. We do this by examining a particular search problem and proving that Whittle's index policy can be applied to it. For this particular problem, then, the policy is not novel; what is important is the proof of applicability of a well-vetted policy. To illustrate the significance of this policy we examine a traditional navigational technique which has largely resisted quantification. Having done this, the remainder of this dissertation will be devoted to the logical extension of this policy to more complex problems.

This chapter is organized as follows. In Section 2.1 we introduce the relevant problems formulations and review existing scholarship. Section 2.2 proves the main result of this chapter, namely that the Cow Path Problem (CPP) is Whittle-indexable and presents the index policy. Section 2.2.4 provides a number of examples attesting to the good performance of the index policy for the CPP and the Line Search Problem (LSP), a special case. In Section 2.3 we illustrate the significance of this result by using it to examine a traditional navigational technique, referred to as aiming off, which is well-known to human navigators but is not produced by any existing path planning algorithms.

2.1 Preliminaries

2.1.1 The Line Search Problem

In [17] Bellman posed the following problem

Problem 2.1.1 (Line Search Problem (LSP)). *Suppose that we know that a particle is located in the interval $(x, x+dx)$, somewhere along the real line $-\infty < x < \infty$ with a probability density function $g(x)$. We start at some initial point x_0 and can move in either direction. What policy minimizes the expected time required to find the particle, assuming a unit speed and*

1. *assuming that the particle will be recognized when we pass x, or*

2. *assuming that there is a probability $p > 0$ of missing the particle as we go past it?*

Also, what would be the optimum starting point x_0?

We will restrict our attention to Case 1 and refer to it as the Line Search Problem (LSP).

Significant early results on the LSP originate with [38] and are elaborated upon by Beck *et al* [12, 14, 13, 11]. The most significant achievement is proving, from first-order optimality conditions, that the LSP on differentiable distributions reduces to the problem of finding the first turn-around point.

In [13], the authors analytically derived the optimal search pattern for uniform distribution on $[a, b]$, and the symmetric triangular distribution on $[-1, 1]$ starting from zero. They were also able to numerically find a policy for searching the unit normal distribution, again starting from zero. In Section 2.2.6 we will review these results in detail. For the Gaussian distribution with density ϕ the first-order optimality conditions give the following recursive partial differential equation (PDE), wherein x_i

is the absolute value of the ith turning point.

$$\frac{\partial x_{n+1}}{\partial x_n} = (x_n(x_n + x_{n+1}) - 2)\frac{\partial x_n}{\partial x_{n-1}} - \frac{\phi(x_{n-1})}{\phi(x_n)}\frac{\partial x_{n-1}}{\partial x_{n-2}} \tag{2.1}$$

Given $x_0 = 0$ and a guess of x_1 the system in Equation 2.1 can be solved recursively. Values of x_1 that are too low result in sequences that are eventually non-increasing and those that are too high create sequences that diverge, *i.e.*, go to infinity in one direction after a finite number of turns. As a result it was possible to search for the optimum path simply by searching over x_1. This search is made difficult only by the remarkable numerical sensitivity of this process. The authors report that changing x_1 by 10^{-30} changes x_7 by nearly 3. On the computers of the day, finding an answer that was stable beyond the eighth turn was an impressive feat. Using this approach to find the optimum starting point, at least at the time, would have been unrealistic.

The authors of [89] extend the numerical approach of [13] to any Lipschitz-continuous measure having finite absolute first moment. Furthermore they show that it is not necessary to search on x_1. Instead they solve System 2.2.

$$H_i = \begin{cases} 1 + G(x_{i-1}) - G(x_i) \\ G(x_{i-1}) - G(x_i) - 1 \end{cases}$$
$$x_1 = x_0 + 0.5/g(x_0)$$
$$x_{i+1} = x_i - H_i/g(x_i) \tag{2.2}$$

wherein G denotes the cumulative distribution of g. Having done this (and aided by higher-precision arithmetic tools), they are able to search on x_0 to solve the problem of the optimum starting point, a problem that we will examine at length in Section 2.3.

In [25, 89, 56] the authors, apparently independently, solve the LSP for distributions consisting of n atoms in $O(n^2)$ time using dynamic programing. The key insight is that state space can be represented as the current path of the agent and the *frontiers*, or limits to which the search has already explored. Observing that the sequence of frontiers is non-repeating and the policy at the "boundary" is trivial (*i.e.*, if one direction has been completely searched, search in the other direction), Bellman's equation can be solved exactly in one pass through the state space by starting at the boundary and moving the frontiers toward the starting state.

In [3] the authors present a discretization scheme for continuous distributions that gives a provable, arbitrary approximation when used with the above dynamic programming approach. This is an important and surprising result, essentially establishing that the LSP is *not* PSPACE-hard.

Their result is as follows. Let $\overline{g}^1 < \infty$ denote the absolute first moment of the distribution, and $|G|^{-1}$ be the inverse cumulative of its absolute value. For a provable approximation of $1 + \epsilon$ to the continuous LSP, one can use the approach from [25] with a discretization fidelity of $\epsilon\overline{g}^1/64$ and truncate the distribution at $r = |G|^{-1}(1 - \epsilon\overline{g}^1/24)$. If this search fails, the policy is to pursue a doubling strategy in which the ith turn after completing the discretized search is at $x_i = 2^i r$.

2.1.2 The Cow Path Problem

A generalization of the LSP from the real line to a set of rays is called the Cow Path Problem (CPP). In the CPP, m agents are searching for a unique goal that lies on one of n rays diverging from a single origin (with $m < n$). We will call this set of locations the "region," $\mathcal{A} = \{1 \dots n\} \times \mathbb{R}_+$. The agents know their own positions, and each has a sensor that can detect the goal only if the agent is at the location of the goal; otherwise it gives no information.

When the literature shifted the discussion from the LSP to the CPP there was a subtle but important shift in the objective. While Bellman's problem asks for minimum expected time, beginning primarily with [8], the contemporary objective is different.

In search problems such as the CPP, the most widely accepted theoretical framework, nowadays, is competitive analysis (see [1, 52] for surveys) and not minimum expectation. In this framework, the goal is to determine a search strategy that minimizes the *competitive ratio*, which, informally for the moment, can be thought of as a worst-case constant-factor guarantee.

A notable exception is [56], who point out that by assuming a prior distribution over the goal location, it is quite natural to pose the CPP as a path planning problem. Those authors point out that if the distribution has discrete, finite support, one can apply algorithms for the discrete MDP in an efficient fashion. In fact it is possible that approach from [3] can still provide an arbitrary, provable approximation. However, the underlying solution of these methods scales exponentially with the number of paths. In particular, with k discrete points on n paths with m agents, the dynamic programming approach must construct $O\binom{n}{m}$ tables with $O(k^{\binom{n}{m}})$ values each.

2.1.3 Competitive Analysis

In the usual formulation of the CPP (see, *e.g.*, [37] and the references therein), the objective is to minimize the competitive ratio, Cr, of the search strategy. The competitive ratio of an on-line algorithm A is given by

$$\mathrm{Cr}_A = \sup_{P \in \mathcal{Q}} \frac{c(A(P))}{c^*(P)}, \tag{2.3}$$

where $\mathcal{Q}$ denotes the set of all problem realizations, $c(A(P))$ is the cost of the solution returned by A applied to realization P, and $c^*(P)$ is the cost that could be found by an optimal offline algorithm with full knowledge of P in advance.

Note the difference between Equations 2.3 and 1.1: c^* need not be achievable by a control policy, while the optimum z^* is achievable by definition. In fact, c^* will *never* be acheivable for any non-trivial search problem.

For example, consider the case with one agent and two paths and assume that the goal location is known to be integer. From [1], the best-possible[1] schedule of turn-around points, $z(i)$, is $z(i) = (-2)^i$. If the goal is at distance n from the origin, the search takes $2(\sum_{i=0}^{\lfloor \lg n \rfloor + 1} 2^i) + n$, wherein lg denotes the base-2 logarithm. The optimal offline algorithm would be able to travel straight there, so $c^* = n$. The competitive

[1]An on-line algorithm with the minimum achievable competitive ratio is called "best-possible" rather than "optimal" to avoid the confusion between c^*, in Equation 2.3 and z^* in Equation 1.1.

ratio is therefore

$$\sup_n \frac{2(\sum_{i=1}^{\lfloor \lg n \rfloor + 1} 2^i) + n}{n} = 9.$$

In contrast, in this dissertation we will try to minimize the *expected* distance travelled before reaching the goal. To do this we must have a probability distribution, $\phi : \mathcal{A} \to \mathbb{R}_+$, for the goal location. As pointed out by [56], given such a prior distribution, it is straightforward to pose the CPP as a path planning problem in the MDP framework.

One of the arguments in favor of competitive analysis is that it does not have the requirement of a prior. However, if one takes a Bayesian view of probabilities, a prior is required regardless. Philosophical arguments aside, we would argue that the Bayesian requirement of a prior is, at least, *not unreasonable*—the region is a well-defined set and there is nothing to prevent one from using an uninformative prior.

Regardless of how informative this prior distribution may be, this formulation is meaningfully distinct from that in Equation 2.3. Minimizing the expectation for the uninformative distribution is seldom equivalent to minimizing the competitive ratio. We illustrate this difference by example.

Consider the uniform distribution over the integers $-n, \ldots, n$. In the limit of $n \to \infty$, we have already discussed the best-possible competitive strategy, which turns around at $z(i) = -2^i$. This strategy searches for an expected distance of $33n/8$. In the framework of expected distance, the optimal policy for the uniform distribution is given by [13]: For any finite n, the optimal policy simply searches each direction to the end. This strategy only searches for an expected distance of $3n/2$—less than half as far.

Remark 2.1.1. *For deterministic algorithm A, with competitive ratio Cr_A, there is a probability measure over instances $\mathcal{Q}$ such that $\mathop{\mathbb{E}}_{P \in \mathcal{Q}} [c(A(P))]$ is arbitrarily close to $\mathrm{Cr}_A \times c^*$. This is accomplished by assigning probability one to the instance (or an instance in the sequence) that maximizes Equation 2.3. In other words, the competitive ratio provides a* tight *bound on performance in the worst-case over* probability distributions.

It has been noted (see *e.g.*, [62]) that under competitive analysis, the performance of an algorithm can be improved by randomization. This should further illustrate the difference between the objectives: In the MDP framework this would constitute direct contradiction to Theorem 1.2.1. The reason for this improvement is that it makes the construction in Remark 2.1.1 impossible. For example, consider the policy $z(i) = \pm(-2)^i$ with the sign chosen randomly. The worst-case over probability distributions now puts support on both positive and negative n and the search will have expected length $15n/2$ rather then $9n$.

2.1.4 Search Games

Another interesting formulation for search is game theoretic. There are two players, a hider and searcher. A (mixed) hiding strategy H is a probability measure over the region and a searching strategy S is a probability measure over paths that cover the region. The game is zero-sum: The hider's payout is the time at which he is found and the searcher's payout is its negative. This formulation can be applied to a broader set of spaces and we will revisit it in subsequent chapters.

This formulation does not give search strategies for arbitrary measures. Instead it finds strategy–measure pairs that are Nash equilibria. These equilibria are of interest because they represent the worst cases over probability measures for an optimal search policy (with respect to expectation).

For search games on weakly-Eulerian graphs, [40] shows that these equilibria can be readily found. At these equilibria, the search strategy S is uniform over the set of minimum edge covers. The hiding strategy H puts atoms on the leaves and uniform distributions on Eulerian sub-graphs.

For more complex graphs, finding equilibria quickly becomes intractable. The authors of [40] approximate the equilibria for the graph consisting of three unit edges connecting two nodes, but it is non-trivial. When generalized to the graph with k (odd) edges between two nodes, their approach has complexity in $\Omega(e^k)$.

To illustrate the distinction between the game theoretic formulation and the competitive formulation recall the example of searching the integers $-n, \dots, n$, discussed in the previous section. This equilibrium corresponds to probability 0.5 on $\pm n$, and a search strategy that searches each direction to the end, picking the initial direction randomly. The expected length of this search is $2n$.

By way of enunciation, the game formulation provides the worst-case over measures for the policy with minimum expected search time. When randomized algorithms are considered, the competitive formulation provides the worst-case over measures of the expected ratio of search time to the distance from the searcher's starting point to the hiding location.

Hence, the competitive formulation is often cast as the game theoretic formulation in which the hider's reward (and the negative of that of the searcher) is the ratio of discovery time to this initial separation. In an unbounded region, this is a natural response to the fact that the hider could achieve unbounded reward under the initial formulation.

2.2 Indexability of the CPP

In this section we examine the CPP from the minimum expectation perspective. We assume that we have a prior distribution for the location of the goal in the form of cumulative distributions F_i for each path i.

As we discussed in Section 1.2.1, the main source of difficulty stems from the fact that the state must include a measure over goal locations. In the CPP, however, the belief has a special structure that can be exploited in the solution of the problem. Due to the limited sensor model, either the goal is found or the posterior belief is the prior distribution with explored regions zeroed-out (and re-normalized). Hence for a given prior distribution, the belief can be represented by the limits of the explored regions, which we will refer to as the "frontier" z, which is simply a vector in $\mathbb{R}^n_+$. The key observation is that the time required to switch between the frontiers on different paths is the sum of a "tear-down" and a "set-up" cost. Given this observation, the result from [45], although not directly applicable, is strongly suggestive of the Whittle Indexability of the CPP.

Results establishing indexability, while not requiring that unplayed arms do not evolve, universally require that the arms evolve *independently.* This property is also absent from the CPP. There is only one goal; finding it on a particular path tells a great deal about whether it will be found on a different path. In this section we prove the indexability of what we will call the Poisson CPP in which we do not insist that there is exactly one goal. In particular, we assume that the probabilities of there being a goal in disjoint sub-regions are independent, but that the *a priori* expected number of goals in the region is one.

We will establish indexability of the Poisson CPP by constructing a subsidy scheme for each branch that satisfies Definition 1.2.2. To do this, we assume that an agent pays a unit cost per unit distance moved, and we determine a system of subsidies that make it neutrally profitable for the agents to conduct the search.

We will define our subsidy, γ, such that an agent is paid only if it finds the goal. An equivalent formulation (which would be more semantically consistent with [92, 45]) would be one in which the reward is unitary and we penalize motion. We use this definition entirely for intuitive appeal: If we think of the goal as a commodity, say, a quantity of natural gas, then γ would be its "price". Posed as such, the Whittle index will represent the price for gas at which exploration becomes profitable in expectation. This intuition will become increasingly valuable in subsequent chapters when we attempt to generalize our policy to problems which are not Whittle Indexable.

Remark 2.2.1. *Trivially, for $\gamma < 0$ the set of states for which it is profitable to move is surely empty since the agent receives no other rewards.*

2.2.1 Switching Costs

Examining the CPP as a RBP, the "state" corresponds to that of the processes, as opposed to that of the agents. Specifically, "paths" naturally correspond to processes and we define the state of path i as the pair $(a_i, z_i) \in \{0,1\} \times \mathbb{R}_+$ where a_i indicates whether the path is being actively searched, and z_i denotes the "frontier" of the path. This lets us define the state of the entire problem as $(a, z) \in \{0,1\}^n \times \mathbb{R}_+^n$. For the treatment that follows, we will not explicitly track the locations of the agents and instead assume that there is an agent at frontier of path i if and only if $a_i = 1$.

If an agent switches from path i (with state $(1, z_i)$) to path j (with state $(0, z_j)$) the agent must move a distance $z_i + z_j$ before it can activate path j. This is problematic because we would like the subsidy to depend only on the state of the path being activated. We will address this problem by changing the costs analogously to [44], which has an intuitive interpretation in our formulation.

We change the costs as follows. For each active path the agent must maintain a "return counter" that always contains the cost of returning to the origin. This way when path i is abandoned, the cost of activating path j is only z_j. This reflects the true costs assuming that the agent will eventually return to the origin, which does not happen if the agent finds the goal. Therefore, we also must adjust the reward as follows. If an agent finds the goal, it gets the payment γ plus the value in the return counter.

Lemma 2.2.2. *Assume that there is a non-zero probability of finding the goal on path i.*[2] *From any state, there exists a sufficiently large, finite subsidy $\gamma < \infty$, for which it is profitable to explore path i.*

Proof. Let $F_i(z_i)$ denote the cumulative probability of finding a goal at or before z_i on path i.

The expected cost of exploring path i to distance d is upper-bounded by $2d$. The expected reward will be $(F(d) - F(z_i))\gamma$, which is positive for some $d > z_i$, by assumption. For such a d, we can set $\gamma > 2d/(F(d) - F(z_i))$ which guarantees that it is profitable to explore path i. □

2.2.2 Subsidy Scheme

We are interested in finding the minimum subsidy, $\gamma_i^*(a_i, z_i)$, at which path i in state (a_i, z_i) becomes profitable[3]. Since we will only be considering a single path at a time,

[2]If there is zero probability, we are free to remove the path from the problem altogether.

[3]To avoid excessive use of infima, we will use the word "profitable" to encompass the case in which the expected profit is zero.

we will drop the subscripts.

For an agent exploring path with state (a, z) we can compute the expected cost of searching to frontier z'. For notational compactness let $F(x_1, x_2)$ denote $F(x_1) - F(x_2)$.

$$\mathbb{E}\left[c((a,z),z')\right] = 2(1-a)z + \alpha\left(\int_z^{z'} \chi dF(\chi) + (1 - F(z',z))2(z'-z)\right) \tag{2.4}$$

In the original version of the problem, we knew that there is exactly one goal. Therefore $\alpha = (1 - \sum_j F(z_j))^{-1}$ is a normalization term that depends on the the states of other paths. For the Poisson CPP, $\alpha = 1$ and we will subsequently drop it.

This lets us define the minimum subsidy under which any search is immediately profitable

$$\gamma^*(a,z) \equiv \inf_{z'>z} \frac{\mathbb{E}\left[c((a,z),z')\right]}{F(z',z)}. \tag{2.5}$$

We now define the states, $\Pi(\gamma)$, for which search is immediately profitable under γ.

$$\Pi(\gamma) \equiv \{(a,z) \text{ s.t. } \gamma \geq \gamma^*(a,z)\} \tag{2.6}$$

Remark 2.2.3. *It is clear from Equation 2.6 that* $\gamma_0 \leq \gamma_1 \implies \Pi_i(\gamma_0) \subseteq \Pi_i(\gamma_1)$.

We proceed by deriving the optimal policy in the subsidy-γ problem. We can write Bellman's equation for the the expected reward of the optimal policy, V_γ^*, when started from state (a, z). Note that unlike Equation 1.2, Equation 2.7 uses the non-discounted form.

$$V_\gamma^*(a,z) = \max\left(0, \sup_{z'>z}\Big(\gamma F(z',z) - \mathbb{E}\left[c((a,z),z')\right] + (1 - F(z',z))V_\gamma^*(1,z')\Big)\right) \tag{2.7}$$

Lemma 2.2.4. *Assume* $V_\gamma^*(a,z) > 0$. *For the frontier,* z', *that maximizes (or is a limiting value of the sequence that maximizes, potentially* z *or* ∞*) Equation 2.7,*

$V_\gamma^*(1,z') = 0$, *i.e.*,

$$V_\gamma^*(a,z) = \max\left(0, \sup_{z'>z}\left(\gamma F(z',z) - \mathbb{E}\left[c((a,z),z')\right]\right)\right). \tag{2.8}$$

Proof. Suppose to the contrary that $V_\gamma^*(1,z') > 0$ which is maximized by $z'' > z'$, *i.e.*,

$$\begin{aligned} V_\gamma^*(1,z') &= \gamma F(z'',z') - \int_{z'}^{z''} \chi dF(\chi) + \\ &\quad (1 - F(z'',z'))(2(z' - z'') + V_\gamma^*(1,z'')) \end{aligned}$$

Expanding Equation 2.7 and collecting terms we have

$$\begin{aligned} V_\gamma^*(a,z) &= \gamma F(z'',z) - 2(1-a)z - \int_z^{z''} \chi dF(\chi) + && (2.9) \\ &\quad (1 - F(z'',z))(2(z - z'') + V_\gamma^*(1,z'')) - && (2.10) \\ &\quad F(z',z)\left(\gamma F(z'',z') - \int_z^{z''} \chi dF(\chi)\right) + && (2.11) \\ &\quad F(z'',z')(1 - F(z,z'))2(z' - z'') && (2.12) \end{aligned}$$

From our assumption that $V_\gamma^*(a,z') > 0$, term 2.11 must be negative. The term 2.12 is non-positive and zero only if there is no probability of finding the goal between z' and z''. We remove these terms and arrive at the inequality:

$$\begin{aligned} V_\gamma^*(a,z) &< 2(1-a)z + \gamma F(z'',z) - \int_z^{z''} \chi dF(\chi) + \\ &\quad (1 - F(z'',z))(2(z - z'') + V_\gamma^*(1,z'')). \\ &= \gamma F(z'',z) - \mathbb{E}\left[c((a,z),z'')\right] + V_\gamma^*(1,z'') \end{aligned}$$

Since z', rather than z'', maximizes Equation 2.7, we have the following contradiction

$$V_\gamma^*(a,z) \geq \gamma F(z'',z) - \mathbb{E}\left[c((a,z),z'')\right] + V_\gamma^*(1,z'')$$

□

Lemma 2.2.5. *The optimal policy in the subsidy-γ problem is to search if and only* *if* $x \in \Pi(\gamma)$.

Proof. We begin with necessity, proving that it is optimal not to search if $x \notin \Pi(\gamma)$

Substituting Equation 2.5 into Equation 2.8 and letting $\gamma = \gamma^* + \delta_\gamma$,

$$V_\gamma^*(a,z) = \max\left(0, \sup_{z'>z}\left(\delta_\gamma F(z',z) - \mathbb{E}\left[c((a_i,z),z_i')\right] + \inf_{z''>z} \frac{\mathbb{E}\left[c((a,z),z'')\right]}{F(z'',z)} F(z',z)\right)\right) \tag{2.13}$$

Noting

$$\inf_{z''>z} \frac{\mathbb{E}\left[c((a,z),z'')\right]}{F(z'',z)} \leq \frac{\mathbb{E}\left[c((a,z),z')\right]}{F(z',z)}, \tag{2.14}$$

we can conclude that

$$V_\gamma^*(a,z) \leq \max\left(0, \sup_{z'>z}(\delta_\gamma F(z',z))\right). \tag{2.15}$$

Since $x \notin \Pi(\gamma)$ implies $\delta_\gamma < 0$, we can conclude that $V_\gamma^* x = 0$. Therefore the passive action is optimal.

We now prove sufficiency, showing that for $x \in \Pi(\gamma)$ it is optimal to search. We divide this into two cases.

Case 2.2.5.1. *Let* $x \in \Pi(\gamma)$ *and assume that the infimum in Equation 2.5 is achieved by some* $z'' > z$

Proof of Case 2.2.5.1. By assumption,

$$\gamma^*(a,z) = \frac{\mathbb{E}\left[c((a,z),z'')\right]}{F(z'',z)}.$$

The supremum in Equation 2.13 is over z' rather than z''; hence

$$V_\gamma^*(a,z) \geq \max\left(0, (\delta_\gamma F(z'',z))\right).$$

Since $x \in \Pi(\gamma)$ implies $\delta_\gamma \geq 0$, we can conclude that the active action is optimal. □

Case 2.2.5.2. *Let $x \in \Pi(\gamma)$ and assume that the infimum in Equation 2.5 is achieved by the limit of some sequence $z^i = z^\infty$. We divide this into three sub-cases.*

Case 2.2.5.2.1. $z^\infty = \infty$. *In this limit, the cost of searching to z^i is unbounded, therefore γ^* must be unbounded and δ_γ cannot be positive. This case cannot occur.*

Case 2.2.5.2.2. $z < z^\infty < \infty$

Proof. The expected cost of searching from z to $z' > z$ is continuous in z'. Hence the cost of searching the open interval $[z, z^\infty)$ is equal to that of searching its closure. At the same time, the expected reward is non-decreasing in z'. Therefore this case entails Case 2.2.5.1. □

Case 2.2.5.2.3. $z^\infty = z$

Proof. We rewrite Equation 2.14, with $\lim_{i\to\infty} z^i = \lim_{dz\to 0^+} z + dz$

$$\gamma^*(a,z) = \lim_{dz\to 0^+} \frac{\mathbb{E}\left[c((a,z), z+dz)\right]}{dF(z)}$$

For this limit to be meaningful, we must assume that

$$\phi(z) \equiv \frac{dF(z)}{dz}$$

is well-defined.

Since $\sup_{x>0} g(x) \geq \lim_{x\to 0} g(x)$ we can rewrite Equation 2.13 as

$$\begin{aligned} V_\gamma^*(a,z) \geq \quad & \max\Bigg(0, \lim_{dz\to 0^+}\Bigg(\delta_\gamma\phi(z)dz + \gamma^*(a,z)\phi(z)dz - \\ & \qquad \frac{\mathbb{E}\left[c((a,z), z+dz)\right]}{\phi(z)dz}\Bigg)\Bigg) \\ \geq \quad & \max\Big(0, \lim_{dz\to 0^+} \delta_\gamma\phi(z)dz\Big). \end{aligned}$$

Since ϕ and dz are non-negative, $\delta_\gamma \geq 0$ implies that it is optimal to search. □

Since these cases are exhaustive, this concludes the proof of Lemma 2.2.5 □

Theorem 2.2.6. *The Poisson CPP is indexable.*

Proof. Lemma 2.2.5 proves that $\Pi(\gamma)$ (Equation 2.6) is exactly the set of states for which the active action is optimal in the subsidy-γ problem. From Remark 2.2.1, Lemma 2.2.2, and Remark 2.2.3 we have established that $\Pi(\gamma)$ increases monotonically from the empty set to the entire state space as γ goes from $-\infty$ to ∞, satisfying Definition 1.2.2. □

Corollary 2.2.7. *The Whittle index (defined in Equation 1.5) is exactly $\gamma^*(x)$, given by Equation 2.5. This follows directly from the definition of $\Pi(\gamma)$ given by Equation 2.6.*

Policy 2.2.1. *The* index policy *is to pursue the paths for which γ^* is minimized.*

2.2.3 An Extension

Recall that this section proves the indexability of a modified version of the CPP. Specifically we replace the prior over the goal location with a probability measure for the presence or absence of goals that assumed spatial independence. These probability distributions are initially identical, but as the frontier is pushed back, the posterior distribution of the former is re-normalized while that of the latter is not.

If the problem of interest is the unmodified problem we propose the following simple extension to the index policy: Before computing the index, renormalize the distribution. That is, if the frontier is z, one can multiply all the probability distributions by $(1 - \sum_i F(z_i))^{-1}$.

This extension also represents the first in a sequence by which we will generalize the index policy to a larger space of problems than the Poisson CPP. We expand the index policy to the set of measures that are "separable." Suppose it is the case that at any particular time t, the probability that there are n goals in set X is equal to

$$\sum_{i=n}^{\infty} f_t(i) \binom{i}{n} \Phi_t(X)^n (1 - \Phi_t(X))^{i-n}$$

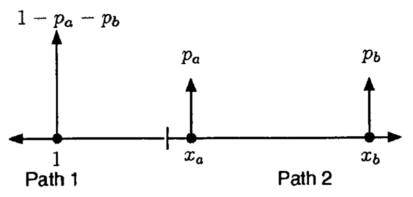


Figure 2-1: Prior belief for three impulses on two paths.

for some Φ_t and f_t with $\Phi_t(\mathcal{A}) = 1$.

For such cases, we will use the index policy with measure $\mu(X) = (1 - f_t(0))\Phi_t(X)$ as the spatial distribution, to minimize the expected time at which the *first* goal is found.

2.2.4 Examples

Toy example

From the nature of PSPACE-hard problems, any numerical evaluation is going to be essentially anecdotal. This is precisely why we have embraced the literature on Whittle Indexability. Nonetheless, we think the following example provides an intuition while examining an interesting and non-negligible portion of the problem space. This example will show the heuristic's non-optimality, but will also demonstrate that it is a good approximation.

Suppose that a single agent is searching two paths and the prior belief consists of three impulses, as shown in Figure 2-1. Note that one impulse is placed at unit distance without loss of generality.

First, for simplicity, we let $p_a = p_b = 0.25$ and let $x_b = 2 - x_a$. Let $\pi_{12}, \pi_{21}, \pi_{212}$ denote the three possible search policies that search according to the order of their subscripts. We compute the expected latencies under each policy.[4]

$$\begin{aligned}
c_{12} &= (1-p_a-p_b)+p_a(2+x_a)+p_b(2+x_b)=2 \\
c_{21} &= p_a(x_a)+p_b(x_b)+(1-p_a-p_b)(2x_b+1) \\
&= 3-x_a \\
c_{212} &= p_a(x_a)+(1-p_a-p_b)(2x_a+1)+ \\
&\quad p_b(2x_a+2+x_b) \\
&= 1.5+1.5x_a.
\end{aligned}$$

We compute the minimum subsidies

$$\begin{aligned}
\gamma_1^* &= \frac{(1-p_a-p_b)+(p_a+p_b)2}{1-p_a-p_b} = 3 \\
\gamma_2^* &= \min\Bigg(\frac{p_a x_a+(1-p_a)2x_a}{p_a}, \\
&\qquad \frac{p_a x_a+p_b x_b+(1-p_a-p_b)2x_b}{p_a+p_b}\Bigg) \\
&= \min\left(7x_a, 5.5-2x_a\right).
\end{aligned} \tag{2.16}$$

We will later refer to the first and second terms of Equation 2.16 as γ_{2a} and γ_{2b}, respectively.

If we explore path 2 to x_a, γ_2^* will change to

$$\gamma_{2ab} = \frac{p_b(x_b-x_a)+(1-p_b)2(x_b-x_a)}{p_b} = 14(1-x_a).$$

Comparing these, we can see that policy π_{212} is optimal for $0 \leq x_a \leq 1/3$ and policy π_{12} is optimal for $1/3 \leq x_a \leq 1$. Comparing the subsidies we see that the index policy is optimal except for $1/3 < x_a < 3/7$. In this range the worst case is 15/14 of optimal, a sub-optimality of about 7%.

We now remove the restrictions on p_a, p_b and x_b and consider all possible priors over three discrete locations on two paths. We will optimize over the locations and

[4]Note that the expected latencies are those from the original problem with exactly one goal.

priors in order to find the greatest sub-optimality.

From this we construct six optimization problems: one for each possible ratio of the latency of policies. Each problem is subject to constraints on γ_1, γ_{2a}, γ_{2b}, and γ_{2ab} such that the numerator is the strategy chosen by the index policy. By way of example, Problem 2.17 assumes that policy π_{21} is optimal, but the index policy instead pursues π_{12}.

$$\begin{aligned} \max_{x_a,x_b,p_a,p_b \geq 0} \frac{c_{12}}{c_{21}} \quad \text{s.t. } & \gamma_1 < \gamma_{2a}, \quad \gamma_1 < \gamma_{2b}, \\ & x_a \leq x_b, \quad p_a + p_b \leq 1 \end{aligned} \tag{2.17}$$

To solve these problems we used Matlab's FMINCON search using the "active-set" algorithm, with 400 random restarts. These were divided into four sets of one-hundred cases, where starting points x_a, x_b were chosen from an exponential distribution with scale parameters of 10, 100, 1000, or 10000. The largest sub-optimality was 38% which occurs at $x_a = 7.2$, $x_b = 23.2$, $p_a = 0.56$, and $p_b = 0.33$ and when the index policy chooses strategy π_{212} although π_{12} is optimal. This same optimum was consistently found.

Of course, this does not prove a lower-bound. However, it does show that if it *is* possible to achieve greater than 38% sub-optimality, such cases must be rare if we are unable to find them using this search technique.

We compare this to the worst-case sub-optimality (over x_a, x_b, p_a, p_b) of the best-possible competitive algorithm. It is easy to show that for $x_a = x_b = 1$, the best-possible competitive ratio is three. From Remark 2.1.1, p_a, p_b can be chosen such that this algorithm will be a factor of three from optimal.

Compared to a factor of three, we can interpret a 38% sub-optimality as strong performance. Notably however, it a substantially greater level of sub-optimality than is commonly reported in numerical studies on indexable problems[5, 18, 44].

2.2.5 Comparison to Competitive Policy

In this section we compare the best-possible competitive policy to the index policy for distributions with continuous support. It is clear that the index policy stands to benefit from knowledge of the prior. To provide the most fair comparison, we will use the least informative distributions: uniform and negative exponential.

Since these distributions are continuous at zero we must modify our definition of the competitive ratio. If we discretize the region with fidelity ε, the best-possible deterministic policy (from [9]) turns at

$$\left(\frac{m}{m-1}\right)^{i}\varepsilon. \tag{2.18}$$

For the analysis here we consider the best-possible competitive policy limit of $\varepsilon \to 0$, in which the agent travels a distance of $2m|x|$ before turning at x.

The uniform prior with infinite support is improper. Instead, we consider uniform probability over finite support ℓ while taking the limit

$$\lim_{\ell\to\infty} \frac{\mathbb{E}_{x\sim U(0,\ell)^m}[w(x)]}{\ell}$$

wherein $w(x)$ denotes the time at which location x is first reached. For the index policy, this evaluates to $(m-1/2)$. For the best-possible competitive policy this limit evaluates to $47/18$ at $m=2$. For higher values of m we resort to numerical simulation, the results of which are presented in Figure 2-2.

Figure 2-3, shows simulation results for the exponential prior. In this case we immediately resort to simulation, using importance sampling to improve the estimate.

In both cases, the index policy significantly outperforms the competitive policy. This is notable because we selected priors that we expected to most favor the competitive policy. These results illustrate the fact that, even with uninformative priors, minimizing the competitive ratio does not minimize the expected wait time.

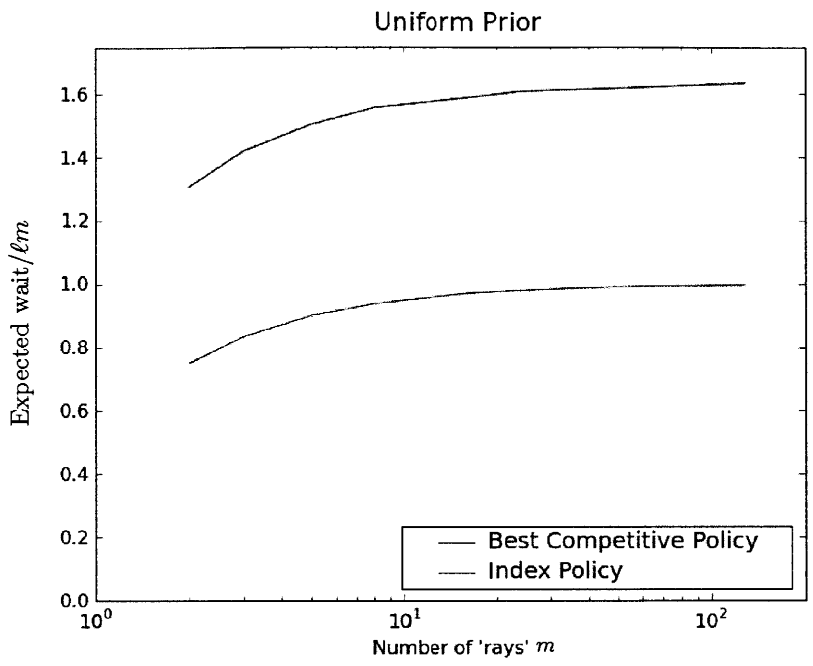


Figure 2-2: Expected wait times for the uniform prior

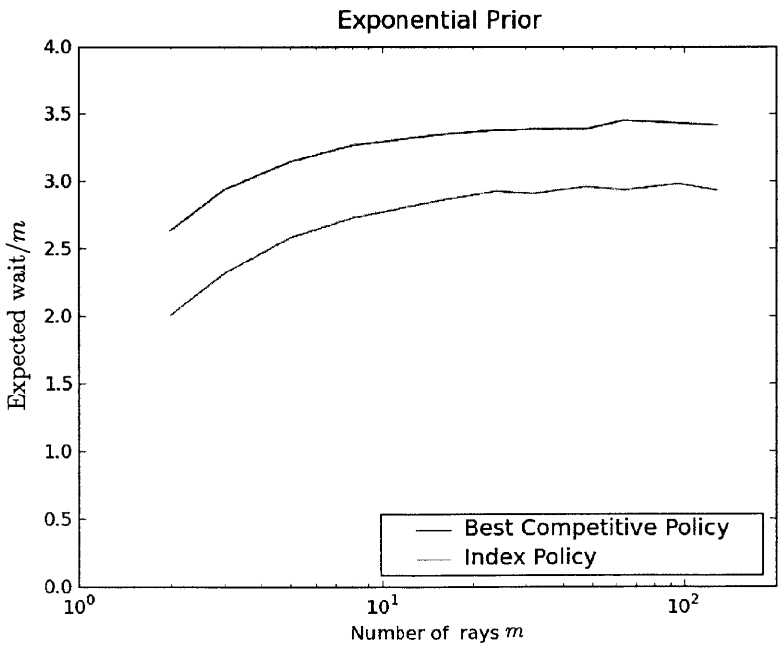


Figure 2-3: Expected wait times for the exponential prior

	[13]	Index Policy
x_1	0.6561	0.5968
x_2	0.9697	0.9525
x_3	0.9998	0.9994
$x_i, i \geq 4$	1.0	1.0
mean	1.1835	1.1864

Table 2.1: Absolute values of the turn sequence for the triangular distribution

2.2.6 Comparison to Results on the LSP

In this section we compare the index policy to the cases addressed by [13], namely uniform, symmetric triangular and unit normal with the origin at zero. We also compare the index policy to the dynamic programming scheme presented in [3].

Under an optimal policy for the uniform case, the agent only turns at the endpoints and the initial direction doesn't matter. This is also the behavior of the index policy.

The symmetric triangular distribution is the probability distribution whose support is the $[-1, 1]$ interval and whose density increases linearly on $[-1, 0]$ and decreases linearly on $[0, 1]$. For this distribution, starting from zero, the authors of [13] proved that the optimal turn sequence (with absolute values x_i) are the solution to the following equations.

$$(x_i + x_{i+1})(1 - x_i) - \frac{1}{2}(1 - x_i)^2 - \frac{1}{2}(1 - x_{i-1})^2 = 0$$

with $x_0 = 0$.

The turns that solve these equations are shown in Table 2.1 along with the turns produced by the index policy. The index policy is about 0.25% sub-optimal for this case.

For the unit normal distribution, the authors of [13] numerically solve a set of partial differential equation, shown in Equation 2.1, to arrive at the turning points shown in Table 2.2. The index policy produces a very similar result, that is within 0.2%.

In Table 2.2 we also include the algorithm from [3], both as described, as well as with an obvious augmentation. As stated, the policy is to run the dynamic program-

	[13]	difference	Index Policy	difference	[3]	improved
x_1	1.4409		1.3392		1.4501	1.4501
x_2	2.6276	1.1867	2.4732	1.1340	2.6061	2.6061
x_3	3.6322	1.0046	3.4632	0.9900	2.6737	2.6737
x_4	4.5203	0.8881	4.3416	0.8784	2.6737	5.3474
x_5	5.3267	0.8064	5.1492	0.8076	5.3474	10.3948
x_6	6.0712	0.7445	5.8932	0.7440	10.3948	20.7896
x_7	6.7681	0.6969	6.5940	0.7008		
x_8	7.4253	0.6572	7.2516	0.6576		
mean	2.9030		2.9087		2.9916	2.9517

Table 2.2: Absolute values of the turn sequence for the unit normal distribution. Also shown are the differences between subsequent turns to aid comparison. For [3] we used $\epsilon = 0.1$. The improved version of [3] removes extra turns when transitioning from the dynamic programming policy to the doubling policy.

ming policy until the $[-R, R]$ interval has been searched, then return to zero and then pursue a doubling strategy. This can introduce one or two superfluous turns. For the "improved" version, shown in the table, we do as follows. Let $s \in \{-1, 1\}$ be the sign of the first endpoint of $[-R, R]$ that is reached at turn i_s. Immediately when this happens, we initiate the doubling strategy $x_i = -sR(-2)^{i-i_s}$.

Since we can efficiently compute the index policy for any distribution, it is now possible to optimize over the starting location. Figures 2-4 and 2-5 show the expected search time under the index policy as a function of starting location for the triangular and normal distributions respectively. In both cases, zero is not the optimal starting location. In fact for the triangular distribution it is the *worst* starting location; for the normal distribution zero is worse than any location less than 2.9 standard deviations from the mean. Table 2.2 compares the optimum found by the index policy to the optimal search reported in [89] for the unit normal from an arbitrary starting point.

We are also now in a position to consider the effect of a non-zero sensor radius for the first time. In particular, assume that the agent detects the particle if it is within a distance r_s. This simply changes the prior distribution as shown in Figure 2-6.We can still apply the index policy as discussed in Section 2.2.3. Figures 2-7 and 2-8 show how the expected search distance varies as a function of starting location and sensor radius. Section 2.3 will make extensive use of these results for the Gaussian

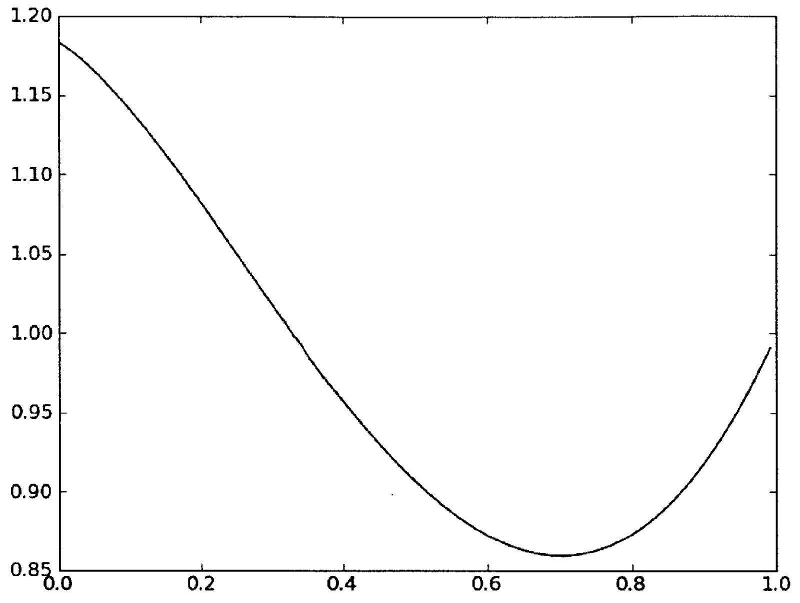


Figure 2-4: Expected search distance as a function of starting location for the symmetric triangular distribution

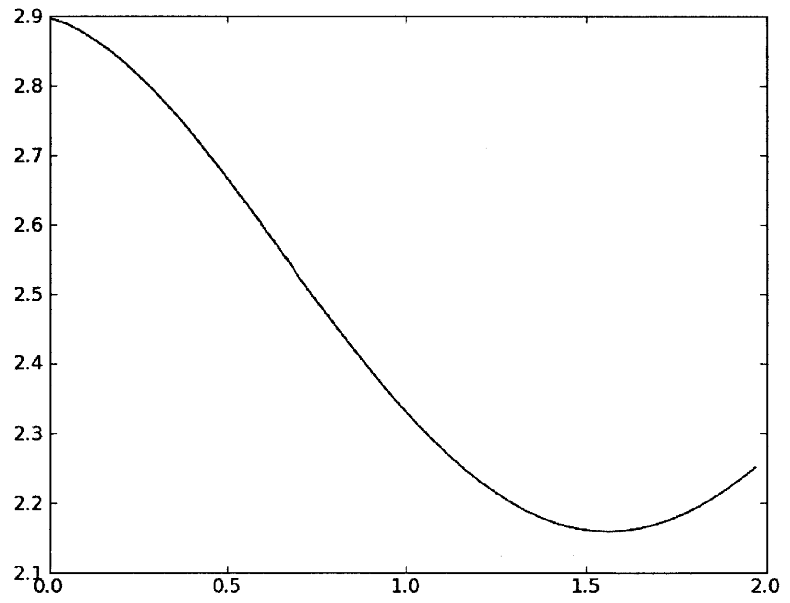


Figure 2-5: Expected search distance as a function of starting location for the unit normal distribution

	[89]	Index policy
x_0	1.5712	1.5728
x_1	2.7352	2.6760
x_2	3.7259	3.6384
x_3	4.6046	4.5032
x_4	5.4040	5.2944
x_5	6.1435	6.0296
x_6	6.8357	6.7304
mean	2.1630	2.1631

Table 2.3: Starting point and subsequent turns for searching the normal distribution.

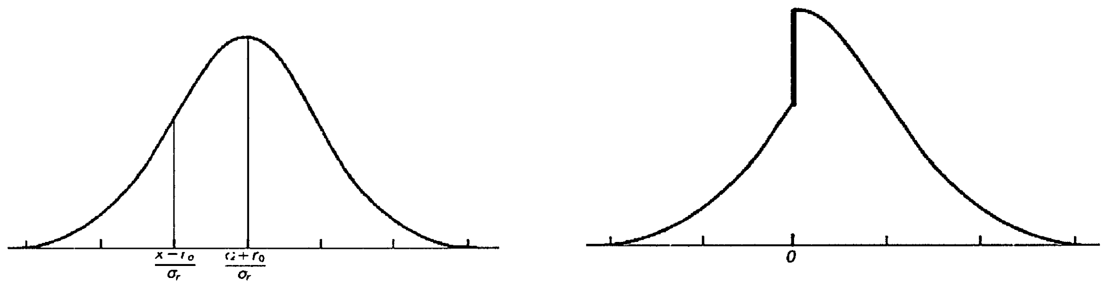


Figure 2-6: Modification of the spatial distribution to reflect non-zero sensor radius.

case in solving a novel path planning problem.

2.2.7 Comparison to the Dynamic Programming Approach

The approach from [3] provides an arbitrarily precise approximation to the LSP by discretization and dynamic programming. This approach requires a continuous measure G with density g, and finite first moment $\overline{g}^1$. Let $|G|^{-1}$ be the inverse survival function of the absolute value of G. To achieve an approximation of $(1+\epsilon)$, this entails discretizing the region into

$$n = \frac{64}{\epsilon} \frac{|G|^{-1}(1 - \epsilon\overline{g}^1/24)}{\overline{g}^1} \tag{2.19}$$

and applying an algorithm with complexity $O(n^2)$.

From Equation 2.19 one can see that as ϵ decreases, n increases at least as fast as its inverse. One should also note the relatively large constant. As a result $O(n^2)$ could actually be very burdensome at the level of accuracy we might want.

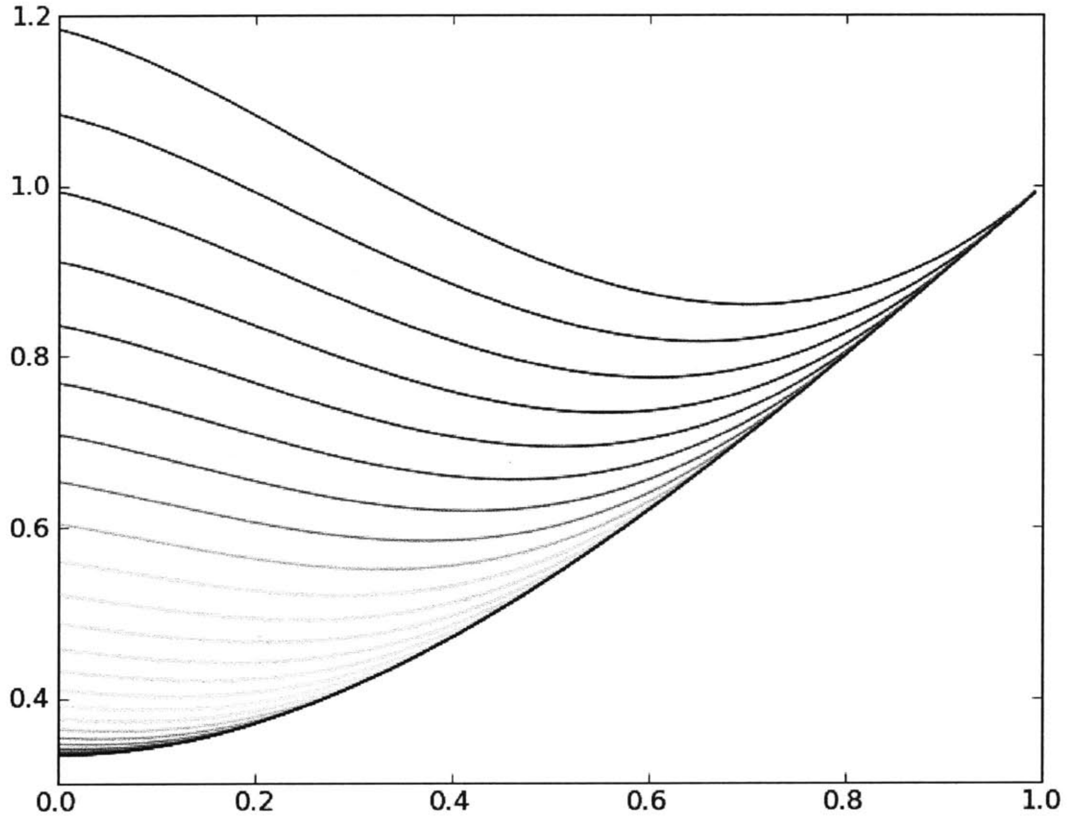


Figure 2-7: Expected search distance as a function of starting location and sensor radius for the symmetric triangular distribution. Blue indicates a sensor radius of zero, red indicates a sensor radius of 1, increments of 0.04.

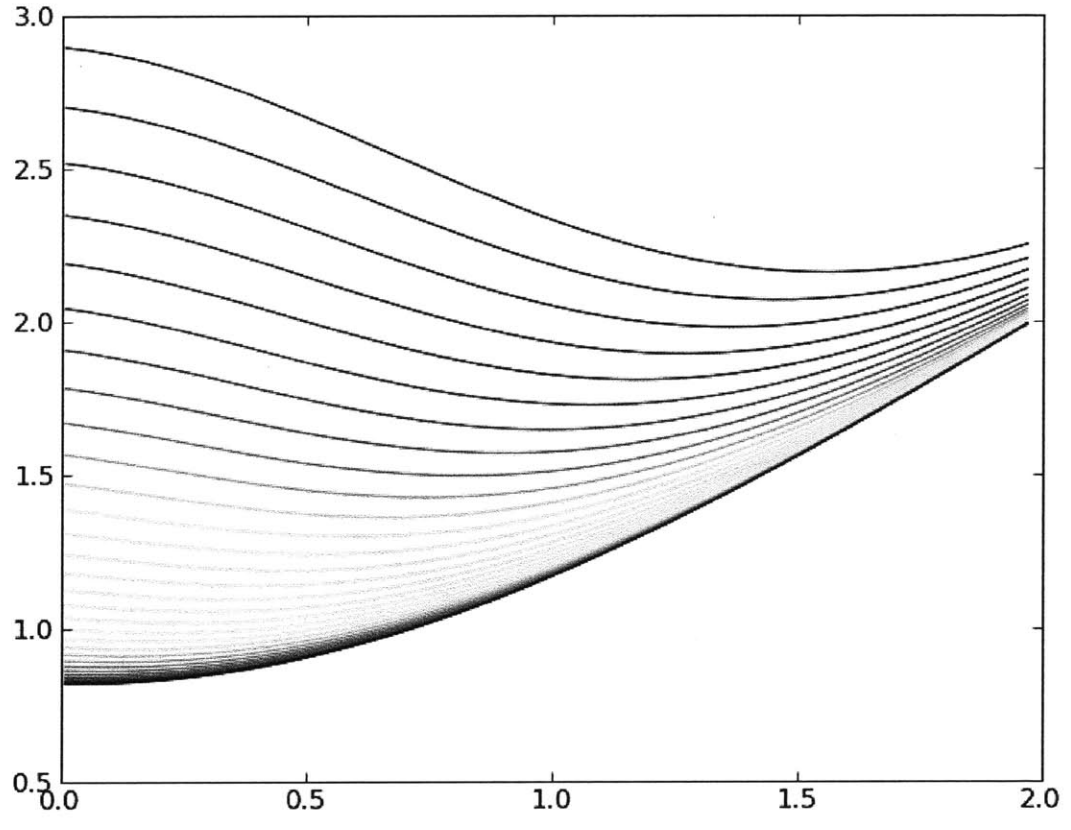


Figure 2-8: Expected search distance as a function of starting location and sensor radius for the unit normal distribution. Blue indicates a sensor radius of zero, red indicates a sensor radius of 2.4, increments of 0.08.

For example, for a guaranteed improvement on our result for the Gaussian case consider $\epsilon = .002$, which gives n of about 300000. At this level of discretization, this approach will require order of 10^{11} operations and, more importantly, use terabytes of memory. Even with $\epsilon = .1$, this approach requires about a half gigabyte assuming double precision variables.

Similarly, for the triangular case this approach will require the computation and storage of more than $110592\epsilon^{-2}$ values.

2.2.8 Infinitesimal Oscillation

The author of [38] provides a necessary and sufficient condition for the non-existence of an optimal rectifiable path for search on the real line. This condition is that the right and left upper-derivatives of the cumulative distribution function are infinite at zero. In particular they prove that for any policy that makes its first turn at $\epsilon > 0$ there is a better one that turns sooner. Put another way, the optimal policy begins with an "infinitesimal oscillation".

For the CPP this same condition is that there two or more paths whose upper-derivatives are infinite at zero.

For instance, the symmetric inverse distribution with density given by

$$\phi(x) = \frac{1}{2e}|x|^{-1}$$

meets this condition for oscillation.

Lemma 2.2.8. *The index policy can only produce an infinitesimal oscillation about the origin and nowhere else.*

Proof. Once away from the origin, the numerator in Equation 2.5 is bounded away from zero. For the subsidy to be finite, the denominator must also be bounded away from zero. Since any atom at z_i would have already been visited, this implies that $z_i' - z_i > 0$. □

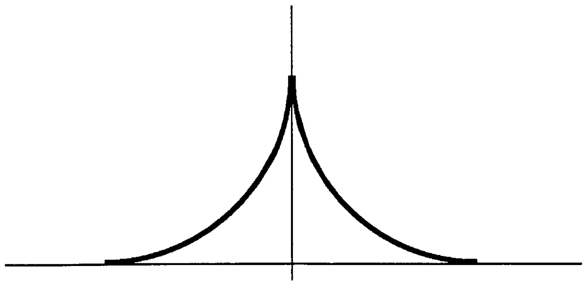


Figure 2-9: The index policy oscillates on this distribution, while the optimum does not.

Lemma 2.2.9. *The index policy produces an infinitesimal oscillation if the optimal policy begins with an infinitesimal oscillation.*

Proof. Assume that the optimal policy begins with an infinitesimal oscillation, which from [38] implies infinite derivatives at zero, which results. We ignore impulses at zero, assuming that they are immediately seen before we apply the policy.

If the agent moves a distance $\epsilon > 0$ in either direction it will see a subsidy of $\epsilon/\infty = 0$ in the other direction. However, the subsidy must have been greater than zero for all but a vanishing fraction of the distance ϵ covered, since the cumulative distribution function must be finite.

This means that to have gone any distance $\epsilon > 0$ would have required moving contrary to the index policy. $\square$

Remark 2.2.10. *The converse of Lemma 2.2.9 does not hold. This can be shown by examining the case with two paths each having density*

$$f_i(x) = r - \sqrt{2xr - x^2}$$

on the $[0, r]$ *interval for* $r = (2(1 - \pi/4))^{-\frac{1}{2}}$, *shown in Figure 2-9*

As a practical matter, this is not a major concern as it is generally safe to assume a minimum quantum of distance, for instance, from a small but non-zero sensor footprint or because of discrete time computation.

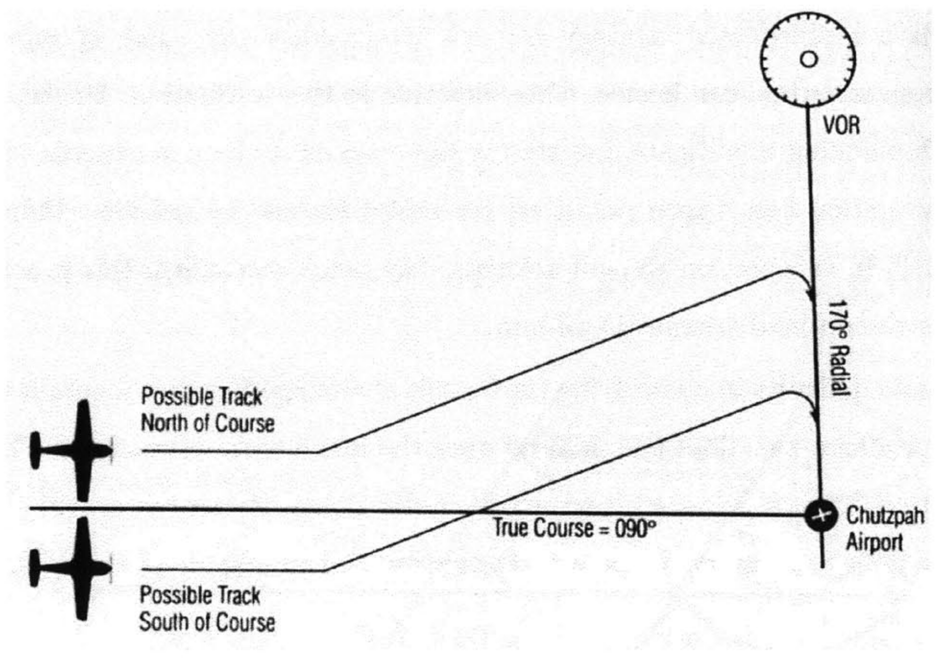

VOR
170° Radial
Possible Track North of Course
True Course = 090°
Chutzpah Airport
Possible Track South of Course

Figure 2-10: from *The Proficient Pilot* [83] [used with permission]

2.3 Aiming Off

This section investigates a traditional navigational technique, known to human experts of various fields as "off-course navigation," "landfall intercept," "single line-of-position," and "aiming off," which has been used by navigators on foot, ancient ships, pre-GPS aircraft, and modern submarines. Using this technique, the navigator deliberately aims to one side of their objective with the intention of following a line feature (*e.g.*, a road, coastline, celestial bearing, or radio beacon) that is known to intersect the objective. By doing so he can search more confidently and decrease the expected distance travelled. Figure 2-10 is an example from *The Proficient Pilot* [83]. The results in Section 2.2.6 confirm this practice.

In addition to navigational applications, *e.g.*, in a Global Positioning System (GPS)-denied environment, another practical application is in search for mobile targets. Suppose an Unmanned Aerial Vehicle (UAV) is being routed to take video of a suspicious mobile target on a road. Rather than move directly toward the mean of the target's position distribution it might be better for the UAV to aim off and acquire the target while following the road.

This is a sophisticated strategy because it considers the *value of information* derived from improved confidence in the direction to the destination. Unsurprisingly, then, path planning to mitigate uncertainty has been left to human experts—to quote a 1657 navigation text "[such practices] are better learned by practice, than taught by pen.[31]" In this section we will attempt, "by pen," to confirm this practice and determine the optimal amount of off-aim.

The main difficulty in quantifying the benefit of aiming off is that it entails the LSP as a sub-problem; how does one proceed once the line feature is reached? The result from Section 2.2 has provided a strong heuristic policy for search on the real line. Using this policy, we are now able pose the problem of aiming off as a straightforward optimization problem.

By way of contrast, the previously accepted solution to the CPP, discussed in Section 2.1.2, does not predict such a behavior: there is no preference as to starting place. On the other hand, the approach taken in [13] is, in principle, capable of answering our question, but in practice this may prove difficult because it involves solving many partial differential equations of the form of Equation 2.1. Another small drawback to this approach is it requires an absolutely continuous probability distribution.

The paper [89] improves on the method from [13] which allows them to find the optimal starting point for the LSP by solving a single PDE shown in System 2.2. This is the first and only scholarly confirmation of the practice of aiming off.

We improve on this result by characterizing the optimal performance *as a function* of starting point. This lets us quantify the trade-off between additional travel distance and additional search distance. The efficiency of the index policy will allow us to go even further, characterizing the optimal search performance as a function of sensor radius as well.

The dynamic programming approach presented in [25] can be applied but one must chose a discretization method. The discretization method provided in [3] provides a provable approximation but, as we discussed in Section 2.2.6, is computationally burdensome at the level of accuracy provided by the index policy.

A similar problem was posed in [81] which they called the Coastal Navigation Problem (CNP) in which a mobile robot attempted to robustly reach a goal location in the presence of features. The most general and exact form of their model encompasses our problem but is a continuous state POMDP which is intractable. The approach taken was to use an AMDP in which they tracked the mean and entropy of the agent location distribution. This approach is promising in that it predicts behaviors similar to aiming off, *e.g.*, wall following. However, to elicit these behaviors they had to apply an artificial reward model in which the robot was penalized for entropy when the distribution mean reached the goal. For the problem we will pose here, even with the modified reward structure, their approach would not aim off. Certainly such an approach could produce this behavior with an appropriate feature set, for instance a sufficient number of moments of the distribution. At least for the moment, however, there is no unified methodology for feature selection.

The goal of this section is to demonstrate a general methodology for determining an optimal control policy in a commonly arising navigation situation. We believe this is particularly interesting because it verifies a practice that human experts have known for centuries but is not produced by any automated planner.

2.3.1 Problem Specification and Assumptions

Assume that the objective is on a straight, featureless road. For the moment assume that the navigator can only detect the objective if he is at its location, Section 2.3.4 will discuss the effects of a non-zero sensor radius. Assume that the navigator starts at an initial position with no uncertainty and that the navigator–objective vector is perpendicular to the road and has length ℓ. We will assume that the navigator is a single integrator

$$\dot{x} = u + w,$$

with control $||u|| \leq v$ and Gaussian noise $w \sim \mathcal{N}(0, I\sigma^2)$. This is a natural model for a navigator with a compass, for example.

We remark that the approach we will take is tractable for any noise model. We choose Gaussian noise because it simplifies the analytical discussion and interpretation of our example. In particular, the location of the navigator evolves according to a Wiener process with drift u. That is, at time t, the position is a Gaussian random variable with mean $\overline{x}(t) = \int_0^t u(\tau)d\tau$ and variance $t\sigma^2 I$. In the UAV search example, this would be equivalent to the target undergoing Brownian motion.

2.3.2 Index Policy for Gaussian Search

We now describe the application of the index policy to the case of a single agent searching a one-dimensional Gaussian distribution. Note that Equation 2.4 scales linearly under a scaling of distance. Since our error model ensures that the uncertainty will always be normally distributed we need only determine the search behavior for all starting locations in the standard normal. The actual search behavior can be recovered with appropriate transformation of units.

At the moment that the agent arrives at the road, define zero to be the mean of the distance-to-objective distribution and define the unit of distance to be its standard deviation. With respect to this zero, let $\hat{x}$ denote the mean of the agent location distribution and z^+, z^- as the extent to which the agent has searched in the positive and negative directions, respectively. Let ϕ and Φ denote the density and cumulative of the standard normal, respectively.

We can simplify the index function by redefining the origin (of the CPP) to be the agent's location, which makes $a = 0$ for both directions. Rewriting Equation 2.4 and correcting signs in the negative direction,

$$\mathbb{E}\left[c^+\right] = \int_{z^+}^{z'} (\chi - \hat{x})\phi(\chi)d\chi + (1 - \Phi(z^+, z'))2(z' - \hat{x})$$
$$\mathbb{E}\left[c^-\right] = \int_{z^-}^{z'} (\chi - \hat{x})\phi(\chi)d\chi + (1 - \Phi(z', z^-))2(z' - \hat{x})$$

Making use of

$$\int_a^b \chi\phi(\chi)d\chi = \phi(a) - \phi(b)$$

we arrive at Policy 2.3.1.

Policy 2.3.1 (Index Policy for Gaussian Search). *The agent should move in the positive direction whenever*

$$\begin{aligned}\inf_{z'>z^+} \hat{x} - 2z' + \frac{\phi(z^+) - \phi(z') + 2(z' - \hat{x})}{\Phi(z^+, z')} &\leq \\ \inf_{z'<z^-} 2z' - \hat{x} + \frac{\phi(z^-) - \phi(z') + 2(\hat{x} - z')}{\Phi(z', z^-)} &\end{aligned}$$

Using Policy 2.3.1 as a search strategy, the expected search time for $\mathcal{N}(0,1)$ as a function of starting location is shown in Figure 2-5. We will denote this function $\hat{d}_s(\hat{x})$ with domain and range in units of standard deviations. The expected search distance of a normal distribution with standard deviation σ is given by $\sigma \hat{d}_s(x/\sigma)$.

2.3.3 Optimization

We assume that since the navigator is trying to minimize travel time he always applies maximum control $||u|| = v$ and attempts to travel in a straight line. Under this type of control, and given a search strategy, the problem is simply to select an initial heading.

Given $\hat{d}_s$, we can now write the expected search time as a function of heading θ and the initial distance to the road ℓ.

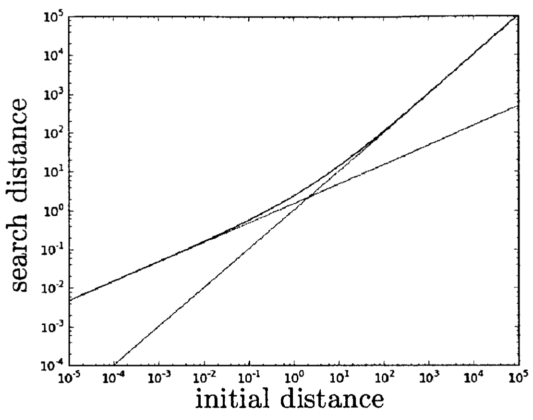
$$d_T(\ell, \theta) \equiv \frac{1}{v}\left(\ell \sec(\theta) + \sigma_r \hat{d}_s\left(\frac{\ell \tan\theta}{\sigma_r}\right)\right) \tag{2.20}$$

The two terms are, respectively, the distance to the road and the search distance subject to a unit transformation. When the agent reaches the road the standard deviation of error is given by $\sigma_r \equiv \sqrt{\frac{\ell \sec(\theta)\sigma^2}{v}}$.

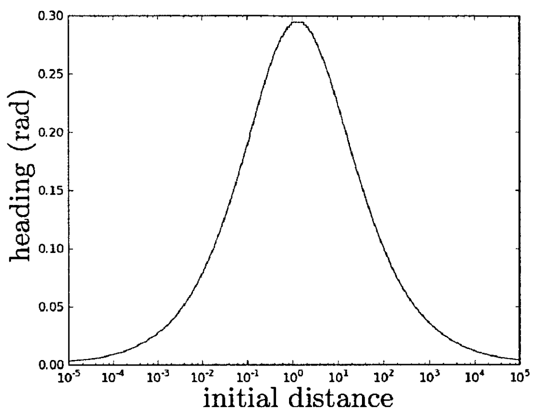
All that remains is to optimize Equation 2.20 over θ. The functions

$$\begin{aligned} d_T^*(\ell) &\equiv \min_\theta d_T(\ell, \theta) \\ \theta^*(\ell) &\equiv \arg\min_\theta d_T(\ell, \theta) \end{aligned}$$

are shown in Figures 2-11a and 2-11b.



(a) Minimum search time



(b) Optimal heading

Figure 2-11: Asymptotic behavior with zero sensor radius as a function of initial distance from the road.

Discussion

The interesting shape of these curves results from the fact that the difficulty of searching grows with the *square root* of the initial distance to the road, ℓ. At large values of ℓ, the amount of offset that corresponds to the minimum in Figure 2-8 grows at a rate proportional to $\sqrt{\ell}$ and so the optimal heading goes to zero. For the same reason, the search time becomes a vanishing component of the travel time.

When the navigator starts very close to the road, search time dominates and is proportional to $\sqrt{\ell}$. In this limit it is best to immediately move to the road. This can be seen by linearizing d_T about some particular θ_0 and examining the limit of $\ell \to 0$. In this limit we can approximate $\hat{d}_s(x) \approx a - bx$ for $a, b \geq 0$. Plugging this linearization into Equation 2.20,

$$
\begin{aligned}
d_T(\ell, \theta_0 + d\theta) &\approx \frac{1}{v}\Bigg[\ell \sec(\theta_0 + d\theta) + \sqrt{\ell \sec(\theta_0 + d\theta)\sigma^2/v} \quad \times \\
&\qquad \left(a - b\frac{\ell \tan(\theta_0 + d\theta)}{\sqrt{\ell \sec(\theta_0 + d\theta)\sigma^2/v}}\right)\Bigg] \\
&= \frac{1}{v}\Big[\ell \sec(\theta_0 + d\theta) + \sqrt{\ell \sec(\theta_0 + d\theta)\sigma^2/v} \quad \times \\
&\qquad \left(a - b\sin(\theta_0 + d\theta)\sqrt{\ell \sec(\theta_0 + d\theta)v/\sigma^2}\right)\Big] \\
&= (1/v)\left[\ell \sec(\theta_0 + d\theta)(1 - b\sin(\theta_0 + d\theta)) + a\sqrt{\ell \sec(\theta_0 + d\theta)\sigma^2/v}\right] \\
&\approx (\ell/v)\left(\sec(\theta_0) + d\theta \sec(\theta_0)\tan(\theta_0)\right)\left(1 - b(\sin(\theta_0) + d\theta\cos(\theta_0))\right) + \\
&\qquad \left(\sqrt{\ell/v}\right) a\sigma\sqrt{\sec(\theta_0)}\left(1 + d\theta\tan(\theta_0)/2\right)
\end{aligned}
$$

Small changes of θ decrease d_T proportionally to ℓ but increase it proportionally to $\sqrt{\ell}$. Therefore in the limit of $\ell \to 0$, $\theta^* \to 0$.

In between these two regimes we see that it is sometimes optimal to choose large heading offsets—over 15 degrees. Referring to Figure 2-5, this should not be surprising. There is a substantial benefit as we move the start point away from the mean. On the other hand, the extra travel distance only increases as $\sec(\theta)$ and the additional search time as $\sqrt{\sec(\theta)}$.

2.3.4 Effects of a Finite Sensor Radius

Suppose that the navigator can see a distance $r_s = \sigma_r \hat{r_s}$ in either direction along the road. We apply the index policy to the distribution formed by removing the center $2r_s/\sigma_r$ about the agent's starting location $\hat{x} = x/\sigma_r$ from the normal distribution. This transformation is shown in Figure 2-6.

Let $\hat{d_s}'(\hat{x}, \hat{r_s})$ denote the expected search time of such a policy. To determine the actual search time we make the following correction. Let χ be the random variable denoting the distance to goal.

$$\hat{d}_s(\hat{x},\hat{r_s}) \;=\; (\mathbb{P}\left[|\chi| \leq \hat{r_s}\right])\mathbb{E}\left[|\chi| \;|\chi \leq \hat{r_s}\right] \;+\; (\mathbb{P}\left[|\chi| > \hat{r_s}\right])(d_s'(\hat{x},\hat{r_s}) + \hat{r_s})$$

The first term is the case in which the agent can immediately see the goal. Otherwise we simply add one sensor radius to the search time. Figure 2-8 shows $\hat{d}_s(\hat{x},\hat{r_s})$ for sample values of $\hat{r_s}$ between 0 and 1.75. Above this value it no longer helps to aim off.

We adjust Equation 2.20 to now include the sensor radius

$$d_T(\ell,\theta) \equiv \ell\sec(\theta) + \sigma_r\hat{d}_s\left(\frac{\ell\tan\theta}{\sigma_r},\frac{r_s}{\sigma_r}\right) \tag{2.21}$$

and optimize over θ.

Figure 2-12 shows the heading that minimizes Equation 2.21 for a variety of speeds and sensor radii. These parameters were chosen to bracket those of a person navigating on foot and the variances are scaled such that the asymptotic behaviors match.

As a function of starting distance ℓ, expected travel distance is qualitatively unchanged from Figure 2-11a and we do not show it. Primarily this is because the navigator must still move to the road and then to the actual location of the objective. In the $\ell \to 0$ limit, the performance improves by the ratio of $\mathbb{E}\left[|\mathcal{N}(0,1)|\right] \;:\; d_s(0)$ which is an improvement of about 45%. This improvement decreases along with the probability of immediately seeing the objective.

Discussion

Unsurprisingly, for large distances, Figure 2-12 matches Figure 2-11b: the sensor radius is becoming increasingly negligible. The interesting behavior is the very steep rise in heading off-aim from zero up to the asymptote. Generally we see a rise from no offset to the asymptote in a little more than a doubling of distance. It seems that once it is no longer "very likely" that the navigator will be able to see the objective immediately, off-aim becomes attractive very quickly.

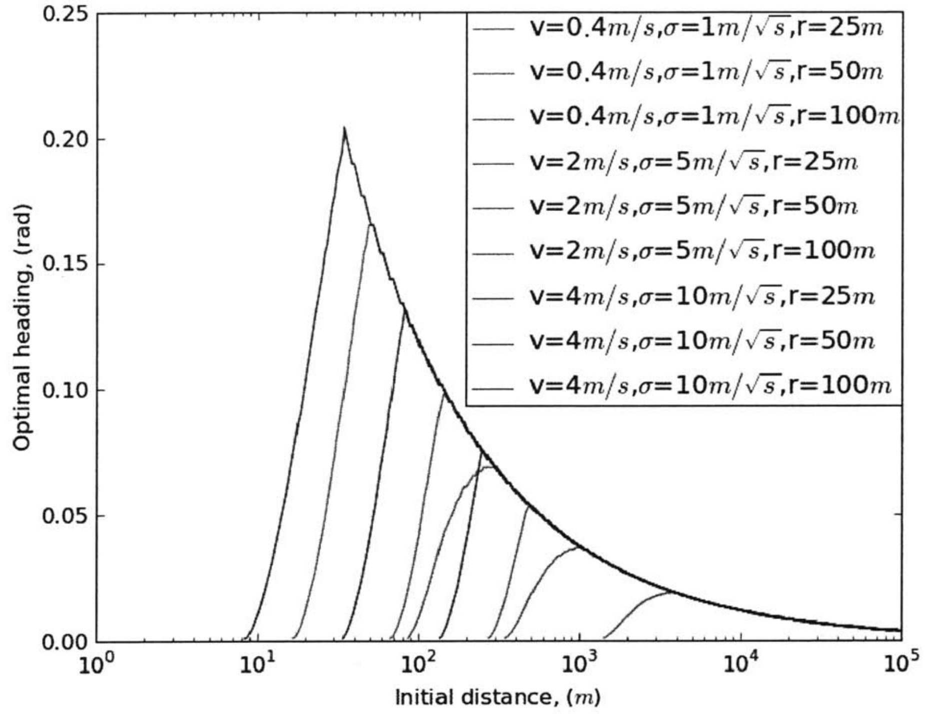


Figure 2-12: Optimal heading as a function of starting distance from the road for various speeds and sensor radii. Within each color, the three curves show increasing sensor radii from left to right. These parameters were chosen to bracket those of a person navigating on foot and the variances are scaled such that the asymptotic behaviors match.

We see a maximum heading offset of 0.2 radians for the fast navigator with the most limited visibility. For these conditions it is advisable to aim off even over very short distances (a few tens of meters). On the other extreme, the slower, more careful navigator with the best visibility has a maximum heading offset of 0.02 radians. In this case the navigator prefers the direct route for distances up to nearly two kilometers.

Orienteering books recommend between 2–3 and 10 degrees of off-aim (*e.g.*, [73, 87]). The surprising degree of non-linearity in this regime explains the lack of any standard 'rules of thumb' for aiming off in orienteering.

2.3.5 Conclusion and Extensions

We were able to efficiently tackle this problem because we were able to pre-compute a function that takes a measure over the road as an input and returns a mean search time. Given that, the path planning problem was just a matter of optimizing over trajectories. The approach here, then, can be straightforwardly applied to more difficult problems, for instance more complex noise or dynamics. The main difficulty in those cases would be determining the set of trajectories over which we should optimize. In any problem in which all "reasonable" trajectories can be efficiently enumerated, this approach will remain tractable. This enumeration will be a central topic in Chapter 4 when we consider search in higher-dimensional spaces.

What made this approach tractable was the fact that we evaluated a black-box policy to determine the value function for the tiny subset of states reachable from the initial conditions. In the language of dynamic programming, what we are doing is using the value function from the CPP to determine the cost-to-go for a searchable and pre-determined set of trajectories. On the surface this is trivial—given an optimal policy, it is tractable simply to use it. More deeply, though, it represents an exploitation of hierarchy: We used the policy from a relatively simple, one-dimensional problem to tackle a seemingly difficult two-dimensional problem. We will revisit this intuition in the concluding remarks of this thesis, Section 6.2.

Chapter 3

Search in Geometric Networks

This chapter extends our index policy to search problems in which the region is a network. Our index policy still has a natural interpretation in such problems. As a result we are able to generalize the policy, by analogy, to a number of problems that are not Whittle indexable.

Section 3.1 provides the essential terminology and formalisms. Section 3.2 presents the generalized index policy, discusses the complexity of the policy as a function of the size of the network, and presents alternative polices of similar character with lower complexity. Section 3.3 considers the observable Minimum Latency Tour Problem (MLTP) as a special case and evaluates the index policy as a heuristic for combinatorial optimization. Section 3.4 describes a methodology for search of one-dimensional subspaces of metric spaces of higher dimension. This chapter will focus on single-agent search with the exception of Section 3.5 which considers case in which multiple agents are simultaneously searching.

3.1 Preliminaries

Definition 3.1.1. *Metric space*

A metric space *consists of the pair of a set S and a distance function $d(\cdot,\cdot)$:*

$S \times S \to \mathbb{R}_+$ *satisfying for all* $s_0, s_1, s_2 \in S$

$$\begin{aligned} d(s_0, s_1) = 0 \quad &\iff \quad s_0 = s_1 \quad \textit{and,} \\ d(s_0, s_1) \quad &\leq \quad d(s_0, s_2) + d(s_2, s_1) \end{aligned}$$

We will never use $d(\cdot,\cdot)$ *to denote the single-element vector norm.*

It is common to refer to the set S as a metric space and leave its distance function implicit. We will disambiguate distance functions as necessary with subscripts $d(\cdot,\cdot)_s$.

Definition 3.1.2. *Lipschitz Continuous function*

A Lipschitz continuous function $f : X \to Y$ *between metric spaces* X *and* Y *is one for which there exists a constant* K*, called the Lipschitz constant, such that for any* $x_1, x_2 \in X$

$$d(f(x_1), f(x_2))_Y \leq K d(x_1, x_2)_X$$

Definition 3.1.3. *Uniform Path*

We define a uniform path or simply path *to be a Lipschitz continuous function* $\mathcal{P} : [0, \ell) \to S$ *from a possibly infinite interval to metric space* $S, d(\cdot,\cdot)$ *for which:*

$$\lim_{\delta \to 0} \frac{d(\mathcal{P}(x), \mathcal{P}(x+\delta))}{\delta} = 1 \quad \forall x \in [0, \ell)$$

We will refer to ℓ *as the length of the path.*

If this length is finite the endpoint *is well-defined as* $\mathcal{P}(\ell) \equiv \lim_{x \to \ell} \mathcal{P}(\ell)$. *If* $\mathcal{P}(\ell) = \mathcal{P}(0)$ *the path is called* closed.

In a slight abuse of notation we use $\mathcal{P}(S)$ *with a set argument* $S \subset [0, \ell)$ *to refer to the set* $\bigcup_{x \in S} \mathcal{P}(x)$ *and, when it is unambiguous,* $\mathcal{P}$ *to refer to the set* $\mathcal{P}([0, \ell))$*, for instance when referring to the Lebesgue measure* $\mathcal{L}(\mathcal{P}) = \ell$.

Definition 3.1.4 (Length space). *Given a metric space* $X, d(\cdot,\cdot)$ *we can define a new metric* $d(x_1, x_2)_I$ *called the* intrinsic metric *as the infimum of the length of every uniform path between* x_1 *and* x_2 *(or infinity if no such path exists). If* $d(\cdot,\cdot)$ *is* *everywhere in agreement with $d(\cdot,\cdot)_I$, then X is called a length space, also known as a path metric space.*

For x_1, x_2 in a length space, if $d(x_1, x_2) < \infty$, there exists at least one path $\mathcal{P}$ whose length, $\mathcal{L}(\mathcal{P})$ is arbitrarily close to $d(x_1, x_2)$. If $\mathcal{L}(\mathcal{P}) = d(x_1, x_2)$, $\mathcal{P}$ is a called a *shortest path.*

Definition 3.1.5 (Cover). *A function $f : X \to Y$* covers *set $S \subseteq Y$ if $S \subseteq \bigcup_{x \in X} f(x)$. A single-valued function is called a* cover *if it is onto, while a set-valued function $f : X \to Y^*$ is called a cover if f covers the set Y.*

3.1.1 Geometric Networks

A geometric network is a length space consisting of a continuous, one-dimensional network structure that formalizes the notion of a roadmap. A geometric network is a network in which the edges are "roads," *i.e.*, continuous segments. On a roadmap, there is a well-defined notion of the distance between any two locations or "addresses." This is in contrast to the typical definition of a network, in which it is ordinarily only meaningful to talk about this distance between *vertices.*

We begin with a representation of the environment as a connected, undirected network $\langle \mathcal{V}, \mathcal{E}, \mathcal{L} \rangle$, with vertices $\mathcal{V}$ and edges $\mathcal{E}$, with edge lengths given by $\mathcal{L} : \mathcal{E} \to \mathbb{R}_+$. From this, let us define the region, $\mathcal{A}$, as the set of pairs consisting of an edge label and a continuous location along that edge, like a street name and an address. Formally, $\mathcal{A} \equiv \mathcal{V} \cup (\mathcal{E} \times (0,1))$. For simplicity however, we will refer to all points in the region, including the vertices, as pairs in $\mathcal{E} \times [0,1]$ with the understanding that $(e_{ij}, 1) \equiv (e_{jk}, 0) \equiv v_j$.

When it is unambiguous, we will use $e \in \mathcal{E}$ to denote the set $\bigcup_{t \in [0,1]} (e, t)$. Given this, it is useful and intuitive to interpret $\mathcal{L}$ as a measure on subsets of $\mathcal{A}$ rather than simply a function on $\mathcal{E}$. When $\mathcal{L}$ is applied to a set S, we interpret it as

$$\sum_{e \in \mathcal{E}} \mathcal{L}(e) \lambda \left(\{ t \in [0,1] \text{ s.t. } (e,t) \in S \} \right)$$

wherein λ denotes the Lebesgue measure on $[0,1]$. When discussing integration and measureability in $\mathcal{A}$, it is with respect to this definition.

The region has a natural and intuitive distance metric: The distance between any two locations in $\mathcal{A}$ is the length of the shortest path between them. Let d^*_{ij} denote the length of the ordinary shortest path between vertices i and k according to $\mathcal{L}$ (this length could be found by using Dijkstra's algorithm, for instance). The distance between (e_{ij}, x) and (e_{kl}, y) for $e_{ij}, e_{kl} \in \mathcal{E}$, $x, y \in [0,1]$ is given by

$$\begin{aligned}
d(v_i, v_j) &= d^*_{ij} \\
d(v_i, (e_{jk}, x)) &= \min\Big(\;\; x\mathcal{L}(e_{jk}) + d^*_{ij}, \quad (1-x)\mathcal{L}(e_{jk}) + d^*_{ik}\Big) \\
d((e_{ij}, x), (e_{kl}, y)) &= \min\Big(\;\; \mathcal{L}(e_{ij})|x-y|, \; \mathcal{L}(e_{ij})(1-|x-y|) + d^*_{ij}\Big) \text{ if } e_{ij} = e_{kl} \\
&\quad \text{otherwise,} \\
&= \min\Big(\;\; d(v_i, (e_{ij}, x)) + d((v_i, (e_{kl}, y)), \\
&\qquad\quad\;\; d(v_j, (e_{kl}, x)) + d((v_j, (e_{kl}, y))\Big)
\end{aligned}$$

3.1.2 Minimum Latency Covers

As before, we are interested in a policy that minimizes the expected time of finding a goal. For the single-agent case in a bounded network, we will refer to this problem as the Minimum Weighted Latency Covering Problem (MWLCP).

Problem 3.1.1 (MWLCP). *Let $\mathcal{P}$ be a uniform path that covers the region. Define the following indicator of whether a location "a" has not been visited by $\mathcal{P}$ before time t.*

$$I^0_{\mathcal{P}}(a,t) \equiv \begin{cases} 0 & \text{if } a \in \bigcup_{\tau<t} \mathcal{P}(\tau) \\ 1 & \text{otherwise} \end{cases} \tag{3.1}$$

Given a measure μ over the region $\mathcal{A}$, we can compute the weighted latency of the path, $w(\mathcal{P})$.

$$w(\mathcal{P}) = \int_0^{\ell} t I^1_{\mathcal{P}}(\mathcal{P}(t), t) d\mu(\mathcal{P}(t)) \tag{3.2}$$

Finding the covering path minimizing Equation 3.2 will be referred to as the Min-

imum Weighted Latency Covering Problem (MWLCP).

Problem 3.1.2 (GSP,MLTP). *If the region is a network and μ is supported by only the nodes, then the Minimum Weighted Latency Covering Problem (MWLCP) is called the Graph Search Problem (GSP). If μ is uniform on the nodes, then the GSP is called the Minimum Latency Tour Problem (MLTP)*

3.2 Generalizing the Index Policy

Notice that we already have a policy for the MWLCP on star-shaped networks. That policy is to pursue the path minimizing Equation 2.5. Recall that the interpretation is that the agent is to pursue the path with the lowest break-even *goal subsidy* if the agent pays unity cost for motion and must return to its starting point if it doesn't find the goal. Stated like this, the generalization is natural.

Let $\mathcal{P}_a$ be a closed path (not necessarily covering) of length ℓ with $\mathcal{P}_a(0) = \mathcal{P}_a(\ell) = a$. The break-even price of $\mathcal{P}_a$ is given by

$$\gamma^*(\mathcal{P}_a) = \frac{w(\mathcal{P}_a) + (1 - \tilde{\mu}(\mathcal{P}_a))\ell}{\tilde{\mu}(\mathcal{P}_a)}. \tag{3.3}$$

Policy 3.2.1. *The index policy is to move in the direction of the path (or path in the limiting sequence) that minimizes Equation 3.3.*

We will refer to the problem of finding such a path as the Minimum Price Path Problem (MPPP).

Conjecture 3.2.1. *For measures corresponding to the hiding policy at a Nash Equilibrium of the search game described in Section 2.1.4 on weakly-Eulerian networks, the index policy is optimal.*

Argument. From [40], maximum search rate is always achievable and any minimum length cover achieves it. For weakly-Eulerian graphs such covers can be found using local methods. The index policy which maximizes its search rate point-wise will always find a path that achieves the maximum search rate. □

3.2.1 Complexity, Relaxation, and Approximation

As posed thus far, the index policy entails comparing every closed path from the agent's location, of which there are infinitely many. In particular, minimizing Equation 3.3 over all closed paths is no easier than solving the MWLCP directly if we are to include consideration of all of the covering paths: For a covering path, the index function is exactly its expected latency.

As in the CPP we can restrict the set of paths being considered. We start by restricting ourselves to paths that are minimum *length* covers of their range. Under this restriction, we can optimize over the turn-around points of which there are now only finitely many. This reduces the computation to a finite number of paths. However, in general graphs there will remain a combinatorial number (in the size of the graph) of distinct paths to compare as we will shortly prove.

The MWLCP is clearly NP-hard. With uniform support across only the vertices, it is equivalent to the MLTP which is NP-hard and has a reputation for being much harder than the Traveling Salesman Problem (TSP). Even restricted to tree topology, the MLTP remains NP-hard [84]. We will prove the NP-hardness of the Minimum Price Path Problem (MPPP) with uniform measure on the vertices and tree topology by reduction to the MLTP.

Theorem 3.2.2. *The minimum price path problem is NP-hard.*

Proof. Let $\langle \mathcal{V}, \mathcal{E}, \mathcal{L}, x_0 \rangle$ be an instance of the MLTP and assume we have a solver for the MPPP. Let ℓ^+ denote the length of the longest edge in $\mathcal{E}$. Construct a network $\mathcal{V}' = \mathcal{V} \cup \{a_0\}$, $\mathcal{E}' = \mathcal{E} \cup \{(a_0, x_0)\}$ and $\mathcal{L}((a_0, x_0)) = d = 2|\mathcal{V}|\ell^+$. Let the measure be uniform over $\mathcal{V}$. Solve the MPPP on this new network and let $\mathcal{P}$ be that solution.

From Lemma 3.2.3, we have $\mathcal{P}$ being a minimum latency cover of $\mathcal{V}'$. Since this cover starts with (a_0, x_0) its latency is the constant d more than that of the latency of the sub-path covering $\mathcal{V}$. Since this d is common to all paths, this sub-path is a minimum latency covering of $\mathcal{V}$. □

Lemma 3.2.3 (helper lemma). *The path $\mathcal{P}$ covers $\mathcal{V}$.*

Proof. Assume the contrary, *viz.*, there exists some vertex $v \in \mathcal{V}$ not in the path that could be added to it by increasing its length by no more than $2\ell^*$. This increases the the first term in the numerator of Equation 3.3 by no more than $2\ell^*$ while decreasing the second term by at least $d/|\mathcal{V}| = 2\ell^*$ and increasing the denominator. Therefore adding v to the path decreases its price, contradicting optimality □

For small enough graphs, it may remain tractable to enumerate all "qualitatively distinct" paths. Two path are *qualitatively distinct* if they cannot be transformed into the same path by inserting or removing disjoint sub-paths of the form $[(e, x) \to (e, y) \to (e, x)]$ for $x, y \in (0, 1)$. For each qualitatively distinct path, the turn-around points y can be solved for efficiently.

If the graph is so large that we must insist on an index policy for search with polynomial complexity (in the size of the network), we must make further compromises. The remainder of this section will propose such compromises.

If we replaced the latency term in Equation 3.3 with the length of the tour, we are attempting to solve now a so-called *minimum ratio* optimization problem. Given a measure μ over region $\mathcal{A}$, let us define the *price* of subset $S \subseteq \mathcal{A}$ as

$$\gamma(S) \equiv \frac{\mathcal{L}_C(S)}{\mu(S)} \tag{3.4}$$

wherein $\mathcal{L}_c$ is the length of a minimum cover of S. In this section we will discuss the problems of finding paths and trees of minimum price.

Problem 3.2.1 (Minimum Ratio Path Problem (MRPP)). *The MRPP is the problem of finding the closed path, $\mathcal{P}_a$ (or a limiting sequence of paths), starting at a, with lowest "price" as defined by Equation 3.4.*

If we require that the path contain only complete edges we will call the problem the Minimum Ratio Circuit Problem (MRCP).

The MRCP is the rooted version of the Tramp Steamer Problem (see, *e.g.*, [57]). Unlike the Tramp Steamer Problem, the MRCP is NP-hard. A simple way to see this is to use the same long-edge construction as in the proof of Theorem 3.2.2 to reduce the MRCP to the TSP.

It was pointed out in [68] that any ratio optimization problem with objective $z(x) = f(x)/g(x)$ for $g(x) > 0$ is no harder than optimizing a $z'(x) = f(x) + \gamma g(x)$ because search for the critical $\gamma^* = \min_x z(x)$ is not difficult. As a result, one can leverage existing algorithms for circuit minimization to solve the MRCP.

Lemma 3.2.4. *Given a ratio γ and a geometric network $\mathcal{A} = \langle \mathcal{V}, \mathcal{E}, \mathcal{L} \rangle$ we can construct an isometric region $\mathcal{A}'$ with $|\mathcal{V}'| \leq 3|\mathcal{V}|$ in which the closed path minimizing $z' = \mathcal{L}(\mathcal{P}) - \gamma\mu(\mathcal{P})$ is a circuit (i.e., uses only complete edges). The* NodeAdding *Algorithm describes this construction.*

Proof. Assume by way of contradiction that all optimal paths turn at some location (e, t) for some $t \in (0, 1)$ and consider any arbitrary optimum. Assume without loss of generality that the path goes from $(e, 0)$ to (e, t) and back. Let $(e, [0, t])$ denote the set $\bigcup_{\tau \in [0,t]} (e, \tau)$. If

$$\gamma\mu\left((e, [0, t])\right) \leq 2t\mathcal{L}(e),$$

we could construct a path with no greater cost by removing this sub-path. Therefore we may assume the contrary.

By the construction in the NodeAdding Algorithm, the vertex at $(e, 1)$ maximizes $\gamma\mu((e, [0, t])) - 2t\mathcal{L}(e)$. Therefore we have

$$\gamma\mu(e) - 2\mathcal{L}(e) \geq \gamma\mu((e, [0, t])) - 2t\mathcal{L}(e).$$

This implies that we can construct a path with no greater cost by replacing this sub-path with one that goes from $(e, 0)$ to $(e, 1)$ and back. □

Corollary 3.2.5. *The MRPP is no harder than the MRCP.*

By use of Algorithm NodeAdding and Lemma 3.2.4 we have reduced the MRPP to the relatively well-studied problem of finding the shortest circuit that includes a given node. At this point, one can use an algorithm of one's choice. In particular, algorithms exist that can provide provable approximation to this problem.

However, since the MPPP is *already itself* a simplification of the problem of minimizing Equation 3.3, the value of a provable approximation is not self-evident, particularly given the size of the bounds available. We now propose an alternative approach that is more specific to the problem at hand.

We will refer to the MRPP on trees as the Densest Sub-Tree Problem (DSP). Although computing minimum latency tours is difficult on trees, the DSP can be solved in polynomial time as we prove in Theorem A.0.1. Furthermore, we can do so without searching on price γ as shown in Algorithm DensestSubtree. In Lemma A.0.2 we prove that the subtree returned by DensestSubtree has only one edge leaving the root. As a result it can be interpreted unambiguously as a policy.

Let us generalize the notion of a spanning tree to geometric networks.

Definition 3.2.6 (Geometric Spanning Tree). *Given a geometric network* $\mathcal{A} = \langle \mathcal{E}, \mathcal{V}, \mathcal{L} \rangle$*, and a root location* a_0*, the Geometric Spanning Tree is a new, tree-shaped geometric network* $\mathcal{A}'$ *with the same total length* $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$*, and an association* $A : \mathcal{A} \to \mathcal{A}'$ *such that* $d(A(a), A(a_0)) = d(a, a_0)$.

A Geometric Spanning Tree can easily be constructed by breaking each loop at the point most distant from the root. This is depicted in Figure 3-1 and described in Algorithm NetToTree. Taken together, Algorithms NetToTree and Densest-Subtree provide a polynomial approach to finding dense sub-regions connected to the agent's location. We refer to this approach as DenseSubnet.

3.2.2 An Augmentation

Figure 3-2 depicts a relatively simple failing of DenseSubnet. In that example, the region is a circular sector with multiple spokes and the measure is uniform on the arc. If the agent is in the center, the algorithm will prefer the interior spokes to

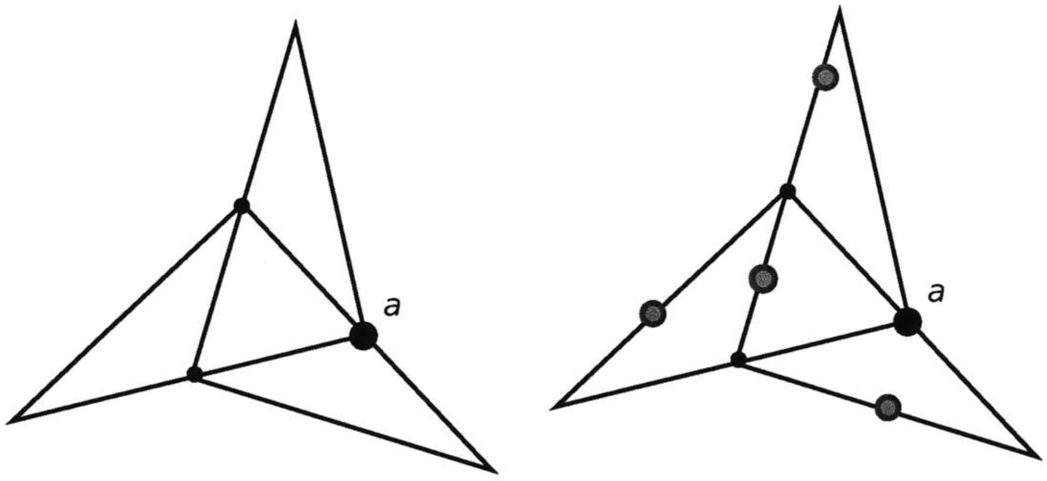


Figure 3-1: Visual description of the Geometric Spanning Tree. Two non-connecting nodes are added to each loop at the point most distant from the root a.

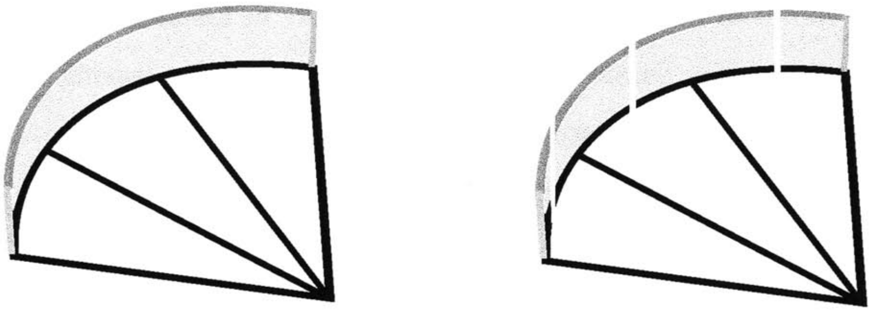


Figure 3-2: Uniform measure on a the arc of a circular sector with additional spokes. The agent is at the center of the circle. The minimum latency circuit starts with an edge-most spoke, while DENSESUBNET initially selects an interior spoke.

the edge-most spokes. This is not what we want as it is obviously inconsistent with aiming off—it is surely preferable to pursue one of the edge-most spokes first.

The problem is that when converting the network into a tree, Algorithm NETTOTREE divides the measure of continuous sub-regions between different sub-trees. We propose the following augmentation to mitigate this effect. In the following we ignore the case in which the densest sub-tree does not exist, *i.e.*, Equation 3.3 is maximized by a sequence of sub-trees converging to one with total length zero.

After applying the DENSESUBNET Algorithm, we consider replacing an initial portion of it with a more profitable path. We do this by constructing a graph with edge lengths corresponding to their profit under the level of subsidy associated with that tree.

Let $\mathcal{S}$ be the sub-tree returned by DENSESUBNET with root a_0. Let $\gamma^* = \frac{\mathcal{L}(\mathcal{S})}{\mu(\mathcal{S})}$. Let a_1 denote the location of the first branching of that tree, or the leaf if that tree has no branchings. Without loss of generality, assume that $a_0, a_1 \in \mathcal{V}$. Let $\mathcal{P}_{10}$ denote the path from a_1 to a_0 in $\mathcal{S}$.

Define $r^- : [0,1] \to \mathbb{R}$ as

$$r^-(t) = td(a_1, a_0) - \gamma^* \int_0^t d\mu(\mathcal{P}_{10}(t\mathcal{L}(\mathcal{P}_{10}))). \tag{3.5}$$

Lemma 3.2.7. $r^-(1) \geq 0$

Proof. Suppose the contrary. The path $\mathcal{P}_{10}$ is a tree that is a sub-tree of the Geometric Spanning Tree for which $\frac{\mathcal{L}(\mathcal{P}_{10})}{\mu(\mathcal{P}_{10})} < \gamma^*$ contradicting the optimality of $\mathcal{S}$ on the DSP.□

Let

$$t^* \equiv \begin{cases} 0 & \text{if } \{t \text{ s.t. } r^-(t) \leq 0\} = \emptyset \\ \arg\min_{t\in[0,1]} r^-(t) & \text{o.w.,} \end{cases} \tag{3.6}$$

and let a^* denote $\mathcal{P}_{10}(t^*)$. Note that if we apply Algorithm NODEADDING with subsidy γ^*, then a^* will be a vertex in the new region.

Next, we will attempt to find a shortest path between a_0 and a^* in a weighted graph whose edge weights correspond to subsidized cost. We construct a weighted graph with vertices $\mathcal{V}$ and edges $\mathcal{E}$ and edge weights $\mathcal{L}'(e) = \mathcal{L}(e) - \gamma^*(\mu(e))$. Note that these edges have both positive and negative weights and so computing shortest paths is still potentially difficult.

The way we deal with this problem is by breaking negative cycles that do not contain a_0 by removing the edge with largest length. If we find a negative cycle containing a_0, this represents a better solution than the specified tree and so we return that. If not, then eventually there are no negative weight cycles and the Bellman-Ford Algorithm can find the shortest path from a_0 to a^*. Algorithm REPLACEPREFIX describes this procedure in detail.

Algorithm NetworkIndexPolicy, summarizes the entire algorithm, which we will refer to as the "simplified index policy," for search in geometric networks in polynomial time.

3.3 Observable Problems in Arbitrary Metric Spaces

One can consider the well-studied combinatorial optimization problem known as the Minimum Latency Tour Problem (MLTP) as a special case of "search" with no uncertainty. This section will apply our proposed policy to the MLTP and analyze its performance.

The MLTP is the problem of finding the tour that minimizes the latency of the requests. Formally, let θ_0 denote the starting location of the agent and θ be the set of requests with $|\theta| = n$ all elements of a metric space. The MLTP is to find an ordering of the requests minimizing

$$\sum_{i \in 1,\ldots,n} \sum_{j \in 1,\ldots,i} d(\theta_{j-1}, \theta_j)$$

Note the difference between this and the ordinary TSP: each inter-request distance is included in the latency of every subsequent request. As we have already noted, the MLTP is NP-hard even with a tree topology.

The literature on the MLTP has been exclusively concerned with achieving the lowest provable approximation factor. Currently that factor resides at 3.58[26] which can be achieved by an algorithm requiring $O(n^3 \log(n))$ time. Although the nearest neighbor algorithm lacks a provable constant factor, we will show that its *expected* performance on randomly generated instances in the Euclidean plane is very good.

Problem 3.3.1 (Prize Collecting Steiner Tree Problem (PCSTP)). *Given a weighted graph $\langle \mathcal{V}, \mathcal{E}, \mathcal{L} \rangle$, a root node $x \in \mathcal{V}$, and a function $R : \mathcal{V} \to \mathbb{R}$, the PCSTP is to find an connected sub-graph $\mathcal{T}$ containing x, that maximizes*

$$\sum_{n \in \mathcal{T}} R(n) - \sum_{e \in \mathcal{T}} \mathcal{L}(e)$$

Problem 3.3.2 (Prize Collecting Stroll Problem (PCSP)). *The PCSP is the same as the PCSTP in which the sub-graph $\mathcal{T}$ must be an open path starting from x.*

The authors of [60] provide an algorithm with a constant factor of 1.665 for the MLTP with unit edge costs. This is notable because this factor is less than half that of the best available for the general problem. We generalize this approach to general distances in Algorithm ModKoutsoupias but in so doing sacrifice the guarantee.

Section 2.2.3 introduced a technique for applying the index policy to problems involving multiple goals. Using this, we can treat MLTP as a special case of a measure over the geometric network consisting of the completely connected graph of the requests. The measure is one that puts probability one on each of the vertices. However since the policy applies a normalization, this is equivalent to the one which assigns a probability of $1/n$ to each node and renormalizes each time a node is visited.

The index policy reduces to finding a sequence of length k minimizing

$$\frac{\frac{1}{n}\sum_{i\in 1,\ldots,k}\sum_{j\in 1,\ldots,i} d(\theta_{j-1},\theta_j) + \left(1-\frac{k}{n}\right)\left(\sum_{i\in 1,\ldots,k} d(\theta_{i-1},\theta_i) + d(\theta_k,\theta_0)\right)}{\frac{k}{n}}$$

$$\frac{1}{k}\sum_{i\in 1,\ldots,k}\sum_{j\in 1,\ldots,i} d(\theta_{j-1},\theta_j) + \left(\frac{n}{k}-1\right)\left(\sum_{i\in 1,\ldots,k} d(\theta_{i-1},\theta_i) + d(\theta_k,\theta_0)\right) \tag{3.7}$$

It is important to note that Equation 3.7 lacks an important quality of a good heuristic: tractability. In fact, solving Equation 3.7 is at least as difficult as solving the MLTP directly. Let us remind the reader that our main goal is to present a general purpose policy and this serves as another example of a problem to which it can be applied. The same machinery discussed in Section 3.2.1 provides polynomial approaches, which we re-examine in the context of the observable MLTP.

Specifically, we replace the MLTP with the MRCP which now has the following formulation

$$\min_{\text{circuits}} \frac{\sum_{i\in 1,\ldots,k} d(\theta_{i-1},\theta_i) + d(\theta_k,\theta_0)}{k}. \tag{3.8}$$

Solving Equation 3.8 is still hard. In fact, even if we replace the requirement of a tour with a tree, the problem is equivalent to the PCSTP, which is NP-hard.

At this point we must resort to further heuristics, for instance those in the literature or that described in Section 3.2.1. For the latter, Algorithm NetToTree will generate the minimum spanning tree, and the sub-tree returned by Algorithm DenseSubnet has a natural interpretation as the most node-dense, rooted sub-tree of the minimum spanning tree. We suppose that Algorithm ReplacePrefix provides little to no improvement.

Conjecture 3.3.1. *The index policy both with and without the above simplifications (specifically, Algorithm* NetworkIndexPolicy*) has a constant-factor guarantee for the MLTP as does its simplified version .*

Argument. This conjecture is based on the fact that many of the algorithms with such guarantees (including, in particular [26]) are based on repeatedly approximating the PCSP or PCSTP. The index policy and it's simplified version do likewise. □

3.3.1 Some Experimental Results

To put the following results into context, let us recapitulate how we arrived here. By theoretically analyzing the CPP, we discovered an index policy. We then generalized that policy *by analogy* to search problems in more general one-dimensional spaces. Next, we introduced *simplifications* of that policy to improve tractability. Since we can model zero-dimensional spaces in this framework, we find that a difficult but well-studied combinatorial optimization problem is a special case of the set of problems we have already addressed.

We examine the performance of the Index Policy, Nearest Neighbor, and Algorithm ModKoutsoupias on instances of the MLTP generated uniformly in the Euclidean unit square.

3.3.2 Multi-Planners

Recall that we are trying to minimize the index function, Equation 2.5, over the set of closed paths. All of the work in this section has been working around the issue raised

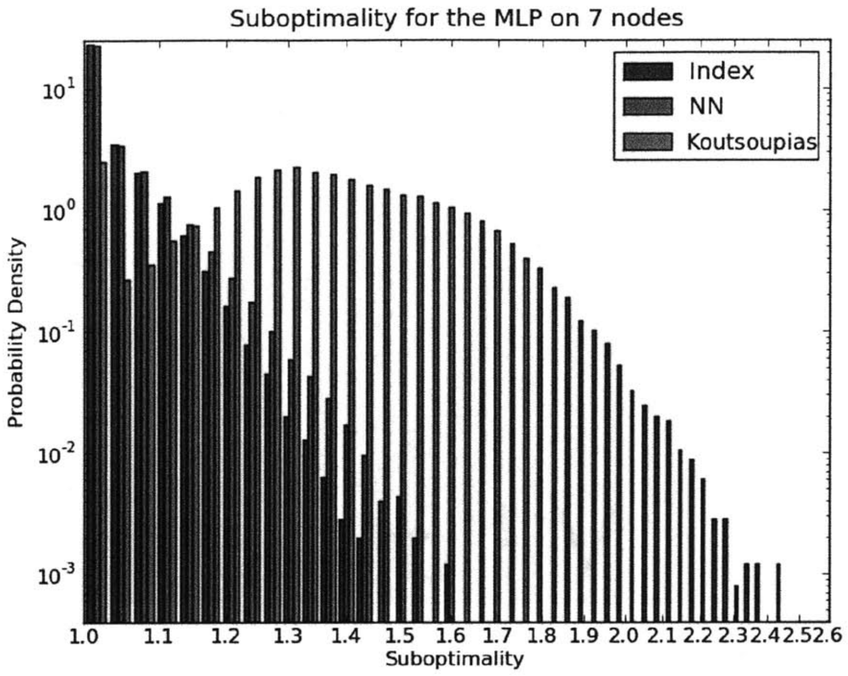


Figure 3-3: Sub-optimality histogram for the MLTP with 7 nodes

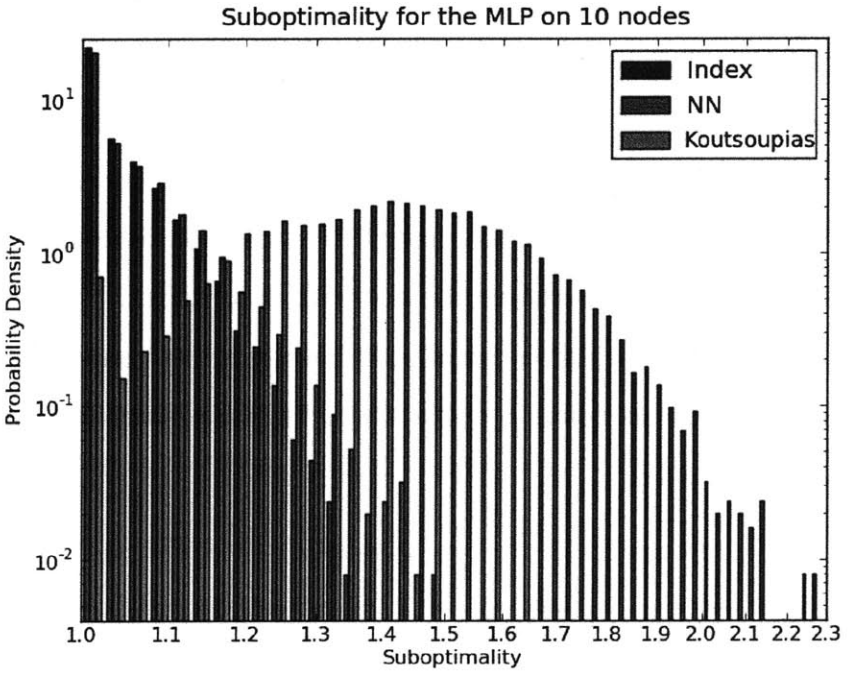


Figure 3-4: Sub-optimality histogram for the MLTP with 10 nodes

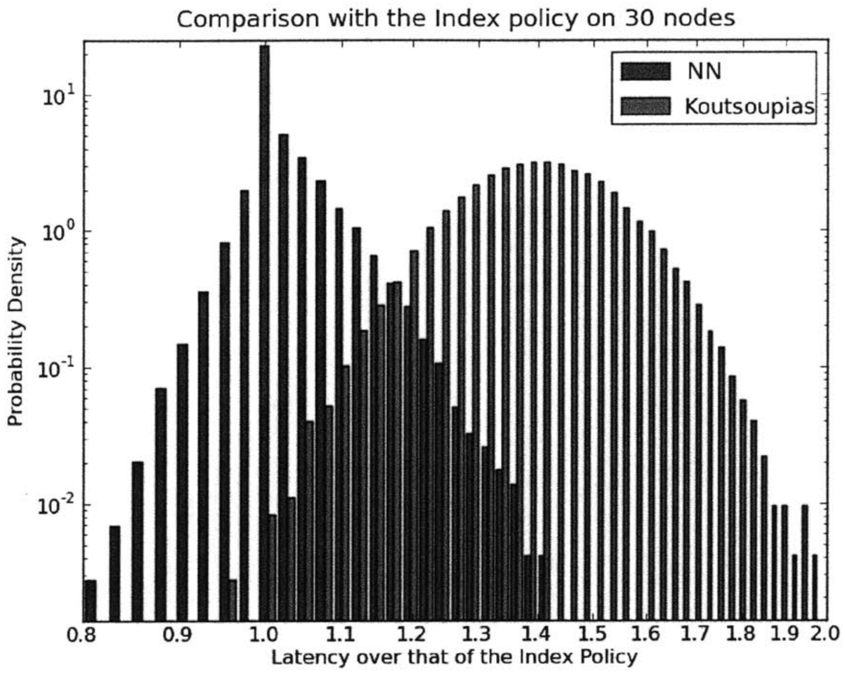


Figure 3-5: Comparison with the index policy for the MLTP with 30 nodes

by the large number of such paths. This too will be the focus of much of Chapter 4. Nonetheless, the *thesis* of this dissertation is simply that the index function is a good metric by which to compare paths.

In particular, given a small set of paths, we can easily compare them regardless of how they were generated. For example, one need not choose between Algorithm NETWORKINDEXPOLICY or an algorithm for the MPPP with a provable guarantee: one can run both and at each instant choose the one with lower index.

Given a set of planners, the indices returned by these planners provide a means for comparing them. This lets us define a *multi-planner* which at each moment chooses the one with lowest index.

3.4 Metric Embedding

This section presents a methodology for searching one-dimensional subspaces of arbitrary metric spaces. We will approach the problem by approximating the topology of

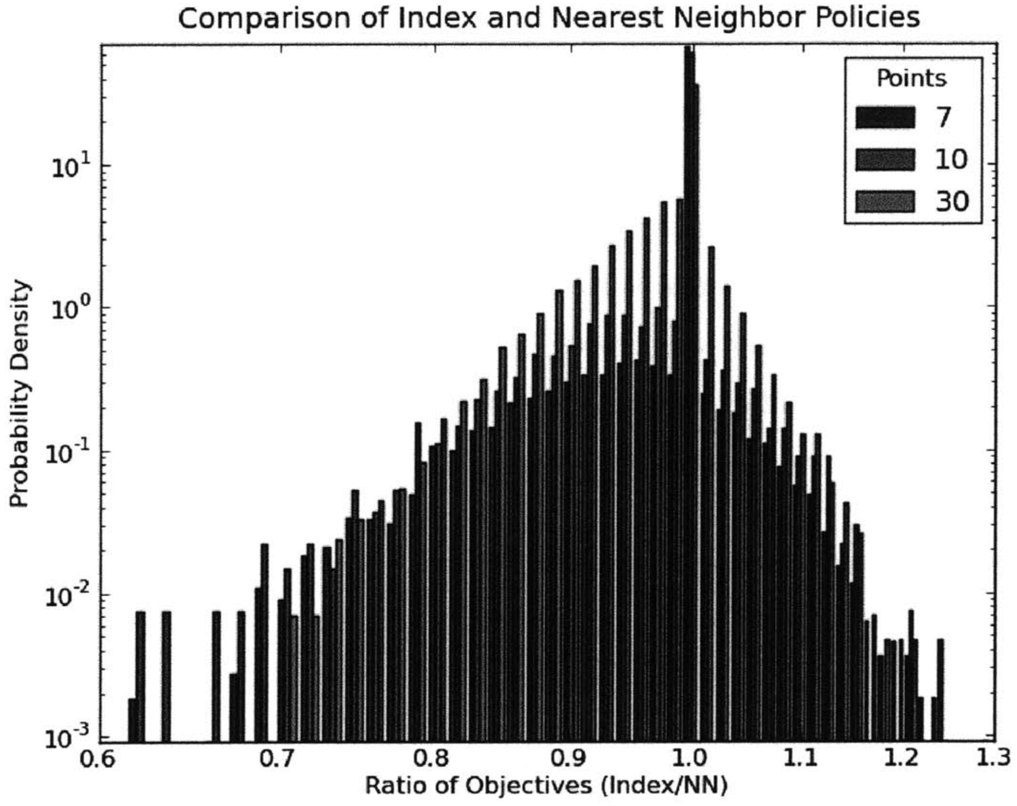


Figure 3-6: Comparison of index and nearest-neighbor policies on the MLTP

the metric space with a geometric network. We argue that this is valid because the outer space is relevant only insofar as it enhances the connectivity of the space being serached. The methodology presented in this section can fairly be called *ad hoc* and fails to meet the standard for an "algorithm" as defined in Section 1.1. Nonetheless, it will form a component of an important unifying example in Section 5.8.

Suppose the region is an arbitrary metric space, but goals are restricted to a subspace with dimension one, which we will refer to as the "goal network" G. This is a natural model for aerial vehicles performing surveillance of road networks, for example. We will tackle this by modeling the region as a geometric network corresponding to the goal subspace and adding finitely many edges to describe possible motion of the agent in the larger space.

What differentiates aiming off, discussed in Section 2.3, is simply that the goal distribution is fixed. If the agent is initially very far from the road, the variance of the goal distribution will not vary greatly by heading. In such a case, the problem of aiming off can be thought of as an instance of the current problem.

We were able handle the problem of aiming off because we could assume that

- the agent should go straight to the road, and
- once on the road the agent should never leave.

The first assumption should be no less valid, but the second is certainly questionable. For instance, in two-dimensional space, the agent might quite routinely *cross* roads without any intention of searching them at all.

Given some minimum distance δ, we can break each edge e into $\lceil \mathcal{L}(e)/\delta \rceil$ new segments and then add edges $\mathcal{E}^+$ to form the completely connected graph in the new node set. This new network G' will have order of $\frac{\mathcal{L}(G)^2}{\delta}$ edges.

Remark 3.4.1. *Let A denote the mapping from locations in G to G'.*

$$\max_{a_0, a_1 \in G} d(a_0, a_1)_{\mathcal{A}} - d(A(a_0), A(a_1))'_G \leq \delta \tag{3.9}$$

From the particulars of Algorithm NETWORKINDEXPOLICY and the issues described in Section 3.2.2, its not clear that we even *want* a fine discretization even if it is convenient computationally. In particular, as we add direct paths from the agent's location to each small segment, the Algorithm DENSESTSUBTREE will ultimately choose the one with the smallest ratio of distance to probability density.

To prevent this, we propose the following means of culling the set $\mathcal{E}^+$. First sort these edges by increasing length, then iteratively add them provided that

$$\frac{d(A(a_0), A(a_1))'_G}{d(a_0, a_1)_\mathcal{A}} > \delta_2$$

In effect we are creating the graph in which for all location pairs, a_0, a_1 we have distance error bounded by *either* an absolute error of δ *or* fractional error of δ_2. This procedure is described in detail in Algorithm METRICEMBEDDING.

3.5 Multiple Agents

The primary focus of this dissertation is search involving a single agent. However, a number of significant applications exist in which multiple agents search *cooperatively.* This section presents possible approaches for applying the index policy to such applications.

In the CPP there was a natural and obvious way to extending the index policy to a multi-agent setting. We knew that we would never want two cows searching the same path and so we could safely assign paths to cows in increasing order of index. This follows from the decoupled nature of bandit problems themselves.

In problems that are not indexable, for instance search on geometric networks, single-agent policies are not so readily generalized into multiple agent policies. Consider the index policy when applied independently and simultaneously by multiple agents. It is entirely possible that the paths that different agents choose to minimize Equation 3.3 may have a non-empty intersection. However, only one of the agents can search a particular location first.

We address this by assigning the intersection to the agent whose path has lower subsidy. This is not the only possible approach, for instance we could assign the intersection based on whichever agent arrives first. We chose this solely because it lends itself to a simple algorithm that is polynomial in the number of agents.

Algorithm auctionPlanner defines a winner-take-all auctioning approach. Each agent computes the path with minimum price and the agent whose path has lowest price is assigned to it. Then the measure associated with that path is removed and the unassigned agents recompute their minimum price paths. This is repeated until each agent is assigned a path.

It should be remarked that this will not necessarily assign cows to different paths in the CPP. Consider two cows on the real-line. Clearly they should go in opposite directions. But if the optimizing path for the first cow is short (or infinitesimal), then the second cow might also try to go in the same direction but plan on going further.

We suppose that this problem could be mitigated by forward simulation. Instead of only zeroing-out $\mathcal{P}_i$ in Line 12 of Algorithm auctionPlanner, one could forward simulate agent i to get a path $\mathcal{P}_i'$ of some minimum length and zero-out $\mathcal{P}_i'$ before the next round of planning. We do not investigate this possibility. Furthermore we believe a more principled treatment of this issue is likely one of the more fruitful directions in which this dissertation could be extended.

An alternative approach to handling the multiple-agent case is simply to partition the region into as many sub-regions as agents and have each agent operate independently. We will examine this strategy in Chapter 5.

Chapter 4

Search in Higher Dimensional Spaces

This chapter generalizes the index policy developed in Chapters 2 and 3 to higher dimensional spaces.

We will concentrate on two main problems. The first of these will be the Lost in a Forest Problem (LFP) in Section 4.2 in which an agent moves with perfect dynamics and a known environment but from an unknown starting state. For this problem we will restrict ourselves to Riemannian Manifolds in which the goal includes the (possibly empty) boundary of the space. The second is a generalization of the Minimum Weighted Latency Covering Problem (MWLCP) (Problem 3.1.1) in which an agent with a perfect, finite-radius sensor moves with known position but the location of the goal is unknown. For this problem we can consider general length spaces and place no special requirements on their boundaries.

In Section 4.4 we will briefly attempt the problem of optimal control in which the agent has complex, noisy dynamics and a limited sensor but nonetheless wishes to reach a goal configuration.

We do not revisit the problem of multiple agents in this chapter. We make the following remarks.

- m agents in a region $\mathcal{A}$ can be treated as a single agent in $\mathcal{A}^m$.

- Partitioning policies remain possible in length spaces.
- Algorithm AUCTIONPLANNER remains applicable and about it we making the following conjecture.

Conjecture 4.0.1. *The expected intersection between maximizing paths selected by different agents decreases as the dimension of the space increases. As this happens, the performance of* AUCTIONPLANNER *can only improve.*

Argument. This conjecture is based simply on the fact that distances expand as dimension increases. Since Algorithm AUCTIONPLANNER encourages agents to separate, this fact should serve to make doing so easier. □

4.1 Preliminaries

Lemma 4.1.1 (Shortest Path Principle). *In a length space, sub-paths of shortest paths are shortest paths.*

Proof. Assume the contrary and consider a shortest path $\mathcal{P}_{01}$ from a_0 to a_1 that has length ℓ_{01}. Without loss of generality assume that there exists a path $\mathcal{P}_0$ from a_0 to $a = \mathcal{P}_{01}(x)$ for some $x \in (0, \ell_{01})$ of length $\ell_0 < x$. We can construct a new path as follows:

$$\mathcal{P}'_{01}(y) = \begin{cases} \mathcal{P}_0(y) & \text{for } y \in [0, \ell_0) \\ \mathcal{P}_{01}(x + y - \ell_0) & \text{for } y \in [\ell_0, \ell_{01} - x + \ell_0] \end{cases}$$

Which is continuous at ℓ_0,has the same endpoints as $\mathcal{P}_{01}$, and has length $\ell_{01} - x + \ell_0 < \ell_{01}$. To establish the contradiction, all that remains is to verify uniformity.

$$\begin{aligned} \lim_{h \to 0} \frac{d(\mathcal{P}'_{01}(\ell), \mathcal{P}'_{01}(\ell - h))}{h} &= \lim_{h \to 0} \frac{d(\mathcal{P}_0(\ell), \mathcal{P}_0(\ell - h))}{h} = 1 \\ \lim_{h \to 0} \frac{d(\mathcal{P}'_{01}(\ell + h), \mathcal{P}'_{01}(\ell))}{h} &= \lim_{h \to 0} \frac{d(\mathcal{P}_{01}(x + h), \mathcal{P}_{01}(x))}{h} = 1. \end{aligned}$$ □

Definition 4.1.2 (Riemannian Manifold). *A Riemannian Manifold is a length space that is locally Euclidean. In particular, a Riemannian Manifold has constant, integer point-wise dimension.*

A Riemannian Manifold is a metric space in which the notions of "angle" and "straight" are meaningful. In particular, the shortest paths on the interior of the space are defined to be "straight" and from this, we can define the set of geodesics which are everywhere locally straight. Additionally, since the space is locally Euclidean, there is a well-defined notion of the angle between paths at a point of intersection.

4.2 Lost in a Forest

In [16] Bellman posed the following problem:

Problem 4.2.1 (Minimization problem). *We are given a convex planar region $\mathcal{A}$ and a random point P within the region. Determine the paths which*

1 Minimize the expected time to reach the boundary, or

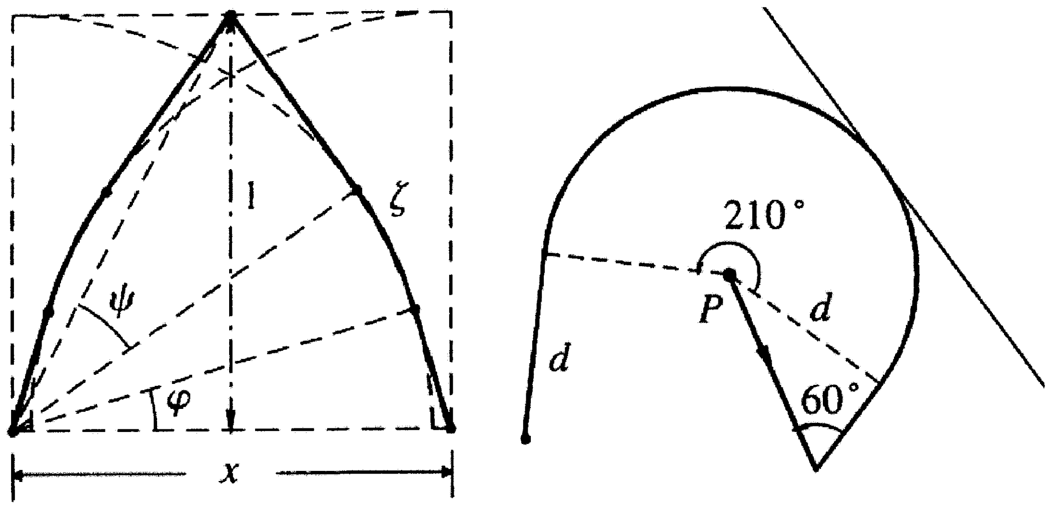
2 Minimize the maximum time required to reach the boundary.

Consider, in particular, the cases

a *$\mathcal{A}$ is the region between two parallel lines at a known distance d apart.*

b *$\mathcal{A}$ is the semi-infinite plane and we are given the distance d from the point P to the bounding line.*

This problem was then known as "Lost in a fog" and later "Lost in a forest." We will refer to Problem 4.2.1-1 as the Min-mean Escape Problem and Problem 4.2.1-2 as the Min-max Escape Problem. Section 4.2.3 develops an index policy for the Min-mean Escape Problem not just for convex planar regions, but for regions which are Riemannian Manifolds. Sections 4.2.1 and 4.2.2 describe all the cases of the each of these problems that have been solved as well as those for which any results are available whatsoever. In Section 4.2.5 we will use them as test cases for the general policy that we will develop.



(a) Zalgaller's path solving Problem 4.2.1-2(a) (b) Isbell's path solving Problem 4.2.1-2(b)

Figure 4-1: Solutions to Problem 4.2.1-2 Cases (a) and (b). From [35] [used with permission]

4.2.1 Existing Work on the Min-Max Escape Problem

The authors of [35] provide a comprehensive survey on the Min-max Escape Problem, which we now summarize. Problem 4.2.1-2(a) was solved in [94] by what is known as the Zalgaller path which is shown in Figure 4-1a having parameters

$$\begin{aligned} \phi &= \arcsin\left(\frac{1}{6}+\frac{4}{3}\sin\left(\frac{1}{3}\arcsin\left(\frac{17}{64}\right)\right)\right) \\ \psi &= \arctan\left(\frac{1}{2}\sec(\phi)\right) \end{aligned}$$

Problem 4.2.1-2(b) was solved in [49] and is shown in Figure 4-1b.

Definition 4.2.1 (Fat). *Let $\mathcal{A}$ be a compact, convex subset of the plane of diameter $\oslash$. $\mathcal{A}$ is* fat *if a 60-degree rhombus with major axis $\oslash$ can be fit inside it.*

The min-max escape path for any fat polygon is a straight line[78]. The min-max escape path for a rectangle is either Zalgaller's path or a straight line whose length is that of the diagonal, whichever is shorter[35].

There are two more cases for which there are accepted but not rigorously proven answers, those being the equilateral triangle and finding a circle from a known

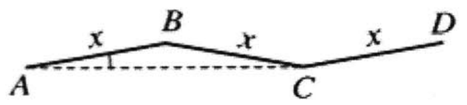
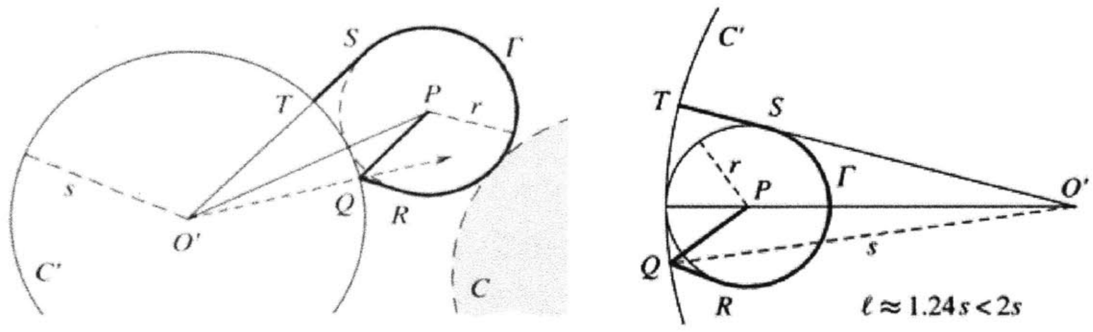


Figure 4-2: Besicovitch's path for escaping an equilateral triangle with unit edge length is the optimum over the set of paths with $AB = BC = CD = x$ and $\angle ABC = \angle BCD = \theta$: with $x = \sqrt{21}/14$ and $\theta = \pi - 2\arcsin(1/\sqrt{28})$.



(a) External Case

(b) Internal Case

Figure 4-3: Gluss's path for finding a circle of radius s from a known distance r. Geometry: $O'TS$ is straight, QR is tangent to Γ and ray $O'Q$ bisects $\angle PQR$. From [35] [used with permission]

distance. For escaping the equilateral triangle, the authors of [24] optimized over parametrized path shown in Figure 4-2 which is hypothesized to be optimal for all paths.

For the problem of finding the circle, the agent initially knows the distance to a circle and whether it is on the interior or exterior, but does not know the direction to the circle's center. The solution proposed by [47] to the external case is an extension of [49] (Figure 4-1b) and is shown in Figure 4-3a.

In [35] the authors examine the interior case and show that the path proposed in [47] for the exterior case is still preferable to a straight path when the ratio of the initial distance over the circle radius is smaller than about one third. This path is shown in Figure 4-3b.

4.2.2 Existing work on the Min-Mean Escape Problem

For the Min-mean Escape Problem very little is known. It is conjectured (*e.g.* in [36]) that a straight path is optimal for the uniform distribution inside a disk. In [36], the authors determine the optimal two- and three-segment paths for the infinite strip (Case (a)). The work [94] also addresses the Min-mean Escape Problem but contains some error, as pointed out by [36] and our results confirm.

For the half-plane (Case (b)), [46] optimizes over paths initially identical to Figure 4-1b but ending with a quadratic curve.

4.2.3 The Index Policy

Before developing the policy, let us restate the LFP in the terminology of the control of mobile agents.

Problem 4.2.2 (The Lost in a Forest Problem (LFP)). *Assume that a mobile agent's state space is a Riemannian Manifold that we will call the region, $\mathcal{A}$, a subset of which we will call the "goal," G, which includes the (potentially empty) boundary of the region. The agent would like to reach the goal from an unknown starting pose, which is measured by μ_0. Assume that the agent is a noiseless single integrator and that its only sensing capacity is that it can tell whether or not it has reached the goal. The Lost in a Forest Problem (LFP) is to find a control policy that finds the goal in minimum expected time.*

The reason it is important that the goal include the boundary is that the motion at the boundary introduces a major complexity. Specifically, there is *information* to be gained on the boundary, and computing the *value of information* is fundamentally difficult. For instance if the region contains a corner, it may be possible for the agent to localize by moving in such a way as to reach it. Doing so might be optimal even if it involves moving away from the goal.

Recall that the index policy as described in Chapter 3 is to pursue the closed path $\mathcal{P}$ starting from the agent's location a minimizing

$$\gamma^*(a) = \min_{\mathcal{P}} \frac{w(\mathcal{P}) + (1 - \mu(\mathcal{P}))\mathcal{L}(\mathcal{P})}{\mu(\mathcal{P})}. \tag{4.1}$$

wherein w denotes the weighted latency of the path, $\mathcal{L}$ its length, and $\mu(\mathcal{P})$ is the probability that the path intersects the goal. The interpretation of this metric remains valid.

The difficulty is in using it. In higher dimensions, the set of closed paths is much more complex than in a network. To be able to apply the index policy to these spaces, we must restrict the set of paths to a searchable space. In Riemannian Manifolds, we will consider the set of *straight* paths.

We begin with some notation. From the geometry of the Riemannian Manifold define the set of headings, Θ, which describe all of the geodesically straight paths. Let $\mathcal{P}^s_{x,\theta}$ denote the geodesic path of heading $\theta \in \Theta$ starting from x. For example, on the surface of a sphere the set $\bigcup_{\theta\in\Theta} \mathcal{P}^s_{x,\theta}$ enumerate the great circles that pass through x. Note that these paths need not be closed or have finite length, for instance helices on tori.

Let $\mathcal{P}^s_{x,\theta}(t)$ denote the location of an agent after pursuing path $\mathcal{P}^s_{x,\theta}$ for time t from initial state x. Let $I_G(\mathcal{P}, t)$ be the indicator of whether the set $G \cap \bigcup_{\tau\in[0,t]} \mathcal{P}(\tau)$ is non-empty.

We define a measure on $\mathbb{R}_+$, Φ_θ such that $\Phi_\theta([a, b]) = \mathbb{E}_x \left[I_G(\mathcal{P}^s_{x,\theta}, b) - I_G(\mathcal{P}^s_{x\theta}, a)\right]$. Note that Φ_θ is not necessarily a probability measure, as it does not necessarily integrate to one in an unbounded region or a region containing straight, closed paths.

Given a heading θ, consider now the problem of the optimal subsidized search distance. In this problem, the agent may search any distance in the θ direction and pays unit cost for motion. If it finds the goal it receives payout γ, but if it does not, it is obligated to return to it's starting point.

The expected payout for searching a distance t is given by

$$\rho(\gamma,t) \equiv \gamma\Phi_\theta([0,t]) - \int_0^t \tau d\Phi_\theta(\tau) - \left(t + \underset{\tau|I_G(\mathcal{P}^s_{x,\theta},t)=0}{\mathbb{E}}\left[d(\mathcal{P}^s_{x,\theta}(\tau),x)\right]\right)(1-\Phi_{\mathcal{P}^s_{x,\theta}}([0,t])). \tag{4.2}$$

In spaces for which $d(\mathcal{P}^s_{x,\theta}(t),x)$ is non-decreasing in t, *e.g.*, Euclidean spaces, we have the simplification,[1]

$$\rho(\gamma,t) \equiv \gamma\Phi_\theta([0,t]) - \int_0^t \tau d\Phi_\theta(\tau) - 2t(1-\Phi_{\mathcal{P}^s_{x,\theta}}([0,t])). \tag{4.3}$$

The minimum subsidy at which any distance is profitable is therefore

$$\gamma^*(\theta) \equiv \inf\{\gamma \text{ s.t. } \sup_{t>0}\rho(\gamma,t) \geq 0\} \tag{4.4}$$

Policy 4.2.1 (Index Policy). *We define the* index policy *as the one that always selects the heading minimizing*

$$\min_{\theta\in\Theta}\gamma^*(\theta). \tag{4.5}$$

To fit the original statement of the LFP into this framework, we let the region be the that of the forest crossed with the initial heading, *i.e.* $[0,2\pi)$. The goal set is that of the original problem, again crossed with $[0,2\pi)$. Let the new distance metric $d(\cdot,\cdot)$ be any metric that is consistent with the euclidean distance in the original forest, $d(\cdot,\cdot)_F$. In Lemma 4.2.2 we prove, from the symmetry of the goal set, that we do not need to consider any policies that turn.

Lemma 4.2.2. *Let $d(\cdot,\cdot)_\theta = d(\cdot,\cdot) - d(\cdot,\cdot)_F$, taking the necessary projections. The choice of $d(\cdot,\cdot)_\theta$ is not relevant because no optimal policy turns.*

Proof. Let Π be an optimal policy and assume, by way of contradiction that it turns. Specifically, let $\langle \mathcal{P}_{x,\phi}(t)_z, \mathcal{P}_{x,\theta}(t)_\theta\rangle$ denote the two components of the agent's location at time t along the path taken by policy Π starting from location $\langle x,\phi\rangle$. Let $t^*(x,\phi), g^*(x,\phi)$ denote the time and place at which this path that reaches the goal

[1]We also suppose that for spaces lacking this property, approximating Equation 4.2 with Equation 4.3 will have a relatively limited effect.

set. We will denote the correspondence using $\mathcal{P}_{x,\phi}^{-1}(g^*) = t^*$. The expected search time can be written as

$$\mathop{\mathbb{E}}_{x,\phi}\left[t^*(x,\phi)\right] = \mathop{\mathbb{E}}_{x,\phi}\left[\mathcal{P}_{x,\phi}^{-1}(g^*)\right] \tag{4.6}$$

Let Π' be the same as Π except without turning and let $\mathcal{P}'_{x,\phi}$ denote the first component of the path taken by Π' from starting state (x,ϕ). Since the goal set is symmetric we know that Π' also finds the goal at $g^*(x,\phi)$. Therefore the expected search time under Π' is also given by

$$\mathop{\mathbb{E}}_{x,\phi}\left[\mathcal{P}'^{-1}_{x,\phi}(g^*)\right] \tag{4.7}$$

But examine arc lengths of these two paths. For clarity we write the metric $D_S(x,y)$ rather than $d(x,y)_S$.

$$\begin{aligned}
t^* &= \int_0^{t^*} D\left(\langle \mathcal{P}_{x,\theta}(t)_z + \frac{d}{dt}\mathcal{P}_{x,\theta}(t)_z, \mathcal{P}_{x,\theta}(t)_\theta + \frac{d}{dt}\mathcal{P}_{x,\theta}(t)_\theta\rangle\ ,\ \langle \mathcal{P}_{x,\theta}(t)_z, \mathcal{P}_{x,\theta}(t)_\theta\rangle\right) dt \\
&= \int_0^{t^*} D_F\left(\mathcal{P}_{x,\theta}(t)_z + \frac{d}{dt}\mathcal{P}_{x,\theta}(t)_z\ ,\ \mathcal{P}_{x,\theta}(t)_z\right) + \\
&\qquad D_\theta\left(\mathcal{P}_{x,\theta}(t)_\theta + \frac{d}{dt}\mathcal{P}_{x,\theta}(t)_\theta, \mathcal{P}_{x,\theta}(t)_\theta\right) dt \\
&> \int_0^{t^*} D_F\left(\mathcal{P}_{x,\theta}(t)_z + \frac{d}{dt}\mathcal{P}_{x,\theta}(t)_z, \mathcal{P}_{x,\theta}(t)_z\right) dt \\
&= \int_0^{t'^*} D\left(\mathcal{P}'_{x,\theta}(t)_z + \frac{d}{dt}\mathcal{P}'_{x,\theta}(t)_z, \mathcal{P}'_{x,\theta}(t)_z\right) dt \\
&= t'^*
\end{aligned}$$

We find that $\mathcal{P}'$ is shorter, contradicting the optimality of Π □

As a result of Lemma 4.2.2 we may restrict our attention to control policies that do not modify the initial orientation.

In problems in which the goal set is not symmetric to orientation, we would need specific information about the metric to properly formulate the problem. Specifically, if there is a time-cost for turning, this describes a distinct metric which we must use instead of $d(\cdot,\cdot)_F$. If there is no such cost, then we are once again free to disregard turning: The agent could simply spin with infinite speed and pursue the solution to the problem in which the goal is taken to be the union over orientations.

4.2.4 Implementation Details

The policy we have described is a function from measures over a Riemannian Manifold to headings. Since arbitrary measures are not finite-dimensional there remain some practical details to nail down. In particular, there is the matter of how sensitive the policy is to our choice of representation.

For instance, supose we track the probability distribution using a particle filter. It is possible that the minimizing t in Equation 4.4 is very short and gets a exactly one particle to the goal. Similarly if we implement this policy in discrete time, there is the problem that the time interval might be unable to accurately capture the supremum. We address this problem by replacing the $\inf_{t>0}$ in Equation 4.4 with $\min_{t\geq\delta_t}$ and round small latencies up to δ_t.

$$\min_{\theta\in\Theta}\min_{t\geq\delta_t}\frac{\int_0^t \max(\tau,\delta_t)d\Phi_{\mathcal{P}_s(\theta)}(\tau)+2t(1-\Phi(0,t))}{\Phi(0,t)}.$$

Nonetheless, without a very large number of particles or large δ_t this has the potential to lead to erratic behavior which is driven by the realization of the few particles closest to the goal. While "particle depletion" is an inherit difficulty of the particle filter itself, the structure of the index policy serves to amplify this issue. For the LFP with uniform support over convex regions this is not a serious consideration. Such depletion can only happen late in the trajectory: after the important decisions have been made. For supports and regions in which "unlikely observations" can happen earlier, a particle filter implementation of the index policy will be sensitive to the resampling strategy.

An additional concern is that Equation 4.5 might have multiple minima. Discretization over time and over headings can lead to zig-zagging when going straight would be preferable. We illustrate this by examining the index policy for the problem of reaching a circle of radius r from an initial distance of r_0 outside it.

Figure 4-4 depicts the state at an intermediate time. The green arc Γ, with radius $s = r_0 + r$, is the set of possible agent locations with respect to the blue goal set G. We will examine the two headings shown in red.

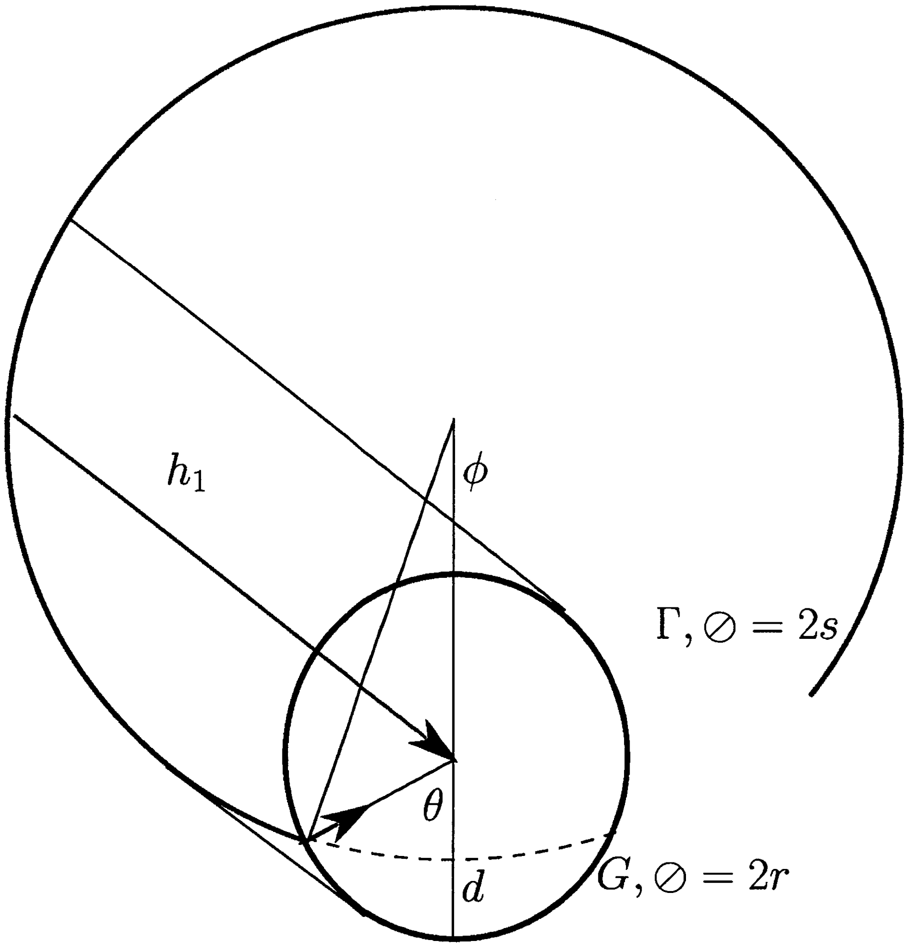


Figure 4-4: Two distinct headings minimizing Equation 4.5

The distance d is the agent's radial distance from its starting point minus r_0. If the distance d is sufficiently large, the heading selected will be h_1, but for sufficiently small $d > 0$ consider moving in direction $\theta = \arccos(1 - d/r)$ a small distance δ_t. Let $\phi = \arccos(1-d/s)$ and let L denote the angular length of Γ. When $L \geq (\theta-\phi)+\pi/2$, we have

$$\gamma^*(h_1) \approx 2s\left(\frac{L}{(\theta-\phi)+\pi/2} - \frac{3}{4}\right) \tag{4.8}$$

and

$$\gamma^*(\theta) \approx 2sL\cos(\theta-\phi). \tag{4.9}$$

From these expressions, we should expect the policy to following heading tracking θ until the distance d is reached whereupon the two headings are have equal subsidy. At this point we have two distinct headings minimizing Equation 4.5. Pursuing either for a non-zero distance will push d to be too large or too small, resulting in the oscillation visible in Figure 4-9.

Naturally, the period of this oscillation can be reduced by decreasing δ_t. However, if we also discretize the set of headings, the oscillations will persist. The period of the large initial oscillations in Figure 4-8 are over 100 time-intervals even while the set of headings is discretized at a fidelity of $\pi/128$—less than 1.5 degrees.

An obvious approach to dealing with this issue is to forward simulate the policy and apply smoothing. Since the topic of "general" approaches to smoothing is vast in itself, we will not address it in particular. We simply remark that for the LFP at least, it is not difficult to find, algorithmically, a frequency cut-off that effectively differentiates the signal from this oscillation.

In the implementations that we use to generate the results in Section 4.2.5, we use discrete time (Equation 4.2.4) and heading. When the smoothed path is significantly different we will show both.

We represent probability using a particle filter. As we previously implied, the resampling strategy can have a profound effect on the outcome. We used two approaches. In one, we assume a very small amount of noise in the dynamics and use standard resampling techniques. If we wanted a robustly successful control policy

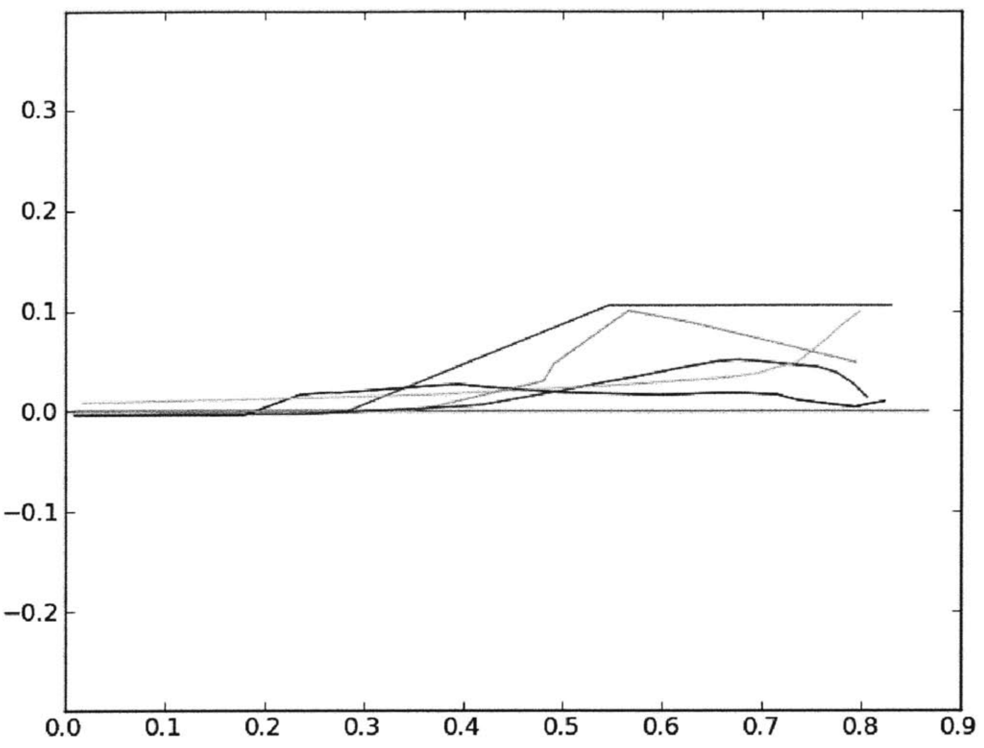


Figure 4-5: Paths for escaping the equilateral triangle: There is little differentiation between these paths in terms of expectation.

with a small number of particles this was an effective strategy. Even though the expected search time differed little, it introduced an appreciable amount of qualitative variation between the trajectories produced by subsequent trials. This was most pronounced for the case of the equilateral triangle in which just about any "reasonable" strategy has comparable expected length. All of the trajectories shown in Figure 4-5 have expected search times of within 3% of one another.

For the purposes of comparison with theoretically derived paths, this is not what we want. More effective in producing repeatable trajectories was to use a very large number of particles and simply not resample at all. For the figures and results in Section 4.2.5 this is the approach we use.

We also explored using a grid-based representation of probability, but found that the necessary fidelity was very high: For problems in dimension greater than three this would likely be prohibitive. The reason for this is as follows. Suppose that δ_t in Equation 4.2.4 is comparably small to the grid spacing. The minimizing heading is able to find a path with favorable round-off errors. When the minimizing distance is short, this can have an appreciable effect. As a result, δ_t must be large compared to this discretization. Since we require δ_t to be small compared to the region, this necessitates a very large number of grid points.

Examine, for example, the trajectory shown in Figure 4-10, which also shows Besicovitch's path for the equilateral triangle. The path is based on a grid with millions of points yet still contains a noticeable discretization artifact. One can verify this by rotating the triangle with respect to the grid axis. Under small rotations, the orientation of the kink moves sympathetically.

A possible solution which we do not explore would be to carefully select headings and distances to optimize over such that these errors are either absent or common to all paths.

4.2.5 Experimental Results

We now compare the trajectory produced by the index policy to the those from the literature for the scenarios described in Sections 4.2.1 and 4.2.2.

Infinite strip

Figure 4-6 and Table 4.1 compare the shape and expected escape time, respectively, for index policy path (blue), Zalgaller's path [94] (green), and the three-segment optimum from [36] (red). Table 4.1 contains the expected search times for these paths. We find the non-smoothed index path to be within 1% of the three-segment optimum.

Half-plane

Figure 4-7 shows the optimum over a parametrized version of Isbell's curve. Figure 4-8 shows the trajectories considered and Table 4.2 gives their expected search times.

The trajectories in Figure 4-8 are aligned to share a worst case. The pink, outer-most trajectory, shown only in Figure 4-8a, is that produced by gradient descent. The red, inner-most trajectory is the min-max solution from [49]. The light blue trajectory, initially similar to the min-max solution is that from [47]. The green and dark blue trajectories are the smoothed and non-smoothed index policy trajectories. The smoothed index trajectory is within 5% of [47]. Although the non-smoothed trajectory is substantially longer, it considerably outperforms the gradient method.

Circle from known distance

We examine the particular case in which the circle has radius $s = 2$ and the initial distance $r = 1$ outside it. The path produced by the index policy is shown in Figure 4-9 both with and without smoothing. The inner-most red path is the min-max path proposed in [46].

Remark 4.2.3. *In the limit of $s \to 0$ all three trajectories are identical and optimal.*

Argument. From [69] it is optimal to trace a circle. □

Equilateral Triangle

Figure 4-10 overlays the index policy path generated by a grid-based representation with Besicovitch's path. We feel that the similarity is quite striking. The Besicovitch path has length ≈ 0.982 while the index path shown has length 0.988, in a triangle with edge length of unity. As it happens, both paths are beaten in expectation by the straight path even though it has length one.

Figure 4-5 also shows a number of paths generated by particle filter implementations of the index policy. Table 4.4 shows the expected escape time for the grid-based index path and the worst-case particle filter path as well as that of Besicovitch's path and the straight path.

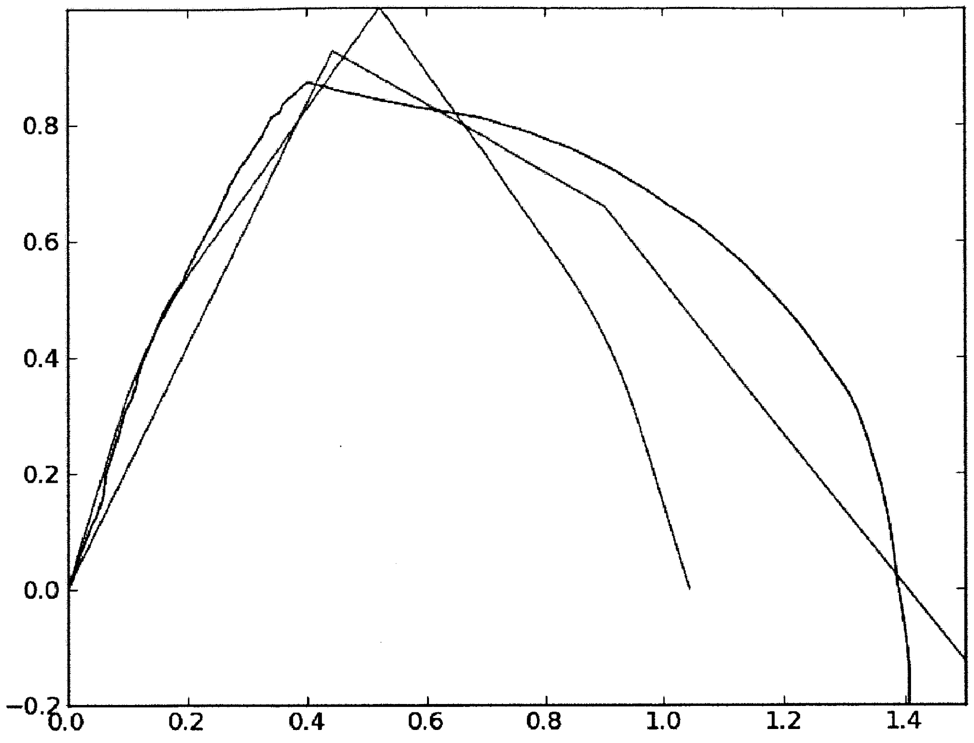


Figure 4-6: Paths for escaping the infinite strip under the index policy (blue), Zalgaller's path [94] (green), and the three segment optimum from [36] (red).

Index Policy	[36] 3-segment	[94] min-max
0.893	0.884	0.917

Table 4.1: Expected escape times from the infinite strip using the trajectories shown in Figure 4-6

[49]	[46]	Index	smoothed	Gradient	$\pi + 1$
3.628 (3.660)	3.469 (3.598)	4.435	3.782	5.876	4.142

Table 4.2: Mean escape time from the half-plane for various trajectories. Our results, based on Monte Carlo, were not consistent with those of [46]. We show ours in parenthesis.

[46]	Index	smoothed	$\pi + 1$
3.853	4.493	4.082	4.142

Table 4.3: Expected search times for the circle from known distance.

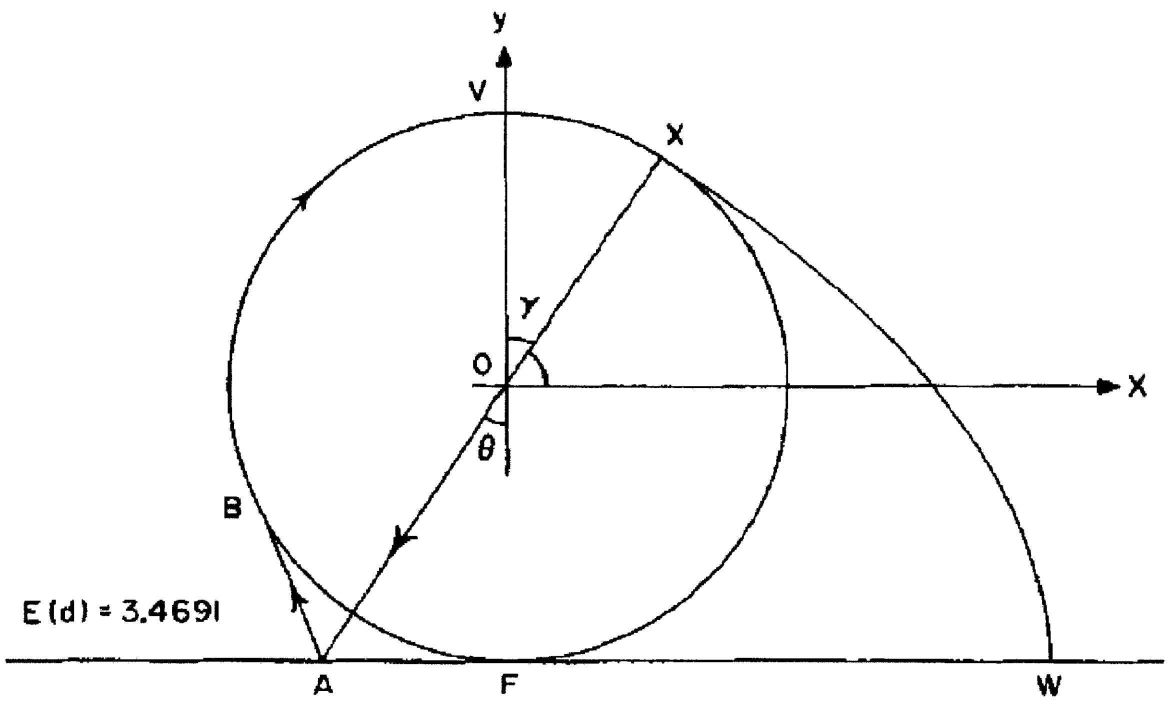


Figure 4-7: Gluss's path for min-mean escape from the half-plane. In [46], the authors optimize independently over θ and quadratic curves XW tangent to the circle and perpendicular to the bounding line.

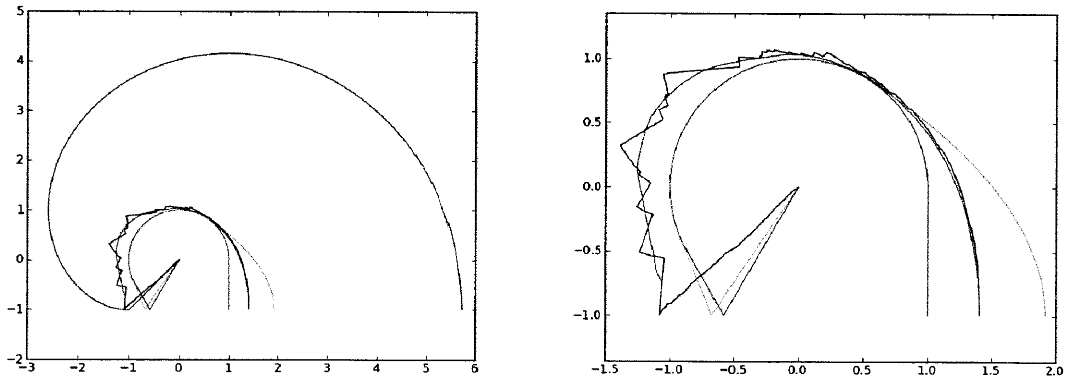


Figure 4-8: The trajectories under consideration for the half-plane. They are aligned such that they share a starting point and a worst case.

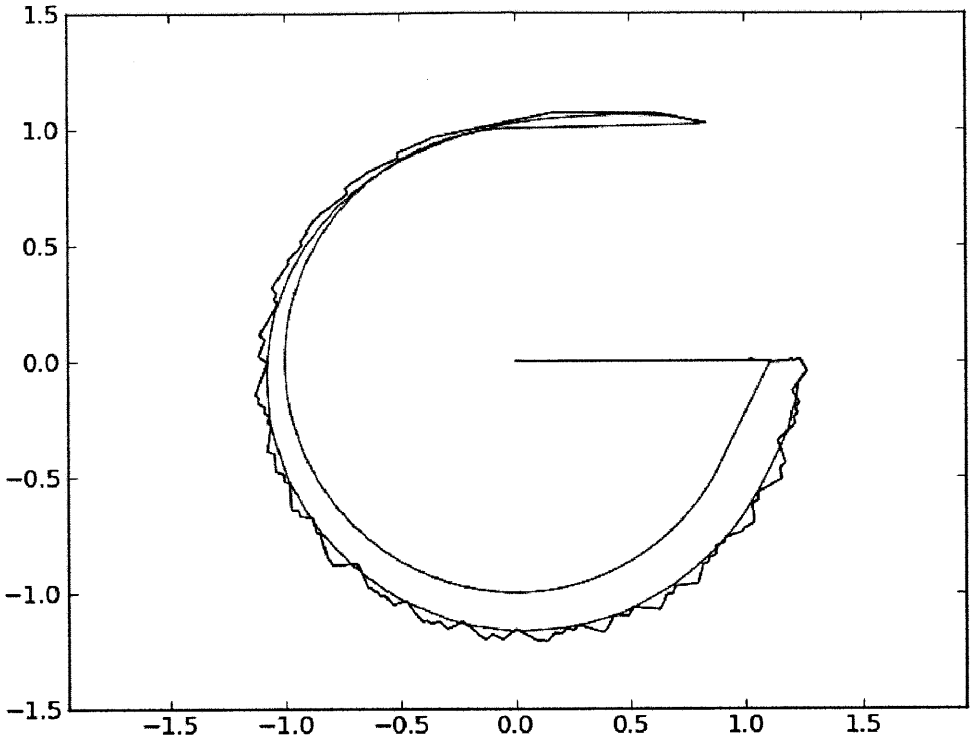


Figure 4-9: Smoothed and non-smoothed trajectories for the index policy for problem of finding a circle of radius 2 from unit distance on its exterior. The inner-most path is the extension the path proposed in [46]

Index Policy (pf)	(grid)	[24]	Straight
0.3200	0.3194	0.3175	0.3129

Table 4.4: Expected escape time from the equilateral triangle with unit edge. The number reported as (pf)is for the worst particle filter path from Figure 4-5, while (grid) refers to the path in Figure 4-10.

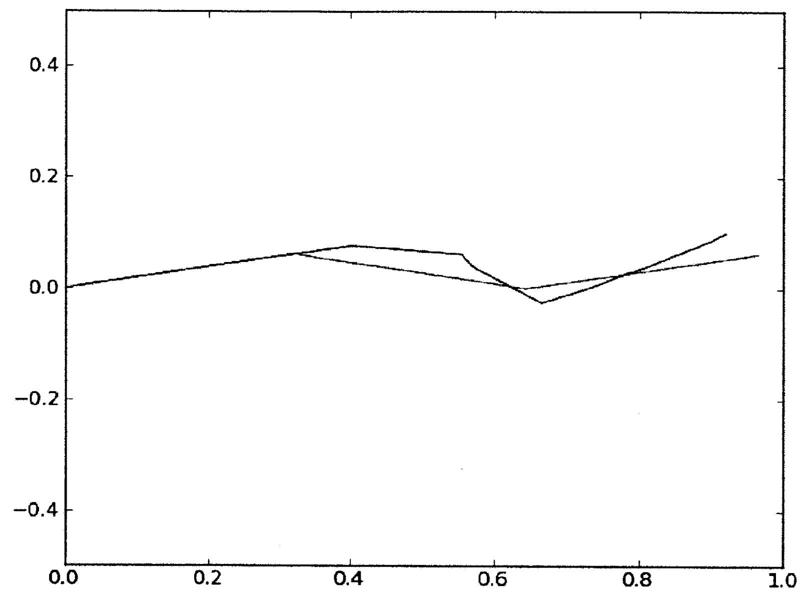


Figure 4-10: The grid-based index path and the Besicovitch path

Other Figures

The index policy goes straight in the 60-degree rhombus and the 60-degree circle section as well as regular polyhedra with four or more sides. It was proven in [91] that the min-max escape path out of any circle section with angle between 60 and 180 degrees is a straight line. As we've already discussed, the min-max escape path out of an equilateral triangle is not straight, although Table 4.4 suggests that the min-mean escape path might be.

4.3 Minimum Weighted Latency Coverage

In contrast to the LFP—in which the goals were known but the agent's state was not—we now return to our original, and more usual formulation in which the agent's location is known but that of the goals are not. For this section we will continue to assume that the agent is a noiseless single-integrator, but now also assume that its initial state is known. We will also now assume that the agent has a perfect sensor of finite radius r_s, *i.e.* it detects a goal if and only if the its location is within r_s of the goal. The problem is to determine a control policy that finds a goal in minimum expected time.

Recall the MWLCP from Section 3.1.2. We generalize this formulation as follows. Define the r_s*-coverage* of a set to be

$$C_{r_s}(X) \equiv \bigcup_{a \in X} B_{r_s}(a). \tag{4.10}$$

wherein $B_{r_s}(a)$ denotes the r_s-ball centered at a. We redefine the indicator function from Equation 3.1 to include the sensor radius.

$$I^1_{\mathcal{P},r_s}(a,t) \equiv \begin{cases} 0 & \text{if } a \in C_{r_s}(\mathcal{P}([0,t])) \\ 1 & \text{o.w} \end{cases} \tag{4.11}$$

The weighted latency of a path of length ℓ is

$$w(\mathcal{P}) = \int_0^\ell t I^1_{\mathcal{P},r_s}(\mathcal{P}(t),t) d\mu(C_{r_s}(\mathcal{P}[0,t])). \tag{4.12}$$

Problem 4.3.1 (Minimum Weighted Latency Covering Problem (MWLCP)). *The Minimum Weighted Latency Covering Problem (MWLCP) is to find a path that is an* r_s*-cover of the region and minimizes Equation 4.12.*

The most significant difference between the MWLCP and the LFP is that in the LFP the boundary was special. Specifically we required that the goal contain the boundary to avoid the particulars of sensing, motion, and inference on the boundary. Now that the agent's state is observable, this is no longer an issue.

4.3.1 Relevant Work

The search problem emerged primarily from the immediate concern posed by submarines in World War II[28]. Early theoretical work was primarily concerned with the optimal allocation of effort, which is how search time would ideally be spatially distributed, ignoring motion constraints (see, *e.g.* [86] and its review [59]).

When trying to determine actual search *trajectories*, this was useful only insofar as it could be used to verify the optimality of intuitive strategies, *i.e.*, sweeps over bounded, planar, Euclidean regions (*e.g.*, [40]), as we demonstrate.

Define the *maximum search rate* as

$$\max_{\mathcal{P}} \max_{t} \frac{d}{dt} I^1_{\mathcal{P},r_s}(\mathcal{P}(t), t)$$

This is the maximum rate at which the agent can see new locations.

Consider the search problem with uniform support over a bounded planar region. Let $A(\mathcal{A})$ denote the *area* of the region. The maximum search rate is $2r_s v$ for an agent with speed v. If there exists a r_s-covering of the region with length $A(\mathcal{A})/(2r_s v)$, then that is surely optimal, since the rate at which goals are found is everywhere maximized. Furthermore, in the limit of $r_s \to 0$ this is the case for the uniform measure in any planar region.

With the exception of the results for search on the real line, based on first-order optimality and discussed in Section 2.2.6, it is by this type of argument that optimality has been established for search problems. For instance, under sufficient restrictions on μ, one can always construct sweeps that are asymptotically optimal in the limit of $r_s \to 0$[40]. The result [3], discussed in Section 2.2.6, establishing the optimality of minimum length covers on weakly Eulerian graphs is another example of such an argument.

However, this approach can only be applied in a very limited set of circumstances, specifically, when there exist paths that exactly achieve the theoretical bounds *and* someone had the ingenuity to construct them.

Coverage

A well-studied and closely related problem is commonly referred to as the coverage problem, which is that of finding a minimum length, as opposed to minimum latency, r_s-cover. This research has been focused on bounded, but non-convex subsets of the Euclidean plane. The survey [27] taxonomizes the available algorithms into two classes:

- randomized and behavior based, and
- those based on cellular decomposition.

The former class of algorithms are metaphorically based on simple strategies seen in nature. The research is devoted to finding conditions under which the region is almost surely eventually covered. These algorithms are generally advocated for the case in which the geometry of the region isn't known in advance. Such an approach is a natural one for an inexpensive floor-sweeping robot, for instance.

The latter class involves dividing the region into sub-regions that can be covered by simple sweeps and then ensuring that the agent or agents sweep each sub-region. These algorithms are advocated for situations in which

- such policies are known, *a priori*, by the above arguments, to be optimal, or
- when thoroughness is more important than speed, like mine sweeping, for example.

4.3.2 Search in Length Spaces

Having defined weighted latency in Equation 4.12, we will use our usual subsidized stopping problem as a means of comparing trajectories. Let $\mathcal{P}$ be a closed path starting from location x. The index or *price* of this path is given by

$$\gamma^*(\mathcal{P}) = \frac{w(\mathcal{P}) + \mathcal{L}(\mathcal{P})(1 - \mu(C_{r_s}(\mathcal{P})))}{\mu(C_{r_s}(\mathcal{P}))}. \tag{4.13}$$

The index policy is simply to pursue the best path with respect to this metric.

Since the space of all paths is infinite-dimensional, we need to restrict the set of paths considered to a smaller set of candidate paths. For this to be an effective strategy we need the set of candidates to be

- searchable, and

- to include good paths.

In Riemannian Manifolds, we considered the straight paths. In length spaces we do not have a meaningful of a notion of "straight," and so we must find another way.

If the region is a finite-dimensional length space, we suppose that a reasonable such set is a minimal set of shortest paths from the agent's location. Let $\tilde{\mathcal{Q}}(x)$ denote the set of shortest paths starting from x. To avoid distracting non-existence issues we allow these paths to use the closure of the region. The set of interest is

$$\overline{\mathcal{Q}}(x) \equiv \{\mathcal{P} \in \tilde{\mathcal{Q}}(x) \text{ s.t. } \nexists \mathcal{P}' \in \tilde{\mathcal{Q}}(x) \text{ s.t. } \mathcal{P} \subset \mathcal{P}'\}.$$

The dimensionality of $\overline{\mathcal{Q}}$ is still potentially larger than that of the region, for example in $[0,1]^2$ using the L_1 metric. We can further reduce the size of this set, by including only one shortest path for each endpoint. Let $<$ be an arbitrary ordering on paths. Let $\mathcal{Q}_x(a)$ denote the set of shortest paths from x to a, $\{\mathcal{P} \in \overline{\mathcal{Q}}(x) \text{ s.t. } \mathcal{P}(d(x,a)) = a\}$.

$$\overline{\mathcal{Q}}'(x) \equiv \bigcup_{a \in \mathcal{A} \text{ s.t. } \mathcal{Q}_x(a) \neq \emptyset} \min_{<}\{\mathcal{Q}_x(a)\}.$$

The set of paths $\overline{\mathcal{Q}}'$ has, at most, the same dimensionality as $\mathcal{A}$. Particularly if Equation 4.13 varies smoothly throughout $\overline{\mathcal{Q}}'$, searching this set is not prohibitively difficult, provided that the dimensionality of the region is not very large.

If the region is a Riemannian Manifold, there is a potentially significant difference between searching over heading, as described in Section 4.2.3, and searching over the set of longest shortest paths. This is illustrated by helical path shown in Figure 4-11 which can be found by searching over headings, but is not in the set $\overline{\mathcal{Q}}'$. In such a case it is surely preferable to search over headings since these will contain the shortest paths as subpaths.

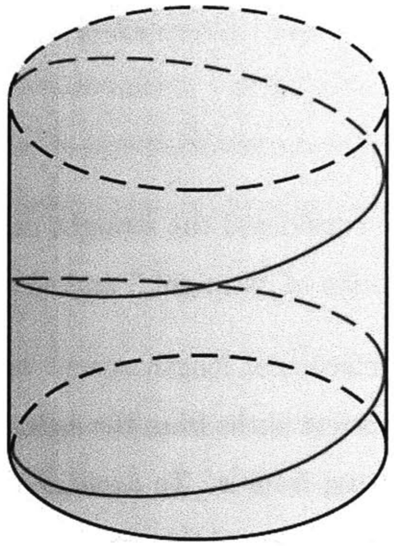


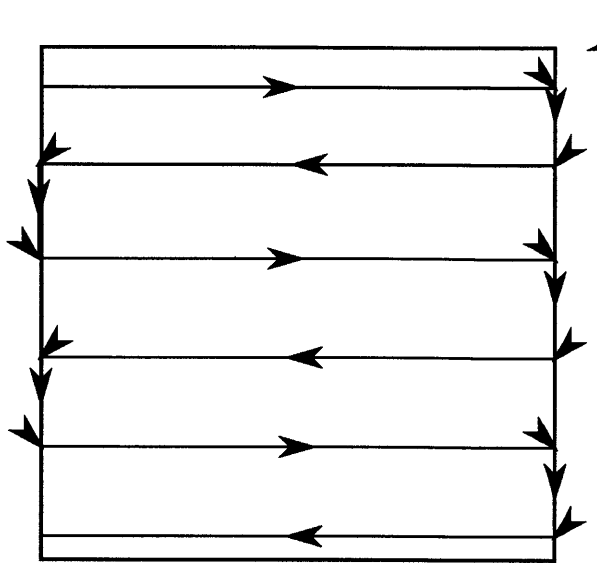
Figure 4-11: The helical path is not a shortest path between any pair of points.

4.3.3 Comparison to Sweeping Policies

There is only a small set of problems for which an optimum is available for comparison, which are described in Section 4.3.1. Since we are not especially concerned with the zero sensor radius limit, (and we have already discussed trees and Eulerian graphs) this set includes only uniform density over Euclidean spaces that can be nearly or exactly covered by a path of width $2r_s$. Theoretical arguments aside, people have been mowing fields for long enough to have established a solid consensus for such cases.

For cases in which such a path is not straight, the index policy is likely to be sub-optimal. The reason for this is that the second term in Equation 4.4 biases the choice towards longer, more successful, straight paths. If $\mathcal{A} = [0,1]^2$ and we use the Euclidean metric, the agent will initially move in the direction of the most distant corner. Which seems obviously wrong...

Although a sweeping seem obvious in this case, it should be remarked that $[0,1]^2$ *cannot* be perfectly covered by such a path for any $r_s > 0$ as is shown in by the red sub-region in Figure 4-12b. The simple sweeping path moves all the way to the boundary, as shown in Figure 4-12a, will actually perform less well than the index policy in some instances with large enough sensor radii.


(a) The simple sweeping strategy

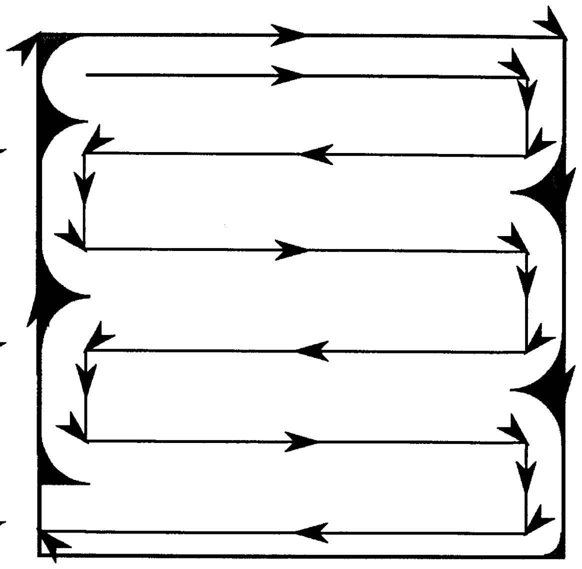

(b) A smarter sweeping strategy

Figure 4-12: Sweeping strategies

To explain this counter-intuitive result consider the last small ε before the boundary. In this motion the agent sees an new area of only $\varepsilon^3/(2r_s)$, but delays the rest of the region by ε. As in the case of the triangular distribution on the real line (Section 2.2.6), exhausting the direction is a losing proposition.

The sweep shown in Figure 4-12b is addresses this and is always better than the index policy. Figure 4-13 compares the performance of the index policy and these two sweep policies while varying the sensor radius.

4.4 Control Problems

This section will briefly discuss the possibility of extending the index policy to problems involving control of dynamical systems that are not well-described by deterministic single-integrators. Section 4.4.1 discusses the possibility of more complex dynamics. Section 4.4.2 will discuss the ramifications of introducing noise into the dynamics of the system.

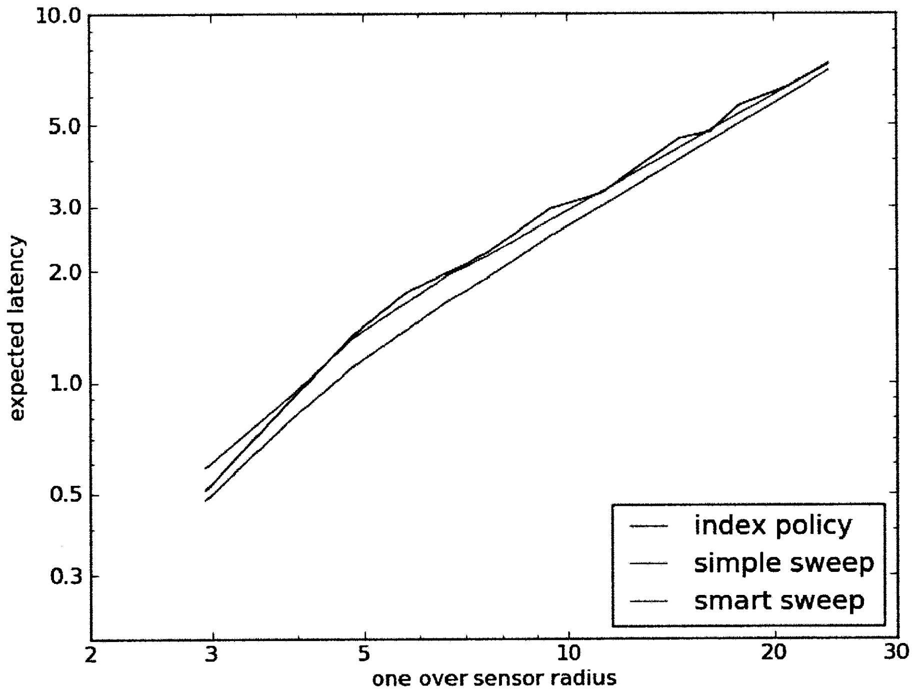


Figure 4-13: Search times as a function of sensor radius for various policies for searching the uniform distribution on the unit square. The starting points were chosen to be ideal for the sweep policies: r_s from the corner.

4.4.1 Complex Dynamics

We argue that, as a practical matter, the single integrator model is typically sufficient. The reason for this is that for typical mobile agents in search applications, the *path planning* problem of "how to *find* the goal" is not of the same scale as the control problem.

Suppose the agent is a person on a unicycle with a chair on his chin: If the goal is more than a few meters away, his dynamics and motion planning can be treated independently. In geographic search, *i.e.*, involving physical robots in physical spaces, we posit that this is typically the case: the regions (and the variation within them) are large compared compared to the dynamical limitations of the agents.

Nonetheless, it is not impossible to incorporate other dynamical models into the search policy we have developed. Notice that the index function is simply a metric by which we can compare closed paths through the search space. This metric, coupled with a searchable parametrization of the set of closed paths, constitutes a control policy. Although finding a good such parametrization may be quite difficult for an arbitrary dynamic model, it is not necessarily difficult for the types of models one typically encounters.

For example, search with a fixed-wing aircraft is a problem of significant practical importance. Typically these are modelled as Dubins vehicles which move at constant speed and are subject to a bounded turning rate. For such a vehicle, we propose that an adequate parametrization would be to the set of "running tracks" consisting of two parallel lines joined by half-circles of a fixed radius, shown in Figure 4-14. In particular, we propose that the radius of these half-circles be that of the sensor or turning radius, whichever is larger.

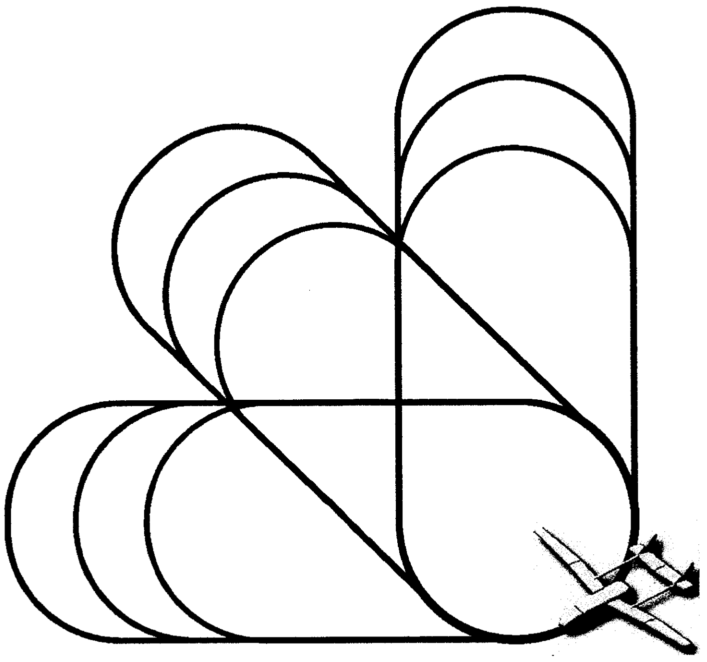


Figure 4-14: Proposed paths for the Dubins vehicle

4.4.2 Noisy Dynamics

Let us examine problems of stochastic control of the following form. Let the agent be a bounded-input, single integrator subject to noise w with a perfect binary sensor.

$$\begin{aligned} \dot{x} &= u + w(x), ||u|| < 1 \\ y &= \begin{cases} 1 & \text{if } x \in G \\ 0 & \text{o.w.} \end{cases} \end{aligned}$$

wherein w is an arbitrary random process. Suppose that our loss function is

$$\mathcal{L}(t) = I(x(t) \notin G)$$

We are interested in finding the optimal control. In an unbounded space of dimension two or higher, we will likely have to restrict our attention to discounted reward to ensure the existence of optimal policies. That is, it might be the case that there does not exist a policy that can spend a non-vanishing fraction of its time in the goal state. For a bounded space (with appropriately defined dynamics at the boundary) there exist policies for which the time-average reward is finite (module other necessary conditions, of course, such as adequate control authority and a measureable goal).

Whenever the agent is outside of the goal, the index policy can be used to drive the agent back into it. We see this as another example of the index policy's general versatility, but for this general problem there are a number of important sources of sub-optimality.

We have already discussed the issue of the dynamics on the boundary. Additionally, it is possible that the optimal policy does not try to move directly to the goal in minimum time at all. That is, it could be the case that a longer path might make it easier to *stay* in the goal once it is reached.

Deciding whether this is the case is a value-of-information problem for which the only tool available, for all practical purposes, is ADP. While there certainly *could* exist a description of the agent location distribution that makes ADP both tractable and effective, there does not exist a general algorithm for finding it. Since the goal of this dissertation is to present a general purpose policy, we can not consider this possibility, instead we assume that whenever $y = 0$ the agent tries to reach the goal as quickly as possible.

Furthermore, since the noise is arbitrary and state dependent, the optimal open-loop trajectory may be arbitrarily complex. This can be shown by defining a "noise maze" in which motion is deterministic until the agent touches a wall, which introduces large errors. If these errors are large enough we can be assured that the optimal policy is to "solve" the maze. Since we have committed *a priori* to search over a certain subset of trajectories, we have no guarantee that this set include such a solution.

The noise need not be complex to demonstrate this. Consider a two wheeled robot with much greater directional accuracy than angular accuracy. Also consider a case in which the direction of the noise is known and constant, but whose magnitude varies. For both of these cases, the optimal open-loop trajectory may be to aimoff. If we restrict the considered trajectories as described in Section 4.2.3 and Section 4.3.2, we will not find these trajectories. Section 6.2 suggests a possible extension of this dissertation to approach these issues.

Chapter 5

Dynamic Search and Routing Problems

This chapter will discuss the dynamic versions of some of the search problems we have covered thus far as well as introduce the Dynamic Search and Repair Problem (DSRP) in which agents must not only find goals but spend time at them.

As a motivating example we consider a scenario in which a fleet of mobile surveillance assets are responsible for detecting and responding to possible improvised explosive devices (IEDs) along a network of roads. We suppose that the system will be externally notified of some requests, *e.g.*, someone may report a suspicious object, but not necessarily all requests. The agents are also responsible for patrolling the region and detecting IEDs themselves.

5.1 Introduction

In a dynamic search problem, goals, which we will now refer to as "requests," arrive in the region according to a space-time random process. We will assume that the region is a length space and that the arrival process be separable. Specifically this means that it can be described with two components, a probability mass function $\Lambda(t)$ over the number of requests by time t, and a spatial probability measure Φ from which the locations of these requests are drawn independent, identically-distributed (i.i.d.).

The probability of there having been exactly n requests in subset S at or before time t is given by

$$\Pr(n \in S) = \sum_{i=n}^{\infty} \Lambda(t,i) \binom{n}{i} \Phi(S)^n (1 - \Phi(S))^{(i-n)} \tag{5.1}$$

The previously considered problems can be considered instances of dynamic problems in which $\Lambda(t) = \Lambda(0)\ \forall t$.

The system objective remains minimizing the expected "latency" of these requests, which is now defined as the amount of time between a request's arrival and when it has been "serviced." We will consider two different cases for what constitutes *service.*

- A request is serviced when it is detected.
- A request is serviced when an agent has moved to its location and spent a random amount of time s doing on-site tasks.

We refer to the former of these formulations is known as the PPP as proposed by [34]. This is exactly the dynamic version of the minimum weighted latency sensor cover problem discussed in Section 4.3. We will refer to the latter as the DSRP as which extends the DTRP proposed by [21] to problems involving search.

The majority of this chapter will be devoted to the DSRP. The two features that differentiate the DSRP from the PPP are that, in the DSRP, in order to service a request

1. the agent must move to the location of a request, and
2. the agent must spend a random amount of time providing on-site service.

The on-site service times are drawn i.i.d. according to distribution s. We will assume that service is non-preemptive meaning that once an agent starts providing service it will continue to do so until until that service is completed.

Definition 5.1.1. *Random variable X has non-increasing conditional mean if*

$$\mathbb{E}[X | X > x] - x$$

is non-increasing in x.

Lemma 5.1.2. *If s has non-increasing conditional mean then there is an optimal policy that does not preempt service.*

Proof. Assume some optimal policy, Π, preempts a request r time t_0 after starting; it then does a sequence of actions K of length t, servicing n other requests, before returning to r. We construct a new policy Π' identical, except before pursuing K it completes the request r. If we assume that the on-site service time is cumulative, *i.e.*, the agent can resume where it left off, Π' reaches the location of r at the exact moment that Π finishes it. This allows Π' to resume emulation. If we do not make this assumption, Π' can wait for Π to catch up.

With this new policy, the latency of r is decreased by t, and that of the n other requests is increased by $\mathbb{E}[s|s > t_0] - t_0 \leq \overline{s}$, by assumption. The expectation of t is $n\overline{s}$, wherein l denotes the length of the tour visiting all the requests Therefore the expected latency under Π' is no greater for these $n + 1$ requests than under Π. Since the policies are otherwise identical, this imples that Π' is also optimal. □

In Section 5.7.3 we will discuss the case in which service times lack this property as well as cases in which there are multiple *classes* of requests to which we might give preferential treatment. In these cases, preemption is an important consideration.

5.1.1 Assumptions

We will assume that each agent has a perfect sensor with finite range r_s, as we did in Section 4.3. In addition to this sensor, in the case of the DSRP we will also consider the possibility of the external detection of requests. We will assume that a request at location a, is immediately detected upon arrival with probability $\rho_{\text{ext}}(a)$, and with probability $1 - \rho_{\text{ext}}(a)$ it will never be externally detected, rather it will only be detected when an agent comes within r_s of it. This allows us to model immobile sensors or the fact that some hikers carry cellular phones with spatially varying access to a network.

We assume that an agent maintains a belief state at time t about the realization of the process that can be encapsulated as the triple $\mathcal{I}(t) \equiv \langle \theta(t), Z(t), \mu(t) \rangle$

1. a list $\theta(t)$ of detected requests,

2. a probability mass function $Z(t)$ over the number of undetected outstanding requests, and

3. a spatial probability measure $\mu(t) : \mathcal{A}^* \to [0, 1]$ measuring the probability that a randomly selected outstanding request is in a particular subset of the region.

5.1.2 Policies

The range of a policy is simply a control—what each agent should do for the next infinitesimal amount of time. The index policy is a "re-optimization policy" meaning that at each instant, we take a snapshot of the state and solve a static problem. The solution to the static problem is *not* a policy but a *plan*, *i.e.*, a deterministic sequence of actions of potentially non-vanishing length. The *policy* will then only execute the first action of the plan before re-planning. The traditional receding horizon planner is another example of such a policy.

It should be noted that in this framework, the plan is not allowed to be contingent on future observations. Unlike in the static search problems in previous chapters, this a major limitation in dynamic problems. The planner is unable to explicitly calculate the value of information. In a general POMDP this would be a large weakness, but in our problem somewhat less so. The reason being

- when an agent observes a location, that observations implies little about other locations and

- if the sensor radius is small compared to the region, information gathering and information exploitation are tightly coupled.

5.1.3 Organization

Section 5.2 reviews the relevant research threads. As an appetizer to the DTRP, Section 5.3 we will discuss the PPP which we address using the algorithms presented in Chapter 4 without modification. Section 5.4 transitions from the PPP to the DTRP, considering the DTRP in which the on-site service requirement is zero. Sections 5.5 and 5.6 extend much of the existing theory for the observable DTRP to length spaces. In Section 5.7 we extend the formalism of the DTRP to search, extend the index policy to address it. In Section 5.8 we describe a scenario of interest that we believe underscores the contribution of this dissertation.

5.2 Review of Existing Work

This section attempts to sketch the broad historical context of several research communities that are relevant to the topic of Dynamic Vehicle Routing (DVR). To that end, we focus on surveys and milestone works.

5.2.1 Spatial Queueing

The beginning of Dynamic Vehicle Routing (DVR) can be traced to the early 1970s when researchers tried to apply queuing theory to emergency service systems. As a result, these researchers used what we term a "service-oriented approach" in which they attempted to minimize the latency of these systems. In this framework, queueing theory and geometric probability form the basis of analysis.

This early period is well-summarized by [64]. Such efforts were indisputably successful and well-received. Major cities around the world subjected their police, fire, and ambulance systems to analytical scrutiny; money was saved and services improved.

However, in order to apply the arguments from queueing theory and geometric probability, certain trivializing assumptions needed to be made. For instance, either the on-site service requirement or the travel time was considered relevant to analysis, but not both. Typically the system could become "full" at which point new requests are completely (and permanently) ignored.

While these problems are fundamentally stochastic, the types of questions that could be successfully addressed were static in nature. Primarily this meant answering staffing questions: "How many operators, ambulances, *et c.*, are needed to provide some minimum quality of service?" The dynamic questions, *i.e.*, how agents should react to real-time information, were left to the human decision makers.

5.2.2 Dynamics

The proliferation of information technology *circa* 1990 put a great deal of pressure on the Operations Research community to handle the dynamics of information. Notwithstanding the Internet, a number of important enabling technologies created this emphasis including the GPS, highway monitoring systems, and improved mobile communications infrastructure. Simultaneously, just-in-time production and e-commerce were emerging as important business models. This precipitated what could be called a "Operations Research Renaissance" during which Vehicle Routing Problems (VRPs) gained substantial commercial import.

For many problems the static formulation is obviously unsatisfactory. This recognition led to a surfeit of new problem formulations as researchers tried to tack dynamics onto well-understood problems. Early attempts like [53, 50, 19] were offline algorithms designed to handle *a priori* uncertainty and did not adapt to real-time information. Dynamics were handled with queuing theory and spatial problems were solved statically.

It wasn't until [79] that the interplay was recognized. This paper introduced the modern notion of Dynamic Vehicle Routing (DVR) and the Dynamic Traveling Salesman Problem (DTSP) as an instance of it. In the Dynamic Traveling Salesman Problem (DTSP) the region was a graph and the arrival process was temporally Poisson supported by the nodes. Subsequently, [21] extended the problem with the Dynamic Traveling Repairperson Problem (DTRP) in which the region was a compact convex subset of the Euclidean plane. Initially they considered only Poisson processes as well, but in a later paper [23] they extended their analysis to separable processes with Lipschitz continuity in the spacial component.

This capstone works of this research thrust came shortly after with [22, 23, 93, 72]. These provided

- asymptotically optimal results in the light-load limit as well as
- constant-factor optimal results in the heavy-load limit,

for requests arriving on

- a convex subset of the plane with seperable, Lipschitz continuous demand or
- the nodes of a graph.

The surveys [80] and [20] discuss the crossroads at which the vehicle routing community subsequently found itself. They were frank in admitting only a few first steps had been made and recognizing the need for real-time algorithms that exploited the dynamic information. Having seemingly exhausted the available theoretical tools, however, this research thrust largely dried up.

5.2.3 Combinatorial Optimization

Meanwhile, work on many of the other VRP variants continued apace and are taxonomized by [41, 62].

Coming from the VRP (and before that the TSP), as opposed to queueing theory, these researchers brought a "cost-oriented approach," *i.e.*, one concerned with finding policies in which the agent accrues the minimum cost. This cost is typically the total time taken or distance covered. Since these algorithms were originally developed for non-dynamic problems, the algorithms strengthen dramatically if more information is available up front. For instance

- if some fraction of the requests' locations and arrival times are known initially (with that fraction referred to as the "degree of dynamism" [66, 63]), or

- if we are given some temporally advanced notice of them [51, 2].

To characterize the quality of these approaches, the most widely accepted theoretical tool is competitive analysis, as defined in Section 2.1.3. This framework was introduced by [85] and popularized by [7]. A thorough survey of its use in DVR can be found in [52].

5.2.4 Scheduling

A closely related branch of queueing theory examines Polling Systems of which [88] is an excellent survey. In a polling system a server moves on a network between a finite set of queues to provide some service. Traditionally, the goal has been to find an optimal static polling schedule that visits all the queues.

In the case of Continuous Polling Systems (CPSs) (see, *e.g.*, [4, 33]), there may be uncountably-many queues distributed over a continuum of the form considered in Chapter 3, typically in a single loop. As with ordinary polling systems, the work has focused on finding optimal static policies. Dynamic results are restricted to the heavy-load limit, in which the optimal policy can be shown to converge to the solution to the static problem[32]. Notably, [70] considers a simple dynamic polling problem and also approaches it from the perspective of Whittle Indexability.

5.2.5 Multi-Agent Control

More recently, mounting excitement about cheap, capable robots, has spurred new in interest spatial queueing. With [39], the control community started to talk about robots providing a "service layer" and rediscovered the DTRP.

A number of important extensions have been made adding realism to the problem, a good summary of which can be found in [77, 34]. Much of the work has focused on the control aspect of the problem and has worked to incorporate realistic dynamics, not only of the vehicles but also of human operators. Particularly relevant extensions include sensing constraints in [34] and the addition of deadlines in [74].

A related area of research is focused on coverage problems. This includes the work described Section 4.3.1 but more recently has been extended to refer to problems involving the efficient deployment of a fleet of mobile sensors, as described in [30]. In this extension, the solution space is, likewise, the set of control policies, but the system objective is to perform a "locational optimization," the solution to which are fixed locations to which the agents should converge. We discuss such optimization in some length in Section 5.5.

5.2.6 The Intermediate-Load Regime

All the work on DVR described in Sections 5.2.2 and 5.2.5 focuses on asymptotic optimality and leaves open the intermediate-load regime in which the load factor (*i.e.*, the fraction of the time in which the agents are busy doing something) is neither very low nor very high. This is problematic because

- in the light-load limit, agents are severely under-utilized, and
- as one approaches the heavy-load limit optimal performance quickly becomes unacceptably poor.

The implication is that for many practical cases, existing work does not provide a definitive answer.

Far be it to say that existing work is not applicable to the intermediate-load regime, however. Algorithms have been presented (*e.g.*, in [72] and more recently in [74]) that are both optimal in the light-load limit and within a constant factor of optimal in the heavy-load limit. It is certainly plausible that such algorithms perform well in the intermediate-load—in fact, [72] explicitly conjectures as much.

5.3 The Persistent Patrol Problem

In the Persistent Patrol Problem (PPP), all that is required of the agents is that they "detect" requests by coming within a minimum distance, which we refer to as the sensor radius, r_s. The goal is to determine a routing policy that minimizes the expected amount of time a randomly selected request goes undetected. We will refer to problems with this character as PPPs using the formalism presented in [48].

The literature on the PPP is largely contained in the thesis [34] and it's recent extension, [48], who restricted their attention to smooth measures on convex subsets of the Euclidean plane. These works propose a number of algorithms for the PPP problem and analyze them in the heavy-load (*i.e.*, $d\Lambda/dt \to \infty$) and zero-sensor-radius limits. Among other contributions, [34] extends the search results discussed in Section 4.3.1 to the case of dynamic arrivals. Interestingly, the ideal sweep rate is proportional to the *square root* of the arrival density, due to random incidence.

As with the non-dynamic search problem, the theoretical results do not address the intermediate cases involving non-negligible sensor radii. Among the algorithms presented, only one, based on ADP, remains theoretically sound without asymptotics.

5.3.1 Index Policy

The policy we here propose is unchanged from that of the MWLCP discussed in Section 4.3.

Let $\mu(t)$ denote the spatial component of the belief at time t. Let x denote the agent's location and let $\mathcal{P}$ be a closed path starting and ending at x. And let $\Phi_{\mathcal{P}}(\tau)$ measure the probability that a single point selected according to $\mu(t)$ is detected by an agent pursuing $\mathcal{P}$ from x a distance τ.

The index function is

$$\gamma^*(\mathcal{P}) \equiv \inf_{t>0} \frac{\int_0^t \tau d\Phi_{\mathcal{P}}(\tau) + 2(1-\Phi_{\mathcal{P}}(t))t}{\Phi_{\mathcal{P}}(t)}. \tag{5.2}$$

The index policy is to pursue the path minimizing this function. Refer to Section 4.3.2 for parametrizations the set of paths to make this minimization feasible.

5.3.2 Comparison to Optimal Sweeps

As was the case in the MWLCP, in Section 4.3.3 the most, stringent comparison we can make is with a sweeping policy for the uniform distribution in the Euclidean unit

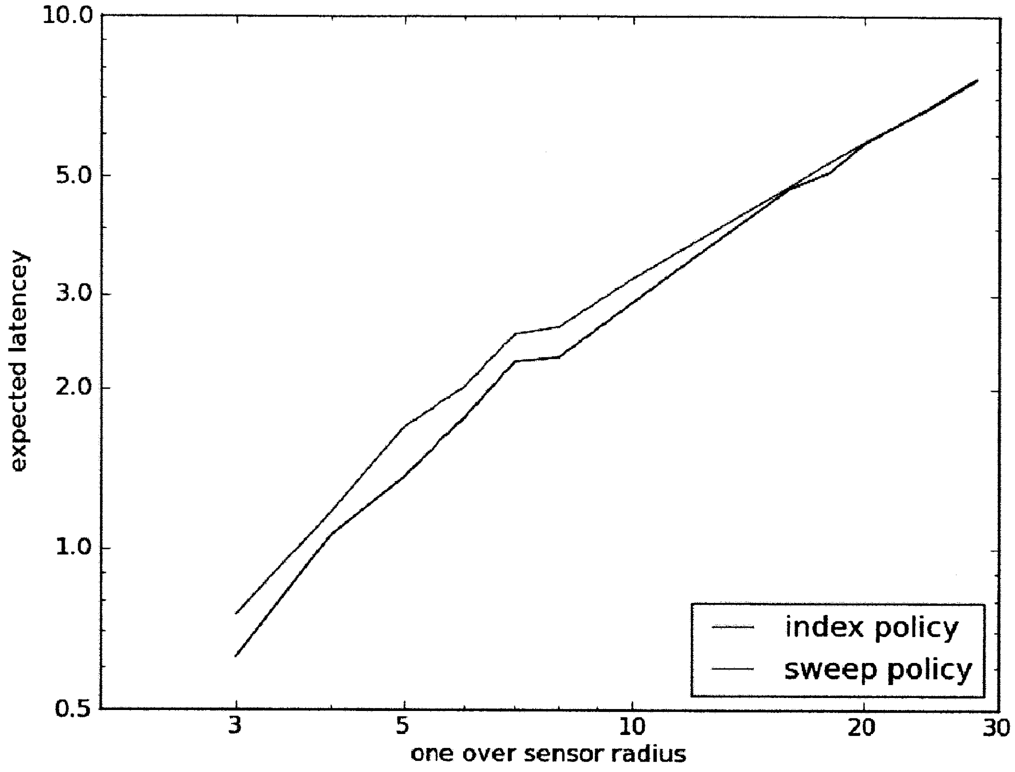


Figure 5-1: Comparison of index and sweep policies in the unit square.

square. In the limit of small sensor radius such a sweep is optimal. Figure 5-1 shows this comparison for range of sensor radii.

The latency values are very similar to those for the MWLCP in the same space. In fact, the latency of the sweep policies is the same as expectation over starting location of that for the MWLCP. On the other hand, the index policy is able consistently do better comparatively. The reason for this being that the index policy does not visit the corners and edges as often as the sweep policy. Figure 5-2 shows an initial portion of the index policy's trajectory for a sensor radius of 0.2.

We do not include the so-called "smart sweep" policy from Section 4.3.3 in Figure 5-1, because it is no longer smart at all. It was able to improve the sweep by leaving the corners and some areas near the edges until the end. Now that there is no "end," this policy spends a disproportionate amount of time in these areas of low search efficiency.

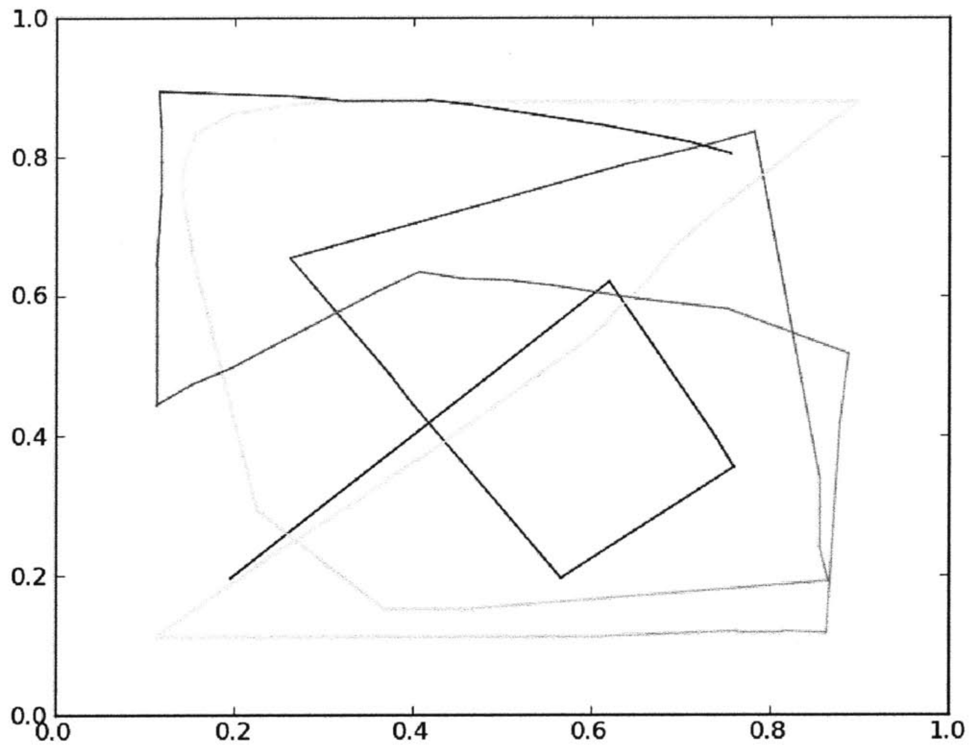


Figure 5-2: The initial portion of the index policy trajectory for $r_s = 0.2$. Blue indicates older locations while red indicates more recent ones.

5.4 An Intermediate Problem

This section considers the problem in which an agent must move to the location of a request, but the on-site service requirement is zero. A new difficulty this raises is that the agent now *knows* about requests that have not been dealt with yet. We address this in the same fashion as in Section 3.3: by folding this information into the spatial measure. Let $\langle\theta(t), Z(t), \mu(t)\rangle$ denote the belief at time t as defined in Section 5.1. We construct a new spatial measure $\hat{\mu}(t)$ by

1. multiplying $\mu(t)$ by $\mathbb{E}[Z(t)]$,
2. adding unit atoms the location of each element in $\theta(t)$, and
3. re-normalizing

When computing the index function we now use $\hat{\mu}$ instead of μ.

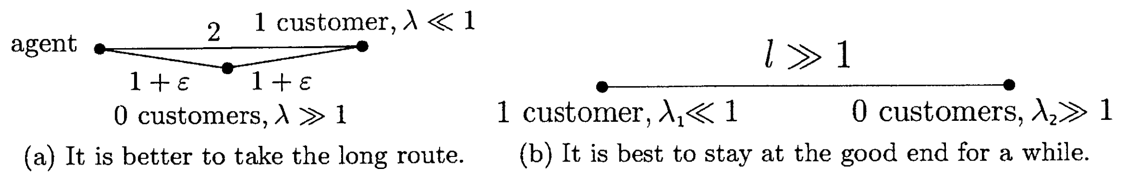
5.4.1 Proposed Augmentations

The index policy is a "snapshot" planning algorithm. At each moment it solves a static problem using the belief at that moment. In a dynamic environment, this does not use all of the information available, in particular, information about the arrival process. The following examples will demonstrate some of the problems this presents and suggest modifications to solve them.

Forward Propagation of Belief

Consider the case shown in Figure 5-3a and assume the arrival process is Poisson with rate λ. Ignore for the moment that it would be impossible to reach this configuration unless it was the initial condition; for the PPP this is an unusual state, but for the DSRP it will not be. We would probably prefer the agent to take the slightly longer path that services the customers who are likely to arrive at point p_2 while the agent is in transit.

To elicit this behavior, we propose the following. Rather than compute the index function, Equation 5.2, using $\Phi_{\mathcal{P}}(t)$, use $\Phi'_{\mathcal{P}}(t,\tau)$ based on the posterior belief



agent
2
1 customer, $\lambda \ll 1$
$1+\varepsilon$
$1+\varepsilon$
0 customers, $\lambda \gg 1$
(a) It is better to take the long route.
$l \gg 1$
1 customer, $\lambda_1 \ll 1$
0 customers, $\lambda_2 \gg 1$
(b) It is best to stay at the good end for a while.

Figure 5-3: Problematic Examples

$\mathcal{I}(t+\tau|\mathcal{P}([0,\tau]))$. That is, forward propagate the arrival process and conditioning on following $\mathcal{P}$ a distance τ *and not observing any requests.*

Loitering

Consider the case shown in Figure 5-3b, in which the agent has a small, non-zero sensor radius, r_s. Suppose that the agent last visited the left queue one time unit ago and has just depleted the right queue. The subsidy required to move to the right is infinite, and so the agent will move left. Assume that it goes a distance of r_s before a new target arrives at the right queue.

After moving a distance $r_s + \delta$ the agent will see the following subsidies.

$$\begin{aligned} \text{To the right:} \quad & \frac{(2-\delta\lambda_2)(r_s+\delta)}{\delta\lambda_2}, \\ \text{and to the left:} \quad & \frac{(2-(1+\delta)\lambda_1)(1-r_s-\delta)}{(\delta+1)\lambda_1}. \end{aligned}$$

For sufficiently large λ_2/λ_1, the agent will change its mind and goes back. Any requests that arrived during this out-and-back will be needlessly delayed—a better policy would have been to stay at the right endpoint.

To address this we propose computing the index function on paths that move to some location and wait there forever. With respect to a snapshot, that would not have been finite, but with the above augmentation, it is well-defined. For the example, the subsidy associated with staying at p_2 would be $1/\lambda_2$ which results in the desired behavior. Furthermore this allows an agent with a large sensor radius to move to the centroid of the region, which we will show to be optimal in the light-load limit in Section 5.5.

In the results subsequently discussed, we implement only a limited version of these augmentations for two reasons. First, computing posterior distributions is already complicated—this *projected* conditional measure is even more complex. For simple processes, specifically Poisson point processes, it can be simply done. But for general temporal processes, particularly in the DSRP with the presence of external sensing, it cannot. But for Poisson point processes it solves a largely non-existent problem. The second reason is arguably aesthetic: the integral of Φ' no longer has a natural probabilistic interpretation. Since the edifice of this dissertation relies heavily on such interpretations, we are loathe to sacrifice them.

The augmentation that we implement is to consider only one such infinite path and that is the one that does not move at all. For all other paths we continue to use the snapshot distribution.

5.4.2 Results

This section describes a simple experiment in which the region consists of a single segment with length l on which there is a uniform, Poisson arrival rate λ. Let m denote the number of agents. The AUCTIONINGNETWORKINDEX algorithm was used to generate multi-agent plans.

Consider the case in which the agent can sense the entire region. In the limit of $\lambda \to \infty$, any optimal policy achieves an expected waiting time of $2l/3m$. In the light-load limit, any optimal policy achieves an expected waiting time of $l/4m$. Figure 5-4, which considers only a single agent, shows the transition. The "interesting" arrival rates encompass more than an order of magnitude. That is, there is a wide swath of operating conditions in which the asymptotic cases are not indicative of performance.

With zero sensor radius, the optimal policy is to sweep, which has the same expected waiting time as the high-rate limit. Figure 5-5 shows the progression from light-load behavior as the sensor radius decreases.

As we increase the number of agents, the quality of service in the zero on-site case should improve at least linearly with $1/m$, since we can subdivide the region. We will explore this possibility at length in Section 5.5. However, Figure 5-6 is suggestive of a super-linear improvement regime in the range of three to five agents for this particular

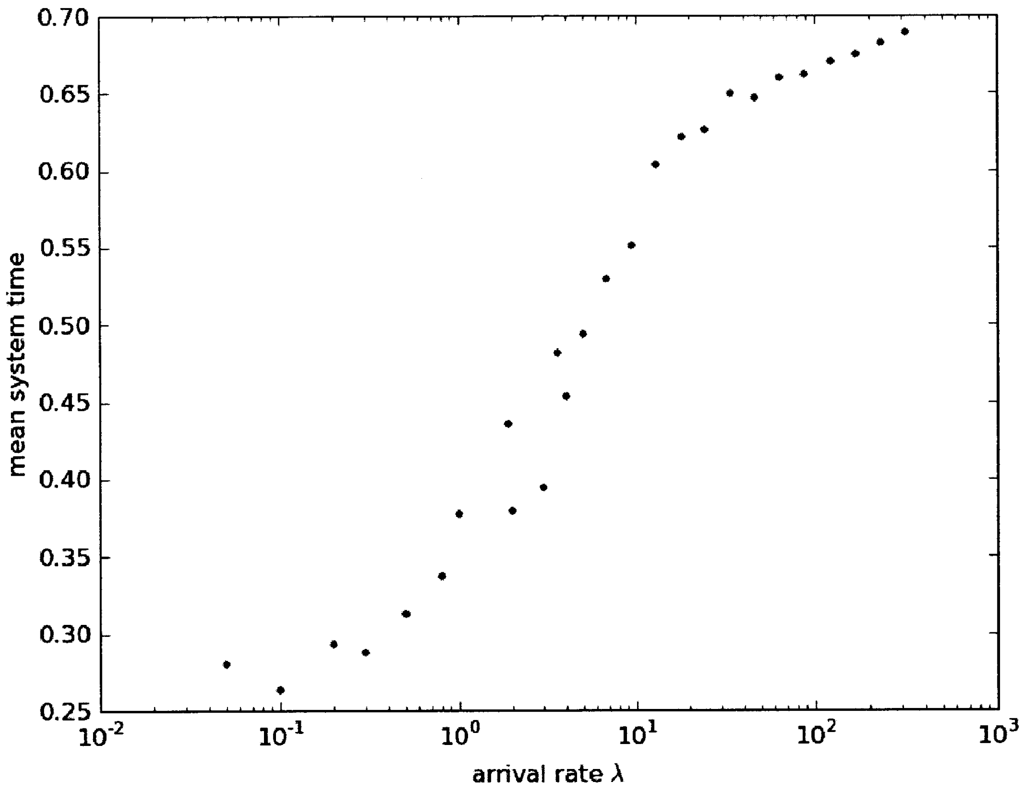


Figure 5-4: System time as a function of arrival rate for a single agent on the unit segment. The sensor radius is one and the on-site requirement is zero.

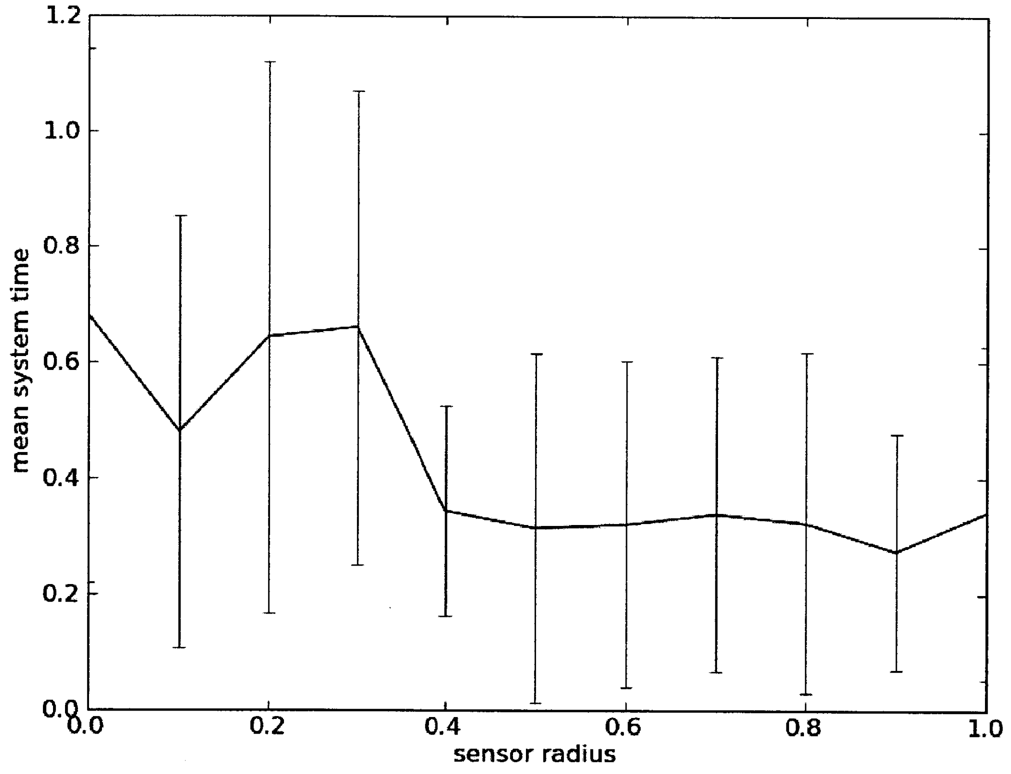


Figure 5-5: System time with increasing sensor radius. The arrival rate, $\lambda = 1$.

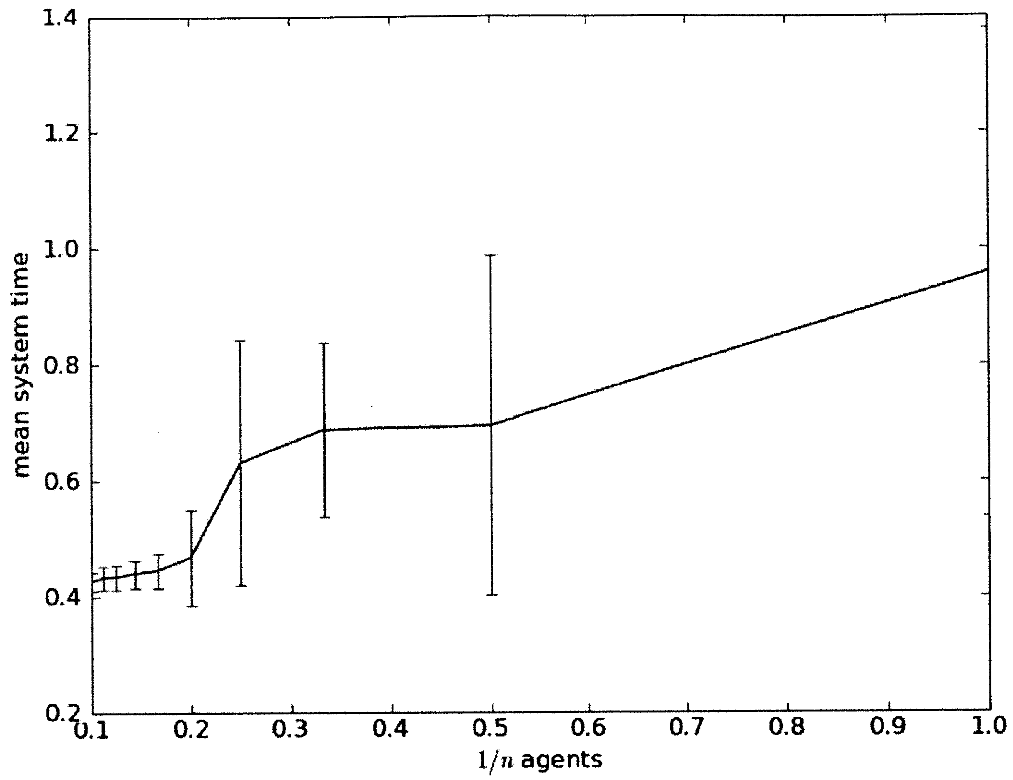


Figure 5-6: Wait time as function of the number of agents. The sensor radius was 0.1 of the region and the arrival rate was held constant. There appears to be a super-linear improvement between three and five agents, after which the system approaches the light-load limit.

example.

5.4.3 Segue

One might argue that the range of achievable performance is rather narrow—the best possible performance is eight-thirds of the worst performance. This level of sub-optimality might be acceptable. However, when we include the on-site service time, s, it can have a dramatic effect on performance. In the light-load, latency is only increased by $\overline{s}$, but the optimal performance in the heavy-load limit is proportional to

$$\frac{1}{1-\lambda\overline{s}}$$

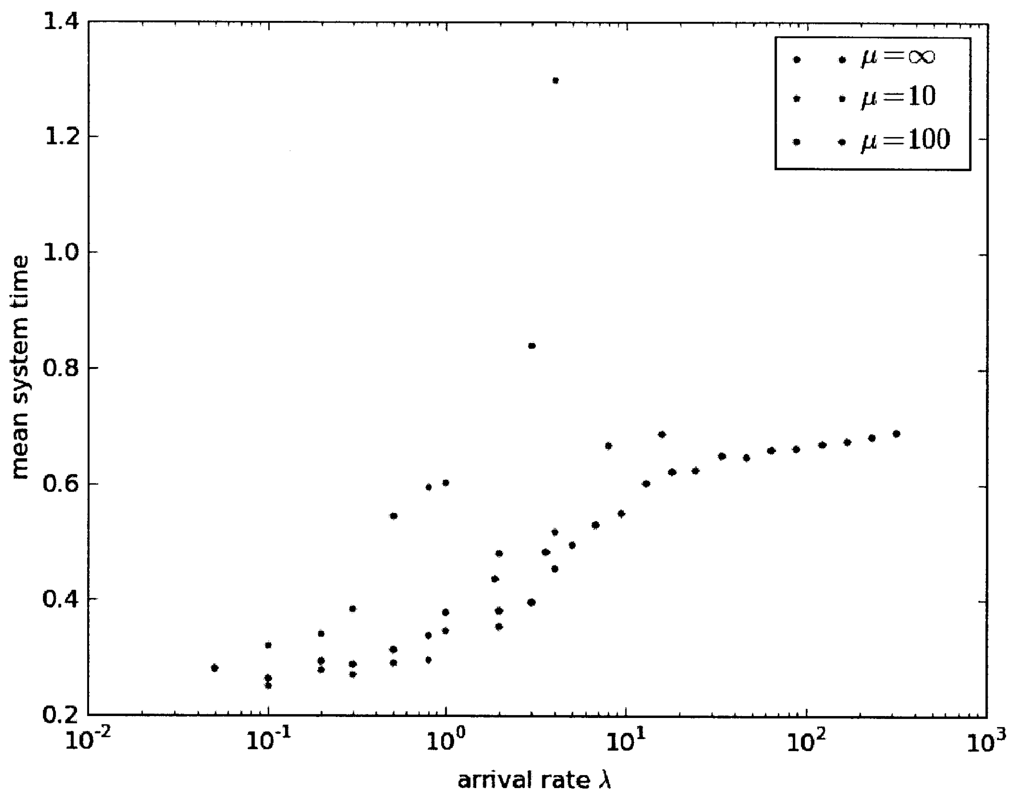


Figure 5-7: System time with increasing arrival rate for two different service rates μ.

as we will prove in Section 5.6. We refer to the quantity $\overline{\lambda s}$ as the load factor.

Even when the service rate is large compared to the arrival rate, it can still play a large role in the system time as evinced in Figure 5-7. The on-site service requirement was assumed to be poisson with rate $\mu = 10$ and $\mu = 100$, with the time unit being the time required to traverse the region. The effect of service time can be seen at load factors around 0.03 and they become very prominent by a load factor of 0.2.

5.5 The Light Load Limit

The light-load limit is characterized by the absence of queueing; when new requests arrive, there is almost surely an agent available to move to its location and serve it immediately. This is a natural model for systems that deal with rare, high-value events. For example, emergency services or high-value repair services are lightly loaded.

The light-load limit is defined as the limit in which the rate at which requests arrive, $\mathop{\mathbb{E}}_{t}[d\Lambda/dt]$, goes to zero. Strictly speaking, we must also require that the arrival process not be too *bursty* for the absence of queueing. A sufficient condition is that the inter-arrival times are i.i.d.. Poisson point processes, for instance, satisfy this requirement.

5.5.1 Voronoi Partitions

Definition 5.5.1 (Centroid, Median). *Given a metric space $\mathcal{A}, d(\cdot,\cdot)$ a monotone function $f : \mathbb{R}_+ \to \mathbb{R}_+$ and a measure $\Phi(\mathcal{A}^*) \to \mathbb{R}_+$, we will refer to*

$$\arg\min_{x^* \in \mathcal{A}} \int_{a \in \mathcal{A}} f(d(a, x^*)) d\Phi(a)$$

as a centroid. The special case of centroid in which f is the identity function will be referred to as the median.

Definition 5.5.2 (m-Partition). *We define a m-partition, $\mathcal{P}$, of a set $\mathcal{A}$, to be a set of m subsets $\mathcal{P}_i \subseteq \mathcal{A}$, for $i \in 1, \dots, m$, such that*

- *$\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$ for $i \neq j$, and*
- *$\bigcup_{i=1}^{m} \mathcal{P}_i = \mathcal{A}$.*

If the value of m is clear from context we will omit it.

Definition 5.5.3 (Voronoi Partition, generator). *Given a metric space $\mathcal{A}, d(\cdot,\cdot)$ and a finite set $c_i \in \mathcal{A}$ for $i \in 1, \dots, m$, a Voronoi partition, $\mathcal{P}(c)$ is an m-partition of $\mathcal{A}$ such that*

$$a \in \mathcal{P}(c)_i \implies d(a, c_i) \leq d(a, c_j) \quad \forall j$$

We refer to c as the generators of $\mathcal{P}(c)$ and c_i to be the generator of the $\mathcal{P}(c)_i$.

Note that Voronoi partitions need not be unique.

Given generators $c = \{c_i \in \mathcal{A}^m \text{ for } i = 1, \ldots, m\}$, let $G_c(a)$ denote the set of closest elements of c to location a.

$$G_c(a) \equiv \{c_i \in c \text{ s.t. } d(c_i, a) \leq d(c_j, a) \quad \forall i\}$$

Similarly, let $C(c)_i$ denote the set of points that are no further from c_i than any other generator.

$$C(c)_i \equiv \{a \in \mathcal{A} \text{ s.t. } d(c_i, a) \leq d(c_j, a) \quad \forall j\}.$$

Remark 5.5.4. *From these definitions it should be clear that,*

$$c_i \in G_c(a) \iff a \in C(c)_i.$$

It should also be clear that any partition $\mathcal{P}$ satisfying

$$\mathcal{P}(c)_i \subseteq C(c)_i \quad \forall i$$

is a Voronoi partition generated by c.

Lemma 5.5.5. *Let $\mathcal{A}$ be a length space and let c be a set of generators. Suppose that $a' \in \mathcal{A}$ is on a shortest path between c_i and some $a \in C(c)_i$. This implies that $a' \in C(c)_i$*

Proof.

$$\begin{aligned}
d(c_i, a) &\leq d(c_j, a) \text{ from } a \in C(c)_i \\
d(c_i, a) &= d(c_i, a') + d(a', a) \text{ from the shortest path principle} \\
d(c_j, a) &\leq d(c_j, a') + d(a', a) \text{ from the triangle inequality} \\
d(c_i, a') + d(a', a) &\leq d(c_j, a') + d(a', a) \text{ from substitution} \\
d(c_i, a') &\leq d(c_i, a')
\end{aligned}$$

□

Lemma 5.5.6. *Assume $a_i \in C(c)_i$ and $a_j \in C(c)_j$. If a_j is on a shortest path from c_i to a_i, then*

$$\begin{aligned} d(c_i, a_i) &= d(c_j, a_i) \text{ and} \\ d(c_i, a_j) &= d(c_j, a_j). \end{aligned}$$

Proof. First statement:

$$\begin{aligned} d(c_i, a_i) &= d(c_i, a_j) + d(a_j, a_i) \text{ from the shortest path principle} \\ d(c_i, a_i) &= d(c_j, a_j) + d(a_j, a_i) \text{ from Lemma 5.5.5} \\ d(c_i, a_i) &\geq d(c_j, a_i) \text{ from the triangle inequality} \\ d(c_i, a_i) &\leq d(c_j, a_i) \text{ from } a_i \in C_i \\ d(c_i, a_i) &= d(c_j, a_i) \end{aligned}$$

Second statement:

$$\begin{aligned} d(c_i, a_j) &\leq d(c_j, a_j) \text{ from Lemma 5.5.5} \\ d(c_i, a_j) &\geq d(c_j, a_j) \text{ from } a_j \in C_j \\ d(c_i, a) &= d(c_j, a) \end{aligned}$$

□

Theorem 5.5.7. *Given any generators c in a length space there exists a Voronoi partition each element of which is a length space.*

Proof. We construct the Voronoi partition $\mathcal{P}$ as follows. For notational compactness we drop the argument from $C(c)$.

$$\mathcal{P}_i = C_i \setminus \bigcup_{j=1}^{i-1} C_j$$

From Remark 5.5.4, this construction gives a Voronoi partition.

Assume by way of contradiction that $\mathcal{P}$ is not a length space. There exists some

$a_i \in \mathcal{P}_i$ such that for every shortest path $P(c_i, a_i)$ there exists a $a_j \in P(c_i, a_i)$ such that $a_j \notin \text{cl}(\mathcal{P}_i)$.

Consider such an a_j. Since $\mathcal{P}$ is a partition, a_j falls into $\mathcal{P}_j$ for some $j \neq i$. By Lemmas 5.5.5 and 5.5.6 $a_i, a_j \in C_i \cap C_j$, but this is inconsistent with how $\mathcal{P}$ was constructed. In particular,

- suppose $i < j$; a_j would not be in partition j, rather, it would be in some partition with index at most i.
- Otherwise $i > j$. In this case, a_i would not be in partition i. Instead, it would be in some partition with index at most j. □

5.5.2 Partition Policies

Theorem 5.5.8 (light-load optimality). *In the light load limit in a bounded region, a policy is optimal if and only if it almost always behaves as follows:*

1. *In the absence of requests, the system is almost always in configurations that maximize the expected reward of the first future request, and*
2. *when a request arrives it is served in such a way as to maximize its immediate reward.*

Proof. From Theorem 1.2.1 we can assume that the policy is a function only of the location of the agents and of the requests. In the light-load limit the probability that there will be more than one request is vanishing. Therefore we can restrict our attention to policies Π that are a deterministic function of the locations of the agent and of a single request location.

$[\implies]$. Consider the performance of policy Π with both properties on a randomly selected request r. Almost surely Π maximizes the reward of r. Maximizing the expected per request reward is the definition of optimality. □

[$\Longleftarrow$]. By way of contradiction, consider some optimal policy that lacks either property. Obviously, a policy lacking the first can be improved by improving the service of the first request and and then emulating the optimal policy.

Consider an optimal policy lacking the second property. Specifically, for some non-vanishing fraction of the time, the system has no outstanding requests but is in a configuration in which the expected reward of the first future request is not maximized.

Consider a randomly selected inter-request interval. Note that in the light-load limit, the length of this interval goes to infinity. We can construct a new policy that replaces a non-vanishing portion of the sub-optimal interval with one that moves into some optimal configuration (or an optimal sequence of configurations), stays there (or follows the sequence), and then rejoins the original policy. Since the region is bounded, the probability that a request will arrive during the transition is negligible.

However, there is a non-negligible probability that a request will arrive during an interval in which the new policy will have lower expected latency than the optimal policy. Therefore the original policy could not have been optimal. □

Definition 5.5.9 (Partition Policy). *Given a partition $\mathcal{P}$, and a list of m loitering locations, $c \in \mathcal{A}^m$ (i.e., generators) we define a light-load partition policy, $\Pi_{\mathcal{P}c}$ as any control policy in which*

- *agent i loiters at c_i in the absence of requests,*
- *if agent i is loitering at c_i and a single request appears at location $a \in \mathcal{P}_i$, the agent will immediately move to a and service the request, and*
- *subsequent requests cannot preempt this first request.*

Theorem 5.5.10. *An optimal light-load partition policy is light-load optimal with respect to all policies.*

Proof. From Theorems 5.5.8 it is sufficient to show that there exists a stationary point that maximizes the expected reward for the first request to arrive.

Consider the expected reward $\mathbb{E}[R]$ for the first request in a time interval $[t_0, t_1]$ for some optimal policy. Assume that t_0 is sufficiently large that we can ignore transient behavior and examine,

$$\mathbb{E}[R] = -\int_{t_0}^{t_1} \mathbb{E}_{\Phi}[d(s(\tau), r)]\, d\Lambda(\tau).$$

Consider in particular the expectation as a function of τ and let s^* be location of the minimizer on this interval. Since the location of the request does not vary by τ, s^* is also a minimizer of the expectation. Since Φ is stationary in time, we have

$$\mathbb{E}[R] \geq -\mathbb{E}_{\Phi}[d(s^*, r)] \int_{t_0}^{t_1} d\Lambda(\tau). \tag{5.3}$$

Thus a policy can do no better than loiter at the median of the set of locations where it would serve the first request to arrive. $\square$

Corollary 5.5.11. *The generators of the optimal partition policy are the medians of their individual partitions, from Equation 5.3.*

Lemma 5.5.12. *Let $\mathcal{P}$ be an optimal partition. There exists a Voronoi partition $\mathcal{V}_{\mathcal{P}}$, equivalent up to a set of measure zero, viz., one for which*

$$\Phi\left(\bigcup_i \mathcal{P}_i \setminus \mathcal{V}_{\mathcal{P}}(i)\right) = 0, \tag{5.4}$$

i.e., they differ only in sub-regions with no arrivals.

Proof. Assume by way of contradiction that there exists $a \in \mathcal{P}_i$ such that

- $d(c_i, a) > d(c_j, a)$ for some j, and
- $\epsilon > 0 \implies \Phi(\mathcal{P} \cap B_\epsilon(a)) > 0$.

Consider $\epsilon > 0$ such that $d(c_i, a) > d(c_j, a) + 3\epsilon$. Let $\mathcal{P}' = \mathcal{P}$ everywhere except $\mathcal{P}'_i = \mathcal{P}_i \setminus B_\epsilon(a)$ and $\mathcal{P}'_j = \mathcal{P}_j \cup (\mathcal{P}_i \cap B_\epsilon(a))$.

Consider the expected wait time in the event that a request falls in $\mathcal{P}_i \cap B_\epsilon(a)$ at location a_i. Under partition $\mathcal{P}$, we have a wait time

$$\begin{aligned} d(a_i, c_i) + d(a_i, a) &> d(a, c_i) && (5.5) \\ d(a_i, c_i) + \epsilon &> d(c_j, a) + 3\epsilon && (5.6) \\ d(a_i, c_i) + \epsilon &> d(c_j, a_i) + 3\epsilon - d(a, a_i) && (5.7) \\ d(a_i, c_i) - d(c_j, a_i) &> \epsilon && (5.8) \end{aligned}$$

Hence, the difference in wait time is bounded away from zero in the event of an arrival falling in this region. Since $\lim_{\delta \to 0} \phi(\mathcal{P}_i \cap B_\delta(a_i)) > 0$, we have $\phi(\mathcal{P}_i \cap B_\epsilon(a_i)) > 0$, *i.e.*, this event has non-vanishing probability. Therefore $\mathcal{H}(c, \mathcal{P}') < \mathcal{H}(c, \mathcal{P})$ which contradicts that $\mathcal{P}$ maximizes Equation 5.3. □

Lemma 5.5.13. *Any Voronoi partition policies with the same generators as an optimal partition policy is optimal.*

Proof. Consider two Voronoi partitions, $\mathcal{V}^1_{\mathcal{P}}(c), \mathcal{V}^2_{\mathcal{P}}(c)$ generated by c. Consider a randomly selected request at location $a \in \mathcal{V}^1_{\mathcal{P}}(c)_i \cap \mathcal{V}^2_{\mathcal{P}}(c)_j$. By Remark 5.5.4 $d(a, c_i) = d(a, c_j) = d$. In the light load, under either partition policy, the wait time will be d and the service will subsequently by identical. If one of these is optimal, Theorem 5.5.8, establishes the optimality of both. □

Theorem 5.5.14. *To within sets of measure zero, any optimal light-load partition policy is a fixed point, $\mathcal{P}, c$, where*

- *$\mathcal{P}$ is a Voronoi partition with generators c, and*
- *c_i is the median of $\mathcal{P}_i$ for all i.*

Proof. follows from Lemmas 5.5.11–5.5.13. □

Corollary 5.5.15. *Lemma 5.5.12 and Theorem 5.5.14 imply that we need not simultaneously search for optimal partitions and generators: Instead it is sufficient to search for generators whose whose Voronoi partitions are optimal.*

Having established Corollary 5.5.15, it is clear that the various ingenous algorithms (*e.g.*, [6, 39]) available for generating partitions in the Euclidean plane remain theoretically sound in length spaces.

5.6 The Heavy-Load Limit

In this section we will consider the limit in which the expected number of outstanding requests present in the system goes to infinity under an optimal policy.

5.6.1 Preliminaries

Theorem 5.6.1 (Little's Law[65]). *In any stable queueing system the expected wait time w of a request is related to the expected number of requests N in the the system by*

$$N = \overline{\lambda} w$$

wherein $\overline{\lambda}$ is the mean arrival rate

$$\lim_{t \to \infty} \mathbb{E}\left[\frac{\Lambda(t)}{t}\right].$$

Definition 5.6.2 (Heavy-Load Constant Factor). *Define the load factor $\rho \equiv \overline{\lambda}\overline{s}$. In the DTRP, the heavy-load limit is the limit of $\rho \to 1$. A policy Π is said to be Heavy-Load Constant Factor if*

$$\lim_{\rho \to 1} \frac{w^*}{w_\Pi} > 0$$

wherein w^* denotes the expected latency under an optimal policy and w_Π denotes the expected latency under policy Π.

Definition 5.6.3. *Pointwise Dimension*

The pointwise dimension, $D_P(\Phi, x)$ *of a measure* Φ *at point* $x \in S$ *is*

$$D_P(\Phi, x) \equiv \lim_{\epsilon \to 0} \frac{\log \Phi(B\epsilon(x))}{\log \epsilon}$$

wherein $B\epsilon(x)$ *denotes the* ϵ*-ball centered at* x.

Definition 5.6.4. *Hausdorff content*

The d-dimensional Hausdorff content *of a subset* S *of metric space* $\mathcal{A}$ *is defined as*

$$C_H^d(S) \equiv \inf \left\{ \sum_i r_i^d \textit{ s.t. } r_i \textit{ are the radii of balls that cover } S \right\}.$$

Definition 5.6.5. *Hausdorff Dimension*

The Hausdorff dimension of a subset S *of a metric space* $\mathcal{A}$ *is*

$$H(S) \equiv \inf\{d \geq 0 \textit{ s.t. } C_H^d(S) = 0\}$$

The Hausdorff dimension of a measure μ *is the infimum of the Hausdorff dimensions of sets with measure* 1.

$$H(\mu) \equiv \inf\{H(S) \textit{ s.t. } \mu(S) = 1\}$$

The pointwise dimension and Hausdorff dimension are related as follows [29]

$$H(\mu) = \inf\{\beta \text{ s.t. } D_P(x) \leq \beta \; \forall x \in S \text{ s.t. } \mu(S) = 1\}.$$

5.6.2 Asymptotic Nearest Neighbor Distance

Suppose that a set n points, a_i, are drawn i.i.d. according to probability measure μ in length space $\mathcal{A}$. The quantity of interest is the distance between a randomly selected point and its nearest neighbor. In particular, we are interested in the functional form of this distance as n becomes very large.

The general relationship between nearest neighbor distance and various notions of "dimension" in metric spaces is intimate, and is beyond the scope of this dissertation. Instead, we place the following restrictions on measures that we will consider in this section. First, we assume that $0 < H(\mu) < \infty$, and second, we make Assumption 5.6.6.

Assumption 5.6.6. *We will restrict our attention to measures whose pointwise dimension acheive a maximum within the region.*

Assumption 5.6.6 holds, even for fractals, as long as there is some set S for which $\mu(S) > 0$ such that for all $x \in S$ $D_P(x) = H(\mu)$. In particular we will denote by S_D^+ the largest such set.

For a point at x, the probability that a point randomly selected according to μ is within a distance ϵ is given by $\mu(B\epsilon(x))$. In the limit of $\epsilon \to 0$ this is related to the pointwise dimension by

$$\mu(B\epsilon(x)) \in \Theta(\epsilon^{D_P(\mu,x)}). \tag{5.9}$$

The relationship between pointwise dimension and nearest neighbor distance is well-known (see, *e.g.*, [29]).

$$d^{NN}(x,n) \in \Theta(n^{\frac{-1}{D_P(\mu,x)}}) \tag{5.10}$$

Lemma 5.6.7. *Under Assumption 5.6.6, in the limit of $n \to \infty$, the expected nearest neighbor distance shrinks as*

$$\mathop{\mathbb{E}}_{x}\left[d^{NN}(x,n)\right] \in \Theta(n^{\frac{-1}{H(\mu)}}) \tag{5.11}$$

$[\Omega]$. From Equation 5.10 there exists $k(x)^- > 0$ such that

$$\int_{x\in\mathcal{A}} d^{NN}(x,n)d\mu(x) \geq \int_{x\in\mathcal{A}} k^-(x)n^{\frac{-1}{D_P(\mu,x)}}d\mu(x)$$

From the definition of Hausdorff dimension,

$$\begin{aligned}\int_{x\in\mathcal{A}} k^-(x) n^{\frac{-1}{D_P(\mu,x)}} d\mu(x) &\geq \int_{x\in\mathcal{A}} k^-(x) n^{\frac{-1}{H(\mu)}} d\mu(x) \\ &\geq \min_{x\in\mathcal{A}} k^-(x) n^{\frac{-1}{H(\mu)}} \qquad \square\end{aligned}$$

[O]. From Equation 5.10 there exists $k(x)^+ < \infty$ such that

$$\int_{x\in\mathcal{A}} d^{NN}(x,n) d\mu(x) \leq \int_{x\in\mathcal{A}} k^+(x) n^{\frac{-1}{D_P(\mu,x)}} d\mu(x)$$

Restricting our attention to the set S_D^+ implied by Assumption 5.6.6

$$\begin{aligned}\int_{x\in\mathcal{A}} k^+(x) n^{\frac{-1}{D_P(\mu,x)}} d\mu(x) &\leq \frac{1}{\mu(S_D^+)} \int_{x\in S_D^+} k^+(x) n^{\frac{-1}{D_P(\mu,x)}} d\mu(x) \\ &\leq \frac{1}{\mu(S_D^+)} \int_{x\in S_D^+} k^+(x) n^{\frac{-1}{H(\mu)}} d\mu(x) \\ &\leq \frac{1}{\mu(S_D^+)} \max_{x\in S_D^+} k^+(x) n^{\frac{-1}{H(\mu)}} \qquad \square\end{aligned}$$

Theorem 5.6.8. *In the heavy-load limit, there exists some $c < \infty$ such that the expected waiting time is lower bounded by*

$$W \geq \frac{c}{\overline{\lambda}} \left(\frac{\overline{\lambda}}{1-\rho}\right)^{H(\mu)}. \tag{5.12}$$

Proof. The proof mirrors that of Lemma 2 from [23]. In the following we will drop the argument from $H(\mu)$.

If the queue is unstable then the bound holds trivially since the expected wait is infinite. For the queue to be stable, we must have

$$\overline{s} + \bar{D} \leq \frac{1}{\overline{\lambda}}. \tag{5.13}$$

wherein $\bar{D}$ is the expected distance between two requests served consecutively by the agent.

Consider a randomly tagged request conditioned on the request being at location $x \in S_D^+$ and consider the set $S \equiv B_\epsilon(x) \cap S_D^+$. From the definition of S_D^+, S must have a non-empty interior, specifically, the dimension of S must be H.

Let $n_S^+(S)$ denote the number of requests in S at the moment of the completion of the service of the tagged request, and let $N_S(S)$ denote the time average number of requests in S. Let $W(S)$ denote the expected wait time for a randomly selected request in S. From Little's law we have $\overline{\lambda} W(S)\mu(S) = N_S(S)$. Define

$$\phi(x) \equiv \lim_{\epsilon \to 0} \frac{\mu(B_\epsilon(x))}{\epsilon^H}$$

with $0 < \phi(x) < \infty$ for $x \in S_D^+$. Hence, we have

$$\overline{\lambda} W(S)\phi(x)\epsilon^H = N_S(S)$$

By Lemma 1 of [23] (a "law of small numbers" argument), we assume that the arrivals into S are approximately Poisson. Specifically,

$$\mathbb{E}\left[n_S^+(S)\right] = N_S(S) + o(\epsilon^H).$$

Let z^* denote the distance from the tagged request to its nearest neighbor at the time of its completion of service. Since n_S^+ is non-negative integer-valued, we have

$$\mathbb{P}\left[z^* < \epsilon | x\right] = \mathbb{P}\left[n_S^+(B_\epsilon(x)) = 0\right] \le \mathbb{E}\left[n_S^+(B_\epsilon(x))\right].$$

Since S has a non-empty interior, we have $\mu(B_\epsilon(x) \setminus S) = 0$ for almost all $x \in S_D^+$. Therefore

$$\mathbb{P}\left[z^* < \epsilon | x\right] \le \mathbb{E}\left[n_S^+(S)\right].$$

Let $c \equiv N\phi(x)$

$$\begin{aligned}
\mathbb{E}[z^*|x] &= \int_0^\infty \left(1 - \mathbb{P}[z^* < \epsilon|x]\right) d\epsilon \\
&\geq \int_0^\infty \max\{0, 1 - N\phi(x)\epsilon^H - o(\epsilon^H)\} d\epsilon \\
&\geq \int_0^{c^{-1/H}} (1 - c\epsilon^H) d\epsilon - N \int_o^{c^{-1/H}} o(\epsilon^H) d\epsilon \\
&= \frac{H}{H+1} c^{-1/H} - o(N^{-1/H})
\end{aligned}$$

Now we uncondition on x, but continue to restrict ourselves to S_D^+.

$$\begin{aligned}
\mathbb{E}\left[z^*|x \in S_D^+\right] &= \int_{S_D^+} \frac{H}{H+1} c^{-1/H} dx - o(N^{-1/H}) \\
\mathbb{E}\left[z^*|x \in S_D^+\right] &= \frac{H}{H+1} N^{-1/H} \int_{S_D^+} \phi(x)^{-1/H} dx - o(N^{-1/H})
\end{aligned}$$

Define $k_\phi \equiv \int_{S_D^+} \phi(x)^{-1/H} dx$. From $0 < \phi(x) < \infty$ we have $0 < k_\phi < \infty$. Removing the restriction on $x \in S_D^+$ we arive at

$$\begin{aligned}
\mathbb{E}[z^*] &= \mathbb{E}\left[z^*|x \in S_D^+\right] \mu(S_D^+) + \mathbb{E}\left[z^*|x \notin S_D^+\right] (1 - \mu(S_D^+) \\
&\geq \mathbb{E}\left[z^*|x \in S_D^+\right] \mu(S_D^+) \\
&\geq k_\phi \frac{H}{H+1} N^{-1/H} - o(N^{-1/H}) \\
&= k' N^{-1/H}. \qquad (5.14)
\end{aligned}$$

For some k' and large enough N.

Since $\bar{D} \geq \mathbb{E}[z^*]$ and $N = W\overline{\lambda}$ by Little's law, the theorem follows from Equation 5.13 and Equation 5.14. □

Corollary 5.6.9. *By comparing Equations 5.14 and 5.11 it is clear that a sufficient condition on a policy being Heavy-Load Constant Factor is that the average inter-request travel distance is $O(\bar{D})$.*

Corollary 5.6.9 can be used to prove that many of the heavy load policies developed for the Euclidean plane remain constant factor optimal in length spaces.

Lemma 5.6.10. *The index policy is Heavy-Load Constant Factor.*

Proof. For path $\mathcal{P}$ visiting $n_{\mathcal{P}}$ requests, the index function can be bounded by

$$\left(\frac{n}{n_{\mathcal{P}}} - 1\right) \text{TSP}(\mathcal{P}) \le \gamma(\mathcal{P}) \le \left(\frac{2n}{n_{\mathcal{P}}} - 1\right) \text{TSP}(\mathcal{P})$$

wherein TSP denotes the length of the TSP on the nodes covered by the path. In the limit of large $n_{\mathcal{P}}$ the length of the minimum spanning tree has length proportional to $n_{\mathcal{P}} d^{NN}$[61]. Since a traversal of a spanning tree is a tour with length twice that of the tree, this ensures that the length of the TSP tour is $O(n_{\mathcal{P}} d^{NN})$. Since every node must be connected to *two* neighbors we have that the lengh of the TSP tour is $\Theta(n_{\mathcal{P}} d^{NN})$

Assume by way of contradiction that the path minimizing the index function has an inter-request distance in $\omega(d^{NN})$. Clearly this path must include a vanishing fraction of all requests for this to be possible, letting us approximate $\frac{n}{n_{\mathcal{P}}} - 1$ as $\frac{n}{n_{\mathcal{P}}}$ and giving us $\gamma(\mathcal{P}) \in \omega(nd^{NN})$.

On the other hand, there exits the path, $\mathcal{P}'$, that visits every request for which $\gamma(\mathcal{P}') \in O(nd^{NN})$. Therefore we have, for sufficiently large n, $\gamma(\mathcal{P}) > \gamma(\mathcal{P}')$ which is a contradiction since $\mathcal{P}$ was selected by the index policy. □

Corollary 5.6.11. *The simplified index policy is Heavy-Load Constant Factor.*

Proof. Because the length of the minimum spanning tree also grows as $\Theta(nd^{NN})$, the proof is identical to that of Lemma 5.6.10 replacing the TSP length with twice the spanning tree length. □

Remark 5.6.12. *The nearest neighbor algorithm is not Heavy-Load Constant Factor.*

Proof. We show this in the following example. Let the region be the unit interval and let the arrival process be supported by only the endpoints with arrivals being equally probable on each. Let the service time be deterministically $\overline{s} \ll 1$ and let the temporal arrival process be Poisson with rate $\lambda = \rho/\overline{s}$.

Consider the moment at which the agent has finished the last outstanding request at the zero-endpoint and begins traveling towards the other endpoint. With probability

$$p \equiv \exp\left[-\frac{\lambda}{4}\right]$$

the agent makes it halfway before a new request arrives at the zero endpoint. For any $\rho < 1$ and we can choose a sufficiently small $\overline{s}$ (hence sufficiently large λ) that this probability is as small as we want. As a result we can make the expected number of requests at the other endpoint excede Equation 5.12 for any fixed c. □

However, we present the following modified nearest neighbor algorithm, BOUNDEDNN.

```
Data: Agent location, $x$, outstanding requests $\theta$, distance threshold $d^-$
Result: A request to visit next
$r \leftarrow \arg\min_{r\in\theta} d(x,r)$;
if $d(x,r) \leq d^-$ then
3   | return $r$
else
5   | return a randomly selected request
end
```

Procedure boundedNN

Lemma 5.6.13. *Letting*

$$d^- = k(\lambda W)^{\frac{-1}{H(\mu)}}$$

wherein $k > 0$ and W is as defined in Equation 5.12, Algorithm BOUNDEDNN *is Heavy-Load Constant Factor.*

Proof. Let

$$W^* \equiv \frac{c}{\overline{\lambda}}\left(\frac{\overline{\lambda}}{1-\rho}\right)^{H(\mu)}$$

and assume the contrary, *i.e.* that under this algorithm the wait time is in $\omega(W^*)$. By Little's law, the number of outstanding requests is therefore in $\omega(\lambda W^*)$. From

Equation 5.14 we have that the expected nearest neighbor distance is therefore in $\Theta((\lambda W)^{\frac{-1}{H(\mu)}})$. Since this distance is bounded below by zero and above by the diameter of the region, it must have finite variance. Hence with probability one there will be a neighbor closer than d^-.

As a result we can neglect the occasions when a random request is visited when computing the expected inter-request distance. This distance is in $\Theta(n^{\frac{-1}{H(\mu)}})$ which by Corollary 5.6.9 contradicts our assumption. □

Algorithm BOUNDEDNN relies on having being able to compute an appropriate d^- from the problem specification. In Algorithm MODNN we use the realization of the process to estimate the nearest-neighbor distance.

```
Data: Agent location $x$, set of outstanding requests $\theta$, Parameter $\beta > 2$.
Result: plan consisting of a single request to visit next
$d^{NN} \leftarrow$ the mean nearest neighbor distance within $\theta$;
return Algorithm BOUNDEDNN with $x, \theta, \beta d^{NN}$
```

Procedure modNN

Theorem 5.6.14. *Algorithm* MODNN *is Heavy-Load Constant Factor.*

Proof. Assume by way of contradiction that the expected inter-request distance $\overline{d} \in \omega(N^{\frac{-1}{H(\mu)}})$ under this policy and expected wait time $W \in \omega\left(\frac{c}{\overline{\lambda}}\left(\frac{\overline{\lambda}}{1-\rho}\right)^{H(\mu)}\right)$. The inter-request distance for requests returned in Line 3 are shorter than $\overline{d}$ by design. The inter-request distance returned by Line 5 is bounded by the diameter of the region. To show a contradiction, we will demonstrate that Line 3 is almost always applied.

Let $N^* \equiv c\left(\frac{\overline{\lambda}}{1-\rho}\right)^{H(\mu)}$, for some c such that an upper bound on the expected number of requests under an optimal policy. Let $d^{NN^*} \equiv N^{*\frac{-1}{H(\mu)}}$ be a lower bound on the nearest neighbor distance under an optimal policy.

Consider a ball of radius $\beta d^{NN^*}/2$ around a request at location a returned in Line 5. Let $N \in \omega(N^*)$ denote the number of requests in the region and let n denote the number of requests in that ball. We have $\mathbb{E}[n] \geq N\mu(B_{\beta d^{NN^*}/2}(a))$ since the request was chosen at random. By assumption on W, Little's law, and Equation 5.11 we have $\mathbb{E}[n] \in \omega(1)$ *i.e.* it goes to infinity.

Consider the expected number of requests visited under Line 3 before the next application of Line 5. For this to happen, either all n requests have been served or the agent has left the ball. Since n goes to infinity, we need only examine the later case.

The path taken by the agent must have length exceeding $\beta/2d^{NN}$ and consist of a sequence of nearest neighbors. Recall that the expected nearest neighbor distance within this ball is bounded by

$$d^{NN}(x, N) \leq kN^{-1/D_P(a)} \leq kN^{-1/H}$$

The expected number of nearest neighbors in the sequence is bounded by

$$\frac{\beta d^{NN*}}{2d^{NN}(x, N)} \geq k(N/N^*)^{1/H}$$

which also goes to infinity for finite H. □

5.6.3 Multi-Agent Policies

Conjecture 5.6.15. *Let Π be a Heavy-Load Constant Factor single-agent policy and f be a scheme for assigning requests to agents. Let $n_i(t)$ denote the number of requests assigned to agent i at time t and let $n(t)$ denote the total number of outstanding requests. If, in the limit of $n(t) \to \infty$*

$$\mathop{\mathbb{E}}_{t}\left[\frac{\sum_i n_i(t)}{n(t)}\right] > 0$$

i.e., a non-vanishing fraction of outstanding requests are assigned, and

$$\mathop{\mathbb{E}}_{t}\left[\frac{(n_i(t) - n_j(t))^2}{n_i(t)^2}\right] = 0 \quad \textit{for } i \neq j$$

i.e., there is bounded inequity between agents, then the multi-agent policy in which each agent runs Π on the requests it has been assigned by f is also Heavy-Load Constant

Factor.

Examples of such assignment schemes are

- equitable spatial partitions,
- sequential assignment, and
- random assignment.

See [75] for an extensive discussion of equitable partitioning schemes.

5.7 Search and the Intermediate Load Case

We do not consider the asymptotic cases of the DSRP in isolation.

- The light-load limit corresponds exactly to the PPP.
- In the heavy-load limit the sensor is no longer relevant.

The former is clearly the case since when a request is detected it is almost surely the only request in the region and the only reasonable thing to do is to serve it. The latter is true not simply because agents are sure to see requests, but because the number of requests in places they cannot see becomes predictable. This was formally proven in [34].

Furthermore, we do not give special treatment to the intermediate load case of the DTRP. Instead we will do as we did in Section 3.3 and treat the observable case as just another probability measure in the DSRP.

We will continue to focus on the single-agent case, with the presumption that we can use an algorithm such as those described in Section 3.5 or the partition and assignment schemes described in Sections 5.5–5.6 to adapt a single-agent policy into a multi-agent policy.

5.7.1 Index Policy

The question at hand is if and how we are to modify the index function to handle the particular difficulties that differentiate the DSRP from the PPP. As before, we do so by analogy. The numerator of the index function represents cost incurred by an agent paying unit cost for motion. Previously, time and distance were interchangeable, now we must make the distinction that the agent must pay unit cost per unit *time.* Previously the agent expected to terminate its search receive its reward the moment that it found a goal. Now we include the additional time required to move to it *and* serve it. Having made these changes the index function $\gamma(\mathcal{P})$ has a clear interpretation as the minimum price γ at which the agent will accept a "path service contract" with the following character.

- A single request is drawn according to the spatial distribution.
- The agent pays unit cost per unit time.
- If the agent finds the request along that path, it will deviate to service it.
- If it does so, it is paid γ *upon completion* of that service and stops.

The index policy is to pursue the path of minimum price. Also recall from Section 5.4.1 we also consider the path that does not move. Specifically, we assume that $\gamma(\emptyset)$ corresponds to loitering until the next request to arrive within r_s of the agent's location.

5.7.2 Intermediate Load

Recall from Section 3.3.1 that the Nearest Neighbor Algorithm performed very well on the MLTP for randomly generated instances in the Euclidean plane. In the DTRP being far-sighted is likely to help even less than in the MLTP. This suspicion is confirmed by examining the performance of the Nearest Neighbor Algorithm on the DTRP in the Euclidean plane.

Figures 5-8a.i–5-8c.ii are taken from [21] and [72] which compare various algorithms being proposed. The definitive winner for all but the light-load case was the

ρ (from\to)	0.10	0.19	0.28	0.37	0.46	0.54	0.63	0.72	0.81	0.90
0.10	n	n	n	n	n	n	y	y	y	y
0.19		n	n	n	n	y	y	y	y	y
0.28			n	n	n	n	y	y	y	y
0.37				n	n	n	y	y	y	y
0.46					n	n	y	y	y	y
0.54						y	y	y	y	y
0.63							y	y	y	y
0.72								y	y	y
0.81									y	y
0.90										y

Table 5.1: Statistical significance ($p = .001$) of the difference between the index policy and the Nearest Neighbor Algorithm.

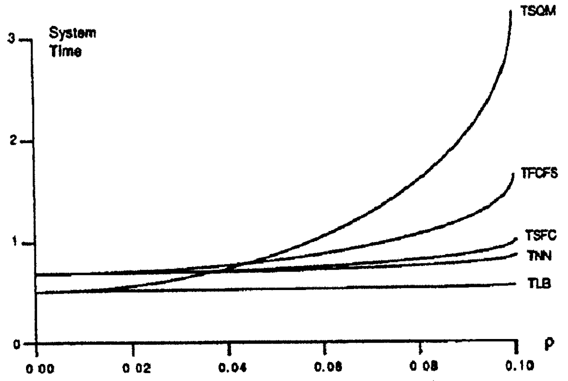
Nearest Neighbor Algorithm. Using the theory from Section 5.5 it is clear that one could make the Nearest Neighbor Algorithm perform optimally in the light load case simply by having it return to the median in the absence of requests.

Figure 5-9 shows the degree to which the index policy improves on this performance. The service times were assumed to be negative exponential as were the inter-arrival times. The service time mean was held fixed at 0.1 while the arrival rate was varied. Each point represents a realization of either a million requests (shown by an 'o') or one hundred thousand requests (shown by an 'x') on which both policies were run.

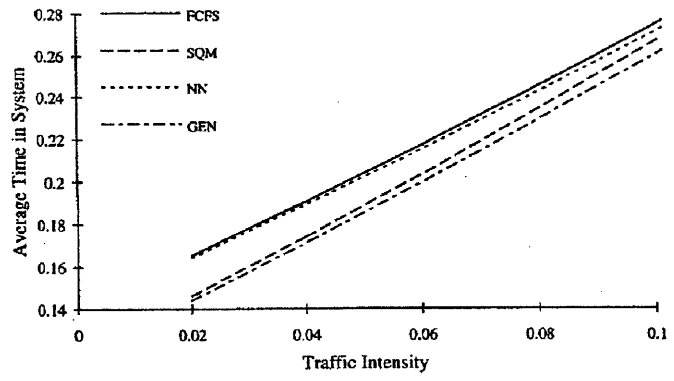
In the absence of requests, both policies loitered at the location of the last request to be serviced for consistency with Figure 5-8. In any case, having the agents return to the center did not appear to affect the comparison.

For larger service rates, there was significant variation in the expected service time between realizations, but the performance of the policies was always very similar.

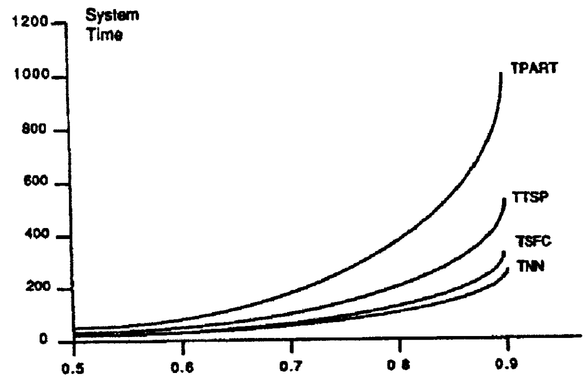
Table 5.1 shows the results of the Wilcoxon signed-rank test on the data shown in Figure 5-9, normalizing at each level of load. A "y" in row i column j indicates a 99.9% confidence in the statement, "The index policy has lower expected latency than the Nearest Neighbor Algorithm for load-factors between i and j." An "n," so located, indicates the lack of such confidence. Nowhere is there substantial confidence in the negation of these statements.



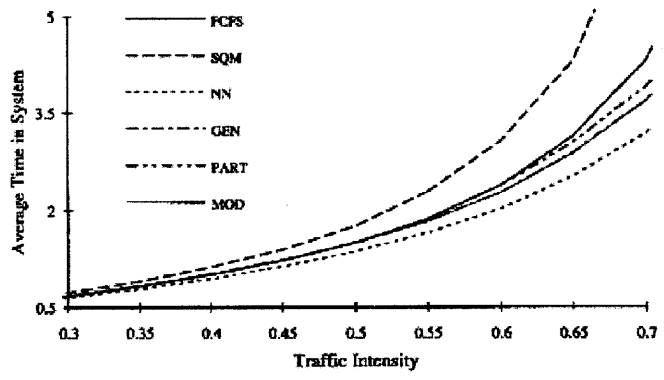
(a.i) Light-load



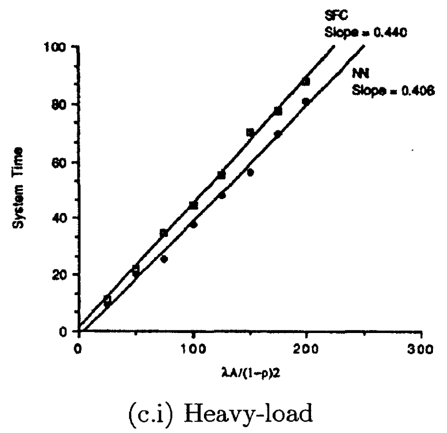
(a.ii) Light-load



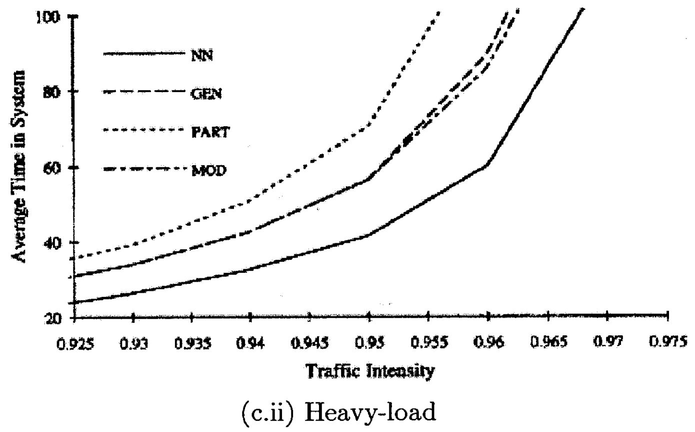
(b.i) Intermediate-load



(b.ii) Intermediate-load



(c.i) Heavy-load



(c.ii) Heavy-load

(i) [21]

(ii) [72]

Figure 5-8: Comparison of algorithms on DTRP [used with permission]

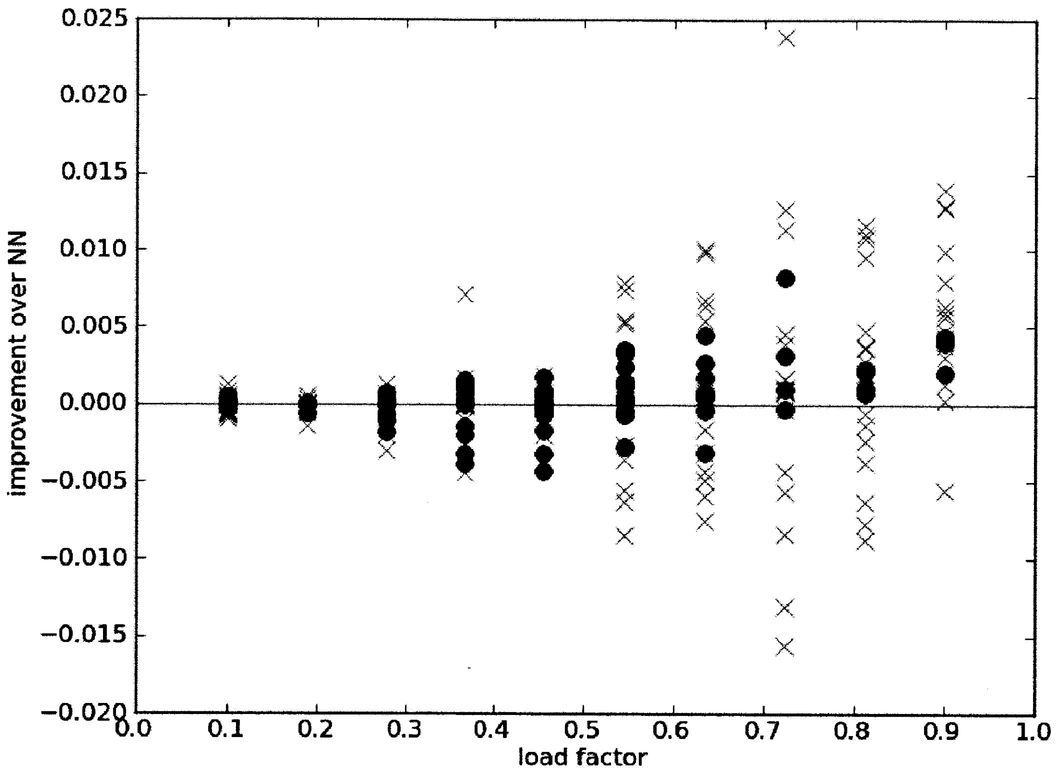


Figure 5-9: Comparison of the index policy and Nearest Neighbor Algorithm. Circles indicate sequences of one million requests, while x-symbols represent sequences of one hundred thousand requests and are shown to better describe the variability.

5.7.3 Multi-Class Requests and Preemptive Service

Suppose that the on-site service requirement, s does not have a non-increasing conditional mean. It is possible that after having provided some amount of service to a request, continuing to service that request might require a higher subsidy than moving to another request. Figure 5-10 shows a probability distribution for the on-site service requirement, s, for which this is the case. For such a distribution an agent should only work on the "hard ones" if it is relatively certain that there are no "easy ones."

A similar effect can be acheived if we will allow the ability to distinguish between different types of service requests. In this case we will assume each request comes from a set of *classes*, Ψ. Assume that the class of a request is drawn i.i.d. from P_Ψ that is both spatially and temporally constant. Each class i may have a different service time distribution s_i, and a different *priority*, q_i, the meaning of which we now define.

Rather than minimize the expected latency of requests, we modify the overall objective to minimize the expected *prioritized* latency of requests, wherein the latency of a request of a particular class is scaled by the"priority" of that class. For simplicity, we assume that when a request is detected its class is is also detected, although this is not strictly necessary.

This section will broach these issues but does not resolve them.

First note that, the subsidy scheme described in Section 5.7.1 handles these cases conceptually: The agent gets its reward when it completes service, and scaling that

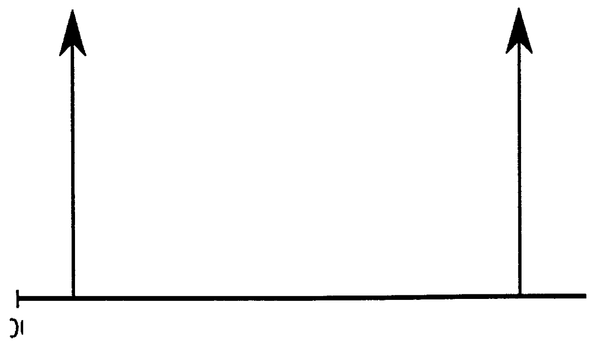


Figure 5-10: A probability distribution lacking a non-increasing conditional mean.

reward by a class's priority is natural. But at an algorithmic level, the index function we have presented does not handle them. We have all along been representing the spatial distribution of requests with a single, normalized measure. Once we are able to differentiate requests this becomes immediately inadequate.

Instead we propose tracking the actual (unnormalized) spatial distribution of each request type. Then to apply the index policy, one must consider not only a path, but a decision rule for whether a request will be served. Consider a request r of type i a distance d away from the agent. Ignore the issue of preemption for the moment. Serving it corresponds to a price of

$$\gamma_r \equiv \frac{d + \overline{s_i}}{q_i}$$

We propose the following decision rule. A request, r encountered along path $\mathcal{P}$ at time t is serviced if γ_r is less than the subsidy of $\mathcal{P}$ from t onward. To compute the subsidy of the path, therefore, we must now condition on the fact that an encountered request is actually serviced.

Preemption further complicates the issue since we now must also condition on the *completion* of service. Let $\gamma(\mathcal{P})$ denote the break-even price for the best path that leaves the location of a request r. The agent will only be willing to work on r as long as

$$\frac{\mathbb{E}\left[s_i - t | s_i > t\right]}{q_i} \leq \gamma(\mathcal{P})$$

Let $t_{\max}(\gamma)$ the largest t for which this holds for all $\tau < t$. With probability $\mathbb{P}\left[s \leq t_{\max}\right]$ the agent will complete the service, but otherwise it will introduce $t_{\max}$ of delay. When determining the break even price of a path, this potential delay must also be accounted for. We have not yet found a general and tractible way of determining, or even estimating, this price. Doing so would constitute an important extension of this research.

We conclude with the following remark.

Remark 5.7.1. *Assume that service is either non-preemptive or that agent's can* *resume tasks without penalty, i.e., request i is "serviced" after the* cumulative *time spent on it is equal to s_i.*

If the arrival process is supported by a single point, the index policy is optimal.

Proof. The subsidy of observed requests will be $q_i/\overline{s_i}$. The queueing policy that sorts using this rule is known to be optimal for non-spatial multi-class queueing.[64] □

Conjecture 5.7.2. *In the limit of the agent's speed to infinity, the index policy is optimal.*

5.8 Flight Testing

Now, finally, we are able to completely describe the policy that we developed for a scenario of interest for the Air Force Research Lab (AFRL). A long standing goal in military UAV control is to allow a single operator to *effectively* control a team of UAVs. Currently UAV operators control only one UAV, in fact often several operators are required for a single vehicle. These operators give commands at the waypoint level, *i.e.*, by telling the vehicles exactly where to go. The vehicles have extremely capable autopilots, but currently lack the necessary decision-making abilities to behave more autonomously.

One of the programs aimed at providing these abilities is called Intelligent Control and Evaluation of Teams (ICE-T) the goal of which is to allow operators to *monitor* and *evaluate* teams of UAVs while controlling them at a much higher operational level. The operator interacts with the autonomy via the Vigilant Spirit Control Station (VSCS) which is described as follows.

> The Vigilant Spirit Control Station (VSCS) is a multiple UAV ground control station. It has been developed by the Human Effectiveness Directorate to allow a single operator to control a team of UAVs. It connects to the UAVs via the STANAG 4586 messaging standard. Automation services connect to VSCS using Common Mission Automation Services

> Interface (CMASI) which is a proof-of-concept messaging protocol for so-called "mission-level" messages. One could think of CMASI as the control architecture—it describes what tools are available to the operator to control multiple UAVs.[58]

The CMASI architecture allows for the discovery and use of high-level *autonomy services.* We implemented an autonomy service that, given a description of a DSRP,provides a multi-agent controller using the algorithms described in this dissertation. The various details of this controller are described in Algorithm IMPLEMENTATION. This controller issued waypoints to the vehicle and also communicated with the operator via the VSCS.

The VSCS and associated autonomy services were used in the 2009 Talisman Saber Exercise (TS) which was the largest force-on-force military exercise in the world[67] and our autonomy is planned on being included in the 2011 TS.

The scenario of interest involves the aerial surveillance of a road network. *Targets,* formerly "requests," arrive in the network according to a known random process and the UAVs must collect imagery of them so that a human operator could classify the target as "lethal" or "benign." The system objective is to minimize the expected time between a random target's arrival and its classification. These targets may be detected in one of three ways: completely exogenously (*i.e.*, at the request of unmodelled assets), exogenously via known assets (*e.g.* fixed ground sensors), or by the UAVs themselves.

Since Automated Target Recognition (ATR) is not a viable option, the search aspect of the problem involves the help of the human operator. When not collecting imagery of a known target, the UAVs collect imagery so as to maximize the probability that it will collect imagery of an unknown target. The presumption is that the human, coupled with whatever decision support services are available, constitute the perfect sensor we have been assuming throughout. Incorporating a model for the human's calculable flaws is a rich and ongoing research endevour, see, *e.g.*, [82].

We assumed a model for the human operators' decision time which lacked non-increasing conditional mean giving the vehicles the ability to preempt, however under realistic arrival rates such preemption was an unlikely occurance. We allowed the road network to be embedded in an airspace with nearly arbitrary no-fly zones, requiring only the assumption of symmetric distances.

Figure 5-11 depicts the state of a specific simulation in which two UAVs are responsible for a security perimeter at Vandenburg Air Force Base (AFB), shown in red. The UAVs under consideration are Bat-3, with six-foot wingspans and capable gimballed cameras. This simulation assumes that the arrival process is supported by the perimeter and is very non-uniform. There are ground sensors at points of interest, such as where roads intersect the perimeter, and in this simulation they are very accurate.

At the moment depicted in Figure 5-11 there are no known outstanding requests. The expected target density is shown as yellow vertical bars. The blue lines indicate the ground track of the sensor for the planned trajectories, specifically the paths that minimize the index function: The inland UAV plans on searching to the high density peak while the more coastal UAV plans on searching the entire shore.

Since there is no point of comparison for our policy, the performance metric under consideration is qualitative. Ultimately the goal is to have a control policy that the operator trusts. We would like for the operator to decide that he or she need not routinely intervene in path planning, specifically because this would provide little to no improvement.

Systematic human experiments using our algorithms in the VSCS are not yet being planned. Anecdotally though, there is little to find fault with in the simulated results. We are entirely confident that operators would trust the autonomy sufficiently well to rely on it for path planning at any time when they have other tasks requiring their attention.

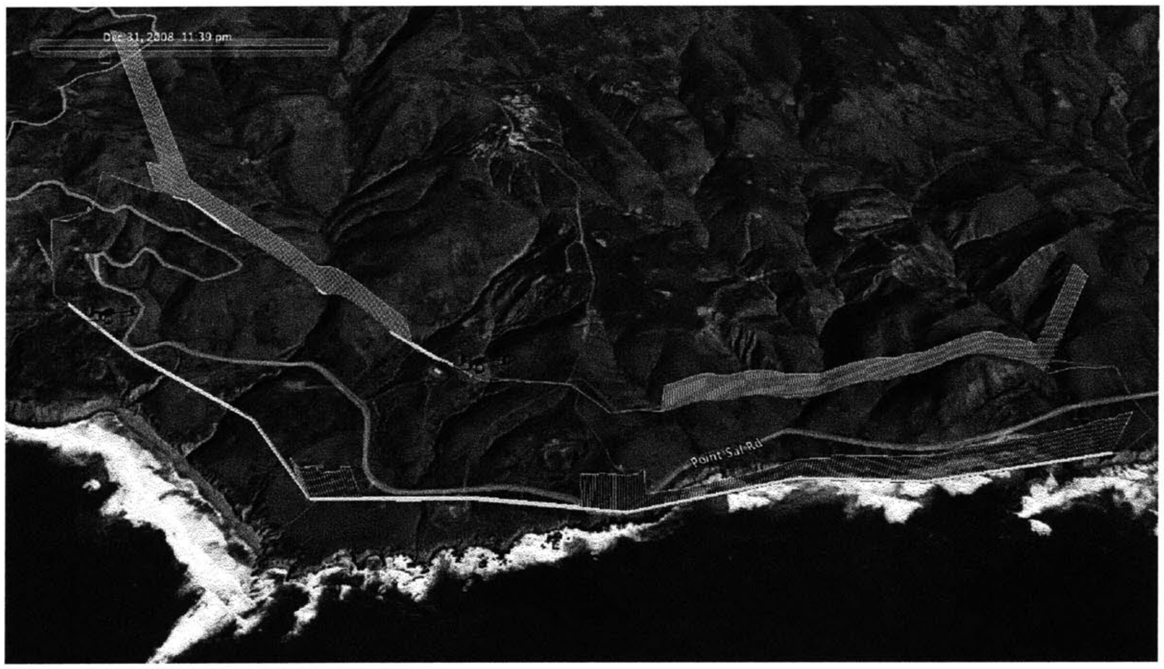


Figure 5-11: Surveillance region at Vandenburg AFB. The yellow vertical bars indicate expected target density. The blue lines indicate the paths with minimum subsidy.

Chapter 6

Conclusions and Possible Future Directions

6.1 Summary

This dissertation presents a unified approach to search based on a single metric, which we call the "index function," by which we compare partial search paths in a length space $\mathcal{A}$. This metric evaluates paths that search a subset of the region. If the path is finitely long it must be closed. The index function is the expected time spent searching the path divided by the probability of success.

Specifically, "time spent searching" is the time at which the agent finds (and in the case of the DSRP *services*) a single goal drawn according to a spatial probability measure μ, or the time required to traverse the entire path, whichever is shorter. By restriction or assumption we represented the spatial distribution of relative agent–goal positions in the region with a single probability measure μ.

Given a path $\mathcal{P} : \mathbb{R}_+ \to \mathcal{A}$ define $I_\mathcal{P}(\tau)$ as an indicator of whether an agent that pursues path $\mathcal{P}$ will be "successful" before time τ if exactly one goal is drawn according to μ. In most cases "successful" simply means finding the goal, but in the case of the DSRP it means both finding and servicing it. Let ℓ denote the length of the path, potentially infinite. The index function is given by

$$\gamma(\mathcal{P}) = \frac{\mathbb{E}\left[\int_0^\ell \tau(1 - I_\mathcal{P}(\tau))d\tau\right]}{I_\mathcal{P}(\ell)}. \tag{6.1}$$

The "index policy" is to select the path that minimizes Equation 6.1. In addition to defining this function, this dissertation presents a collection of recipes for performing this minimization efficiently—describing how to do so in a number of distinct search problems in various classes of length spaces.

6.2 Possible Extensions

Amongst the myriad ways in which this research could be extended and improved upon, we believe the following to be the most fruitful.

- Foremost, theoretical characterization of the index policy, to the extent possible, would be very valuable, as would be a formal proof of the impossibility of such characterization.
- There could be better collaboration in multi-agent policies. We don't doubt that AUCTIONINGNETWORKINDEX can be improved upon.
- We do not address the problem of enumerating paths in high-dimensional spaces. We suppose that the space of paths can be searched *incrementally* to maintain tractability.
- Section 5.7.3 supposes that the index policy can still be applied to multi-class and preemptive service although its computation would require significant modification.
- The search policies discussed here could be integrated into a more general path planning policy.

We elaborate on the last point: Given a policy for search, path planning can sometimes be dramatically simplified as was the case in the example of aiming off.

Suppose that there *is* information available in the environment in the form of features. That is, if the agent detects a feature it is able improve it's state estimate. An irreducibly difficult problem is judging whether this improvement is worth the cost of trying to detect the feature. But given a search strategy, we can attempt to make some headway.

Problem 6.2.1 (Challenge Problem). *Suppose that there are n distinct features $q_i \subset \mathcal{A}$ in which the agent can detect presence, with q_0 being the goal G. Assume that this measurement is noiseless, but suppose that the agent's dynamics are subject to noise. The problem is to find a control policy that takes the agent from an initial location distribution μ, to the goal set G in minimum expected time.*

An initial proposal

Let $U(q_i)$ denote the uniform distribution over the *boundary* of q_i and let μ indicate the robot's position distribution. For each $\phi \in \{U(q_i)\forall i\} \cup \{\mu\}$ and $S \in \{q_i \forall i\}$ compute the expected search time, $d_T(\phi, S)$, from ϕ to S according to the index policy.

One could form a completely-connected, weighted, directed graph $\langle \mathcal{V}, \mathcal{E}, \mathcal{L} \rangle$ with vertices corresponding to the the features (plus the agent's current state) and edge weights corresponding to d_T. At each moment the agent can recompute the edge weights corresponding to $d_T(\mu, q)$ and then search for the next feature in the shortest path to the goal in this graph.

A direction for extension

The important assumption is that when the agent exits feature q, $U(q)$ is a good approximation of μ. This is clearly not a sound assumption in general. This precludes a number of intelligent strategies, like wall-following, which use a feature to hone the state estimate.

The most exciting direction for extension, we feel, incorporates the index policy as an atomic action or "behavior" in a path planning methodology such as that in [81], that *does* attempt to solve Bellman's Equation, and *is* capable, at least in principle, of estimating the value of information. The index policy could add valuable *abstractions*—for instance making the decision tree much shorter. At an intuitive level, the policy would replace long sequences of conditional motion controls with components such as "Search for feature q_i until you find it or the variance of the state estimate exceeds σ^2."

We believe that such an approach should be able to directly handle the following scenario, without resorting to the artifice of Section 2.3. Consider a two wheeled robot whose motion has much greater directional accuracy than angular accuracy. If the robot is trying to reach a goal in minimum expected time, it might be preferable to aim to one side of the goal, travel the correct distance, and then turn 90 degrees to search for the goal. When including search as an atomic action, the two-stage nature of the problem can fall out: the robot should get close, and then it should search.

The fundamental difficulty with such an approach is the one which we have repeatedly pointed out: it is difficult to know *a priori* what constitutes an adequate description of the state distribution for effective planning. Although the index policy might make it tractable to use a higher-dimensional description, it does not address this issue. We suspect that a *general* algorithm for path planning in the same sense as this dissertation presents a "general" algorithm for search could be a white whale.

6.3 Contributions

The major contribution of this dissertation is to develop a general index heuristic for search and show its applicability to a host of problems.

We make the following secondary contributions.

- We prove the Whittle Indexability of the CPP.
- We characterize the practice of aiming off.
- We unite a several problem formulations as MWLCPs.
- We improve on the state of the art in the PPP for large sensor radii.

- We extend existing light-load DTRP partitioning theory to length spaces.
- We extend existing heavy-load DTRP theory to length spaces and complex, *e.g.*, fractal, measures.
- We present modified Nearest Neighbor Algorithms that are Heavy-Load Constant Factor for the DTRP.
- We demonstrate improved performance on the uniform DTRP for the intermediate-load regime for load-factors greater than about 0.5.
- We present the DSRP as an interesting and relevant problem formulation.

We spent the majority of this dissertation discussing a small number of relatively easy special cases. We did this because these are the problems for which there are the most general answers against which we can compare our own. Each comparison, on its own, was unimpressive. When taken together, however, we believe they constitute something remarkable. Consider that essentially the same algorithm produced Table 2.3 and Figure 4-6. In the former we approximate the solution to an elaborate system of differential equation and in latter we approximate the solution to an elaborate geometric optimization.

Only in Section 2.3 when we characterized aiming off and Section 5.8 when we described a realistic scenario did we use our policy to do anything new. But it is here the significance of this dissertation lies: the set of problems for which this work represents a first step. Not only is this practically significant, it also represents a point of comparison against which future work may be judged. A common complaint is that there are no *benchmarks* for the sort of continuous problems covered in this dissertation. Although the fundamental problem presented by the size of the problem space remains, there is now a benchmark algorithm, at least.

Appendix A

Algorithms

Throughout this section we will talk about transformations of geometric networks, *e.g.*, adding nodes or breaking edges. This entails some bookkeeping to maintain the association between the networks. To avoid distracting the reader we will keep this bookkeeping implicit. For completeness, we provide one example in which we provide this bookkeeping. This example is Algorithm NETTOTREE, which is otherwise very simple.

```
Data: Tree-shaped geometric network $\mathcal{A} = \langle \mathcal{V}, \mathcal{E}, \mathcal{L} \rangle$, probability measure $\mu$ on
      $\mathcal{A}$, root vertex $a_0$.
We assume that the agent location is the root and that edges are oriented such
that $(e, 0)$ corresponds to the parent.
Result: A connected sub-region containing $a_0$ maximizing Equation 3.3.
If Equation 3.3 is maximized by the limit of a sequence converging to a tree
containing only the root we return the tree $\{(e, 0)\}$ where $e$ is the edge on
which the limit occurs.

$\mathcal{T}, r, c \leftarrow$ DENSESTSUBTREEHELP($a_0$,0,0,$\mu$)
return $\mathcal{T}$
```

Algorithm DensestSubtree: Finds the densest connected sub-tree containing the root.

Theorem A.0.1. *Algorithm* DENSESTSUBTREE *solves the DSP in polynomial time.*

Proof. It is easy to see that DENSESTSUBTREEHELP has polynomial complexity—we only prove correctness. Assume by way of contradiction that Algorithm DENSEST-SUBTREE returns $\mathcal{T}$ when the optimum is $\mathcal{T}'$, distinct. Recall the function γ from

```
Data: tree $\mathcal{T}$, root node $n$, reward $r$, cost $c$, reward density $\mu$
Result: $\langle \mathcal{T}', r', c' \rangle$ A connected sub-region $\mathcal{S}$,$r' = \mathbb{E}_{\mu}[r(\mathcal{S})] + r$,$c' = \mathcal{L}(\mathcal{S}) + c$
        minimizing $c'/r'$
queue $\leftarrow \emptyset$
for $e \in \delta_o(n)$ do
3   $\alpha \leftarrow \min_{d \in [0,1]} \frac{c + \alpha\mathcal{L}(e)}{r + \mu(\bigcup_{x \in [0,d]} (e,x))}$
    $\mathcal{T}_1 \leftarrow \{(e,x) \text{ s.t. } x \in [0, \alpha]\}$
    $r_1 \leftarrow r + \int_0^d d\mu((e,x)), \quad c_1 \leftarrow c + \alpha\mathcal{L}(e)$
    $\mathcal{T}_2, r_2, c_2 \leftarrow$
      DENSESTSUBTREEHELP$((e,1), r + \mu(e), c + \mathcal{L}(e), \mu)$
8   if $c_1/r_1 > c_2/r_2$ then
        INSERT (queue,$(c_1/r_1, \langle \mathcal{T}_1, r_1, c_1 \rangle)$)
    else
        INSERT (queue,$(c_2/r_2, \langle \mathcal{T}_2, r_2, c_2 \rangle)$)
    end
end
$\mathcal{T}' \leftarrow \{n\}$
$r', c' \leftarrow r, c$
while $queue \neq \emptyset$ do
    $\mathcal{T}_S, r_S, c_S \leftarrow$ EXTRACTMIN (queue)
    if $c_S/r_S \geq c'/r'$ then
19      return $\langle \mathcal{T}', r', c' \rangle$
    else
        $\mathcal{T}' \leftarrow \mathcal{T}' \cup \mathcal{T}_S$
        $r' \leftarrow r' + r_S - r, \quad c' \leftarrow c' + c_S - c$
    end
end
return $\langle \mathcal{T}', r', c' \rangle$
```
Algorithm DensestSubtreeHelp: A recursive helper function for DENSESTSUBTREE

Equation 3.4 and note that we have $\gamma(\mathcal{T}) > \gamma(\mathcal{T}')$ by assumption. Let x be a point where they diverge, *i.e.* x is the root of two non-intersection trees, $\mathcal{T}_x \in \mathcal{T}$ and $\mathcal{T}'_x \in \mathcal{T}'$. Note that we have $\gamma(\mathcal{T}'_x) \leq \gamma(\mathcal{T}')$ by the optimality of $\mathcal{T}'$.

Suppose that x is a node and consider the point at which Algorithm DENSESTSUBTREE returns from the recursion at which x is the root. At this point we have $c'/r' \geq \gamma(\mathcal{T})$, but there is a sub-tree, $\mathcal{T}'_x$, in the queue with $c_S/r_S < c'/r'$. Therefore this tree would be added before returning and it is impossible for Algorithm DENSESTSUBTREE to return a tree containing x but not containing $\mathcal{T}'_x$.

Suppose that x is not a node, so that $\mathcal{T}_x = \mathcal{T}_1$. We break this into two cases, depending on whether $\mathcal{T}'_x$ contains a node or not. Suppose it does not. This contradicts that $\mathcal{T}_1$ is the minimizer found in Line 3.

If it does contain a node, $\mathcal{T}_2 = \mathcal{T}_x \cup \mathcal{T}'_x$ will be returned in Line 19 and, as we will show, will be preferred in Line 8. Let c'_i, r'_i denote $c'_i - c, r'_i - r$.

We have $c/r \geq c'_1/r'_1$, $c/r \geq c'_2/r'_2$, and $(c_1)/(r_1) \geq \gamma(\mathcal{T}) \geq c'_2/r'_2$. Therefore

$$\frac{c_1}{r_1} \geq \frac{c + c'_1 + c'_2}{r + r'_1 + r'_2} = \frac{c_2}{r_2}$$

This implies that $\mathcal{T}_2$ would be added instead of $\mathcal{T}_1$ contradicting our assumption. □

Lemma A.0.2. *The solution to the DSP has a unique first edge, therefore it can be interpreted unambiguously as a policy.*

Proof. At the root node, when the first sub-tree is popped from the queue, ret and cost are both zero. Let r, c be the reward and cost of this first sub-tree, with c/r being minimum over such sub-trees. When any subsequent sub-tree is popped with r', c', we have $c/r \leq (c + c')/(r + r')$ and therefore it will not be added. □

We should point out that although $\{(e, 0) \text{ s.t. } e \in \delta(a_0)\}$ are not distinct elements of $\mathcal{A}$, a distinction is made between them when interpreting the output from Algorithm DENSESTSUBTREE. The output $\{(e, 0)\}$ is to be interpreted as the case in which the sub-tree that minimizes Equation 3.3 does not exist, rather it is limit of sequences of trees $\lim_{t \to 0} \bigcup_{\tau \in [0,t]} (e, \tau)$.

```
Data: A geometric network $\mathcal{A}$, measure $\mu$, starting location $x$
Result: A dense sub-tree containing $x$
$\mathcal{A}' \leftarrow$ NetToTree($\mathcal{A}, x$)
return DensestSubtree($\mathcal{A}', \mu, x$)
```

Algorithm DenseSubnet: Finds a dense subnet containing x by creating a tree and calling DensestSubtree

```
Data: geometric network $G$, spatial measure $\mu$, tree $\mathcal{T} = \langle \mathcal{V}_T, \mathcal{E}_T \rangle \subseteq G$ with
      root $x$
Result: A new tree $\mathcal{T}'$ also rooted at $x$ with equal or lower weight, with no
        branchings outside of $\mathcal{T}$
$\mathcal{P}_{10} \leftarrow$the path in $\mathcal{T}$ from the first branching (or only leaf) of $\mathcal{T}$ to $x$
$t^* \leftarrow$ Equation 3.6
$\gamma^* \leftarrow \mathcal{L}(\mathcal{T})/\mu(\mathcal{T})$
$G' \leftarrow G$ with vertex $v_1$ added at $\mathcal{P}_{10}(t^*)$
$\mathcal{L}'(e) \leftarrow \mathcal{L}(e) - \gamma^* \mu(e)\ \forall e \in \mathcal{E}'$
return ReplacePrefix($G', \mathcal{T}, x, v_1$)
```

Algorithm PrefixReplacement: Tries to improve on the solution returned by DenseSubnet

```
Data: A weighted graph, $G = \langle \mathcal{V}, \mathcal{E}, \mathcal{L} \rangle$, a subtree $\mathcal{T} = \langle \mathcal{V}_T, \mathcal{E}_T \rangle$ rooted at $v_0$
      and containing vertex $v_1$
Result: A new tree $\mathcal{T}'$ also rooted at $v_0$ with equal or lower weight, with no
        branchings outside of $\mathcal{T}$
$G' \leftarrow G$
$\mathcal{P}_{01} \leftarrow$ the path from $v_0$ to $v_1$ in $\mathcal{T}$
$\mathcal{T}' \leftarrow \mathcal{T} \setminus \mathcal{P}_{01}$
Modify $G'$ by collapsing $\mathcal{T}'$ in $G'$ to a single node $v_1^*$
repeat
    Run the Bellman-Ford algorithm on $G'$
    if There is a negative cycle $C$ then
        if $v_0 \in C$ then
            return $C \setminus \{\arg\max_{e \in \delta(v_0) \cap C} \mathcal{L}(e)\}$
        else
            $\mathcal{E}' \leftarrow \mathcal{E}' \setminus \{\arg\max_{e \in C} \mathcal{L}(e)\}$
        end
    else
        return $\mathcal{T}' \cup$ the shortest path from $v_0$ to $v_1^*$ in $G'$
    end
until Forever
```

Algorithm ReplacePrefix: Tries to replace a specific prefix of the plan.

```
Data: A geometric network $G$, measure $\mu$ over goal location, agent location $x$
Result: A direction
$G' \leftarrow G$ with $x$ as a node
$\mathcal{T}_s \leftarrow$ DenseSubnet($G', \mu, x$)
$\mathcal{T}_s' \leftarrow$ PrefixReplacement($G', \mu, \mathcal{T}_s, x$)
return $\mathcal{T}_s'$
```

Algorithm NetworkIndexPolicy: The simplified index policy.

```
Data: region $\mathcal{A}$, belief $\mathcal{I}$ over the region, $m$ starting locations $s$, single-agent
      planning algorithm PLANNER
Result: A path for each agent.
$\mathcal{I}' \leftarrow \mathcal{I}$
$u \leftarrow \{1, \ldots, m\}$                           // set of unassigned agent indices
$\mathcal{P}[i] = \emptyset$ for $i \in u$                                      // set of paths
$\gamma^*[i] = \infty$ for $i \in u$
while $u \neq \emptyset$ do
    for $i \in u$ do
        $\mathcal{P}[i] \leftarrow$ PLANNER($\mathcal{A}, \mathcal{I}', a_0$)
        $\gamma^*[i] \leftarrow \gamma(\mathcal{P}[i])$                          // Equation 3.3
    end
    $i_a = \arg\min_{i \in u} \gamma^*[i]$
    $u \leftarrow u \setminus \{i_a\}$
12  $\mathcal{I}' \leftarrow$ zeroOut($\mathcal{I}', \mathcal{P}[i_a]$)
end
return $\mathcal{P}$
```

Algorithm auctionPlanner: Creates a multi-agent planner out of a single-agent planner by repeatedly running winner-take-all auctions.

```
Data: region $\mathcal{A}$, belief $\mathcal{I}$, agent locations $s$
Result: multi-agent plan
return auctionPlanner($\mathcal{A}, \mathcal{I}, s$, NetworkIndexPolicy)
```

Algorithm AuctioningNetworkIndex: A multi-agent polynomial-time index policy for networks

```
Data: Region $\mathcal{A}$, belief $\mathcal{I}$, agent locations, $s$, a set of planners, $\Pi$
Result: a plan
$\mu \leftarrow$ the normalized spatial measure consistent with $\mathcal{I}$
$\gamma_{\min} \leftarrow \infty$
for PLANNER $\in \Pi$ do
    $\mathcal{P} \leftarrow$ PLANNER$(s, \mathcal{I})$
    $\gamma^* \leftarrow \frac{\sum_i \mu(\mathcal{P}_i)\gamma(\mathcal{P}_i)}{\mu(\mathcal{P})}$                    // Equation 3.3
    if $\gamma^* < \gamma_{min}$ then
        $\mathcal{P} \leftarrow$ PLANNER$(\mathcal{A}, \mathcal{I}, s)$
        $\mathcal{P}^* \leftarrow \mathcal{P}$,   $\gamma_{\min} \leftarrow \gamma^*$
    end
end
return $\mathcal{P}^*$
```
Algorithm MultiPlanner: Compares multiple planning algorithms using the index function

```
Data: Region $\mathcal{A}$, belief $\mathcal{I} = \langle \theta(t), Z(t), \mu(t) \rangle$, agent locations $s$, multi-agent
      observed planner OBS : $(\theta, s) \rightarrow \mathcal{P}$, multi-agent search planner,
      UNOBS : $(\mathcal{I}, s) \rightarrow \mathcal{P}$
Result: a multi-agent plan $\mathcal{P}$
$\mathcal{P} \leftarrow$ OBS$(\theta, s)$
$I^u \leftarrow \{i \in 1, \ldots, m$ s.t. $plan[i] = \phi\}$
$\mathcal{I}' \leftarrow$ ZEROOUT$(\mathcal{I}, \mathcal{P})$
$s^u \leftarrow \{x_i \, \forall i \in I^u\}$ $\mathcal{P}^u \leftarrow$ UNOBS$(\mathcal{I}', s^u)$
$\mathcal{P}[I^u[j]] \leftarrow \mathcal{P}^u[j] \; \forall j \in |I^u|$ return $\mathcal{P}$
```
Algorithm ObservedFirstMultiPlanner: a multi-agent planner that first assigns observed requests according to algorithm OBS and then runs a finds plans for agents which are not assigned requests according to algorithm UNOBS

With multiple agents, the natural analogue of the Nearest Neighbor Algorithm is the minimum matching algorithm. For the observable case, we were able to occasionally improve on the winner-take-all nature of Algorithm AUCTIONPLANNER by combining it into a multi-planner (Algorithm MULTIPLANNER) with Algorithm MATCHING.

```
Data: agent locations $s$, requests $\theta$
Result: a plan assigning each agent to at most one request
return the minimum matching in the bipartite graph formed by connecting
    each agent to each request with an edge whose length is the distance between
    them
```

Algorithm MinMatchingPlanner: An observed planner that computes a minimum matching between agents and requests

```
Data: agent locations $s$, belief $\mathcal{I} = \langle \theta(t), Z(t), \mu(t) \rangle$
Result: a multi-agent plan $\mathcal{P}$
return OBSERVEDFIRSTMULTIPLANNER($\mathcal{A}, s, \mathcal{I}$, MINMATCHINGPLANNER,
            AUCTIONINGNETWORKINDEX)
```

Algorithm Matching: a multi-agent planner that first assigns observed requests via a minimum matching and then uses the auctioning index policy for the remaining agents

```
Data: region $\mathcal{A}$, belief $\langle \theta(t), Z(t), \mu(t) \rangle$,agent locations $s$
Result: a multi-agent plan $\mathcal{P}$
return MULTIPLANNER($s, \mathcal{I}$,
            [MATCHING, AUCTIONINGNETWORKINDEX])
```

Algorithm ConsiderMatchingPlanner: a planning algorithm consisting of two planners, MATCHING and AUCTIONINGNETWORKINDEX being compared using MULTIPLANNER

```
Data: metric space $\mathcal{A}, d(\cdot,\cdot)$, geometric network $G \subseteq \mathcal{A}$, agent locations $s$,
      parameters $\delta_0, \delta_1$
Result: A geometric network approximating the connectivity of $G$
$G' \leftarrow G$;
$\mathcal{V}^+ \leftarrow$ a $\delta_0$ discretization of the edges;
$\mathcal{E}^+ \leftarrow \{(v_0, v_1)$ s.t. $v_0 \in \mathcal{V}^+ \cup \mathcal{V}, v_1 \in \mathcal{V}^+, v_0 \neq v_1\} \setminus \mathcal{E}$;
sort $\mathcal{E}^+$ by increasing length;
for $(v_0, v_1) \in \mathcal{E}^+$ do
    $d_{\mathcal{A}} \leftarrow d(v_0, v_1)_{\mathcal{A}}, d_{G'} \leftarrow d(v_0, v_1)_{G'}$;
    if $d_{G'} - d_{\mathcal{A}} > \delta_0$ and $(d_{G'} - d_{\mathcal{A}})/d_{\mathcal{A}} > \delta_1$ then
        $G' \leftarrow G$ with an edge between $v_0$ and $v_1$ (adding vertices if necessary);
    end
end
return $G'$
```
Algorithm MetricEmbedding: An algorithm for creating a one dimensional region approximating the connectivity of a higher dimensional region.

```
Data: Metric space $\mathcal{A}$, belief $\mathcal{I}$ spatially supported by network $G$, agent
      locations $s$
Result: A plan for each agent
$\mathcal{A}' \leftarrow$ METRICEMBEDDING$(\mathcal{A}, G, s)$
return CONSIDERMATCHINGPLANNER$(\mathcal{A}', \mathcal{I}, s)$
```
Algorithm Implementation: The planner used in the flight-tests described in Section 5.8

```
Data: Geometric network $\mathcal{A} = \langle \mathcal{V}, \mathcal{E}, \mathcal{L} \rangle$ with measure $\mu$, price $\gamma$
Data: A new equivalent geometric network in which the minimum cost path
      (at subsidy $\gamma$) is a circuit
$\mathcal{A}' \leftarrow \mathcal{A}$
for $e \in \mathcal{E}$ do
    $\ell \leftarrow \mathcal{L}(e)$
    $\mathcal{P}_e^+ \leftarrow \Lambda(t) : (e, t/\ell)$
    $\mathcal{P}_e^- \leftarrow \Lambda(t) : (e, 1 - t/\ell)$
    $t^+ \leftarrow \max_{t \in [0,\ell]} 2t\mathcal{L}(e) - \mu(\mathcal{P}_e^+([0,t]))$
    $t^- \leftarrow \max_{t \in [0,\ell]} 2(1-t)\mathcal{L}(e) - \mu(\mathcal{P}_e^-([0,t]))$
    for $a \in \{\mathcal{P}_e^+(t^+), \mathcal{P}_e^-(t^-)\}$ do
        if $a \notin \mathcal{V}$ then
            $\mathcal{A}' \leftarrow$ ADDNODE$(a)$
        end
    end
end
return $\mathcal{A}'$
```
Algorithm NodeAdding: An algorithm for adding vertices to a geometric network such that the minimum ratio circuit only changes directions at vertices

```
Data: network $G = \langle V, E, d \rangle$ and root $a_0$.
Result: a tree-shaped network $\mathcal{T} = \langle V', E', d' \rangle$ and a function $f : \mathcal{T} \to G$
        preserving the distance from the root to each location
$V' \leftarrow V$
MST $\leftarrow$ MinimumSpanningTree$(V, E, w)$
$E' \leftarrow$ MST
for $v \in V$ do
    $f(v) = v$
for $e \in MST$ do
    $f(e, d) \leftarrow (e, d) \forall d \in (0, 1)$
for $\langle e, n1, n2 \rangle \in E \setminus MST$ do
    $le \leftarrow w(e), d1 \leftarrow d((,n)1, root), d2 \leftarrow d((,n)2, root)$
    if $d2 = le + d1$ then
        define new labels $v', e'$
        $V' \leftarrow V' \cup \{v'\}, E' \leftarrow E' \cup \{\langle e', n1, n2 \rangle\}, w'(e') \leftarrow le$
        $f(v') \leftarrow n2, f(e', d) \leftarrow (e, d) \forall d \in (0, 1)$
    else if $d1 = le + d2$ then
        define new labels $v', e'$
        $V' \leftarrow V' \cup \{v'\}, E' \leftarrow E' \cup \{\langle e', n2, n1 \rangle\}, w'(e') \leftarrow le$
        $f(v') undefined, f(e', d) \leftarrow (e, d) \forall d \in (0, 1)$
    else
        define new labels $v1', v2', e1', e2'$
        $\alpha \leftarrow (d2 - d1)/(2le)$
        $V' \leftarrow V' \cup \{v1', v2'\}, E' \leftarrow E' \cup \{\langle e1', n1, v1' \rangle, \langle e2', v2', n2 \rangle\}$
        $w'(e1') \leftarrow \alpha le, w'(e2') \leftarrow (1 - \alpha) le$
        $f(v1') \leftarrow (e, \alpha), f(v2') \leftarrow (e, \alpha)$ (or undefined)
        $f(e1', d) \leftarrow (e, \alpha d) \forall d \in (0, 1)$
        $f(e2', d) \leftarrow (e, \alpha + (1 - \alpha) d) \forall d \in (0, 1)$
return $\mathcal{T}, f$
```

Algorithm NetToTree: An algorithm for creating an "equivalent" tree shaped network that preserves the length of shortest paths from a root node. Includes the details associated with the association between the tree and the original network.

```
Data: A weighted graph $G$, root $r$
Result: a spanning path of $G$, starting at $r$, with low latency
Find the tree $T$ of minimum weight, spanning $G$, in which $r$ has odd degree.;
Find the minimum matching $M_1$ of odd-degree nodes in $T$.;
Add $M_1$ to $T$ to make Eulerian graph $E_1$.;
Find any traversal $S_1$ of $E_1$ and $S_1^r$ its reverse;
/* We use the greedy heuristic, selecting the shortest edge when
   facing a choice                                                  */
Find $M_2$ the minimum matching of odd-degree nodes except $r$.;
Add $M_2$ to $T$ to make a graph $E_2$ with exactly two odd degree nodes.;
Find any traversal $S_2$ of $E_2$;
// same choice applies
return the path among $S_1, S_1^r$, and $S_2$ with minimum latency
```

Algorithm ModKoutsoupias: Our modification of the algorithm from [60]

Bibliography

[1] S. Albers. Online algorithms: a survey. *Mathematical Programming*, 97(1):3–26, 2003. 30, 31

[2] L. Allulli, G. Ausiello, and L. Laura. On the power of look-ahead in on-line vehicle routing problems. In *Lecture notes in computer science*, volume 3595, page 728. Springer, 2005. 122

[3] S. Alpern and S. Gal. *The theory of search games and rendezvous.* Springer, 2003. 30, 31, 47, 48, 50, 56, 105

[4] Eitan Altman and Sergei Foss. Polling on a graph with general arrival and service time distribution. Technical report, INRIA, 1993. 122

[5] P. S. Ansell, K. D. Glazebrook, J. Niño-Mora, and M. O'Keeffe. Whittle's index policy for a multi-class queueing system with convex holding costs. *Mathematical Methods of Operations Research*, 57(1):21–39, 2003. 25, 44

[6] A. Arsie and E. Frazzoli. Efficient routing of multiple vehicles with no explicit communications. *International Journal of Robust and Nonlinear Control*, 18(2), 2008. 141

[7] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Algorithms for the On-Line travelling salesman. *Algorithmica*, 29(4):560–581, 2001. 122

[8] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching with uncertainty extended abstract. *first Scandinavian workshop on algorithm theory*, pages 176–189, 1988. 30

[9] R. A. Baeza-Yates, J. C. Culberson, and G. J. E. Rawlins. Searching in the plane. *Information and Computation*, 106(2):234–252, October 1993. 45

[10] J. S. Banks and R. K. Sundaram. Switching costs and the gittins index. *Econometrica*, 62(3):687–94, 1994. 24

[11] A. Beck and M. Beck. The revenge of the linear search problem. *SIAM Journal on Control and Optimization*, 30:112, 1992. 28

[12] Anatole Beck. More on the linear search problem. *Israel Journal of Mathematics*, 3(2):61–70, 1965. 28

[13] Anatole Beck and Micah Beck. Son of the linear search problem. *Israel Journal of Mathematics*, 48(2-3):109–122, 1984. 28, 29, 32, 47, 48, 56

[14] Anatole Beck and D. J. Newman. Yet more on the linear search problem. *Israel Journal of Mathematics*, 8(4):419–429, 1970. 28

[15] R. Bellman and R. Kalaba. On adaptive control processes. *Automatic Control, IRE Transactions on*, 4(2):1–9, 1959. 20

[16] Richard Bellman. Minimization problem. *Bulletin of the American Mathematical Society*, 2(3):270, 1956. 87

[17] Richard Bellman. An optimal search. *SIAM Review*, 5(3):274, 1963. 28

[18] D. Bertsimas and J. Niño-Mora. Restless bandits, linear programming relaxations, and a primal-dual index heuristic. *Operations Research*, pages 80–90, 2000. 25, 44

[19] D. J. Bertsimas, P. Jaillet, and A. R. Odoni. A priori optimization. *Operations Research*, pages 1019–1033, 1990. 120

[20] D. J. Bertsimas and D. Simchi-Levi. A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations Research*, pages 286–304, 1996. 121

[21] D. J. Bertsimas and G. Van Ryzin. A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research*, pages 601–615, 1991. 19, 116, 120, 152, 154

[22] D. J. Bertsimas and G. Van Ryzin. Stochastic and dynamic vehicle routing in the euclidean plane with multiple capacitated vehicles. *Operations Research*, pages 60–76, 1993. 121

[23] D. J. Bertsimas and G. Van Ryzin. Stochastic and dynamic vehicle routing with general demand and interarrival time distributions. *Advances in applied probability*, pages 947–978, 1993. 19, 121, 144, 145

[24] A. S. Besicovitch. On arcs that cannot be covered by an open equilateral triangle of side 1. *Math Gazette*, 49:286–288, 1965. 89, 102

[25] T.F. Bruce and J.B. Robertson. A survey of the linear-search problem. *The Mathematical Scientist*, 13, 1988. 29, 30, 56

[26] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. Paths, trees, and minimum latency tours. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 36–45, 2003. 76, 78

[27] H. Choset. Coverage for robotics–A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1):113–126, 2001. 105

[28] David Chudnovsky and Gregory Chudnovsky, editors. *Search Theory.* Marcel Dekker, inc., 1987. 104

[29] K. L Clarkson. *Methods for Learning and Vision: Theory and Practice*, chapter Nearest-neighbor searching and metric space dimensions, pages 15–59. MIT Press, 2006. 142, 143

[30] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *Robotics and Automation, IEEE Transactions on*, 20(2):243–255, 2004. 123

[31] John Davis. *Seaman's Secrets.* Gartrude Dawson, 8th edition, 1657. 56

[32] R. D. Van der Mei and E. M. M. Winands. Polling models with renewal arrivals: a new method to derive heavy-traffic asymptotics. *Performance Evaluation*, 64(9-12):1029–1040, 2007. 122

[33] I. Eliazar. From polling to snowplowing. *Queueing Systems*, 51(1):115–133, 2005. 122

[34] J. J. Enright. *Efficient routing of Multi-Vehicle Systems.* PhD thesis, University of California at Santa Barbara, 2008. 116, 122, 124, 151

[35] S. R Finch and J. E Wetzel. Lost in a forest. *American Mathematical Monthly*, 111(8):645–654, 2004. 88, 89

[36] Steven R Finch and John A Shonder. Lost at sea, 2004. 90, 98, 100

[37] Rudolf Fleischer, Tom Kamphans, Rolf Klein, Elmar Langetepe, and Gerhard Trippen. Competitive online approximation of the optimal search ratio. *The 12th Annual European Symposium on Algorithms*, pages 335—346, 2004. 31

[38] Wallace Franck. An optimal search problem. *SIAM Review*, 7(4):503–512, October 1965. 28, 53, 54

[39] E. Frazzoli and F. Bullo. Decentralized algorithms for vehicle routing in a stochastic time-varying environment. *Atlantis*, 2004. 122, 141

[40] Shmuel Gal. *Search Games.* Academic Press, 1980. 33, 69, 105

[41] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 2003. 121

[42] H. Gimbert. Pure stationary optimal strategies in markov decision processes. *24th Annual Symposium on Theoretical Aspects of Computer Science*, pages 200–211, 2007. 21

[43] J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B*, 41(2):148–177, 1979. 23

[44] K. D. Glazebrook, H. M. Mitchell, and P. S. Ansell. Index policies for the maintenance of a collection of machines by a set of repairmen. *European Journal of Operational Research*, 165(1):267–284, August 2005. 25, 36, 44

[45] K. D. Glazebrook, D. Ruiz-Hernandez, and C. Kirkbride. Some indexable families of restless bandit problems. *Advances in Applied Probability*, 38(3):643, 2006. 34, 35

[46] B. Gluss. An alternative solution to the "lost at sea" problem. *Naval Research Logistics Quarterly*, 8(1):117–122, 1961. 90, 99, 100, 101, 102

[47] B. Gluss. The minimax path in a search for a circle in a plane. *Naval Research Logistics Quarterly*, 8:357–360, 1961. 89, 99

[48] V. Huynh, J. J. Enright, and E. Frazzoli. Persistent patrol and detection with limited on-board sensors. In *IEEE Conf. on Decision and Control*, 2010. 124

[49] J. R. Isbell. An optimal search pattern. *Naval Research Logistics Quarterly*, 4(4):357–359, 1957. 88, 89, 99, 100

[50] P. Jaillet. À priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, pages 929–936, 1988. 120

[51] P. Jaillet and M. R. Wagner. Online routing problems: Value of advanced information as improved competitive ratios. *Transportation Science*, 40(2):200–210, 2006. 122

[52] P. Jaillet and M. R. Wagner. *The Vehicle Routing Problem: Latest Advances and New Challenges*, chapter Online Vehicle Routing Problems: A Survey. Operations Research Computer Science Interfaces. Springer, 2008. 30, 122

[53] Antoine Jézéquel. *Probabilistic vehicle routing problems.* PhD thesis, Massachusetts Institute of Technology, 1985. 120

[54] D. S Johnson and L. A McGeoch. The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, pages 215–310, 1997. 19

[55] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99—134, 1998. 21

[56] M. Y. Kao and M. L. Littman. Algorithms for informed cows. In *AAAI-97 Workshop on On-Line Search*, 1997. 29, 31, 32

[57] W. Kern and G. Woeginger. Quadratic programming and combinatorial minimum weight product problems. *Mathematical programming*, 110(3):641–649, 2007. 72

[58] Kingston. 2010. Personal Communication. 159

[59] Bernard Koopman. Review of theory of optimal search. *SIAM Review*, 19(2):361, 1977. 104

[60] E. Koutsoupias, C. Papadimitriou, and M. Yannakakis. Searching a fixed graph. *Automata, Languages and Programming*, pages 280–289, 1996. 77, 179

[61] G. Kozma, Z. Lotker, G. Stupp, et al. The minimal spanning tree and the upper box dimension. *Proceedings of the American Mathematical Society*, 134(4):1183–1188, 2006. 147

[62] A. Larsen, O. B. G. Madsen, and M. M. Solomon. *The Vehicle Routing Problem: Latest Advances and New Challenges*, chapter Recent Developments in Dynamic Vehicle Routing Systems. Operations Research Computer Science Interfaces. Springer, 2008. 33, 121

[63] A. Larsen, O.B.G Madsen, and M. M. Solomon. Partially dynamic vehicle routing-models and algorithms. *Journal of the Operational research Society*, pages 637–646, 2002. 121

[64] R. C. Larson and A. R. Odoni. *Urban Operations Research*. Prentice Hall, New Jersey, 1981. out of print. Available at http://web.mit.edu/urban_or_book/www/book/. 119, 158

[65] John D. C. Little. A proof for the queuing formula: L= w. *Operations Research*, 9(3):383–387, June 1961. 141

[66] K. Lund, O.B.G Madsen, and J.M. Rygaard. Vehicle routing problems with varying degrees of dynamism. Technical report, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1996. 121

[67] Mark Mears. ICE-T program description. AFRL Public Clearance 88ABW-2010-3796, July 2010. 159

[68] Nimrod Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4(4):414–424, November 1979. 72

[69] Z. A. Melzak. *Companion to Concrete Mathematics*. Wiley, 1973. 99

[70] J. Le Ny, M. A. Dahleh, E. Feron, and E. Frazzoli. Continuous path planning for a data harvesting mobile server. In *47th IEEE Conference on Decision and Control, 2008. CDC 2008*, pages 1489–1494, 2008. 122

[71] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, pages 441–450, 1987. 21

[72] J. D. Papastavrou. A stochastic and dynamic routing policy using branching processes with state dependent immigration. *European Journal of Operational Research*, 95(1):167–177, 1996. 19, 121, 123, 152, 154

[73] Don Paul. *The Green Beret's Compass Course.* Path Finder Publications (CA), 10th edition, January 2006. 64

[74] M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler. A stochastic and dynamic vehicle routing problem with time windows and customer impatience. *ACM Mobile Networks and Applications*, 2008. 122, 123

[75] M. Pavone, E. Frazzoli, and F. Bullo. Distributed policies for equitable partitioning: Theory and applications. In *IEEE Conference on Decision and Control*, 2008. 151

[76] M. Pavone, E. Frazzoli, and F. Bullo. Distributed and adaptive algorithms for vehicle routing in a stochastic and dynamic environment. *IEEE Transactions on Automatic Control*, 2009. 19

[77] M. Pavone, S. Smith, F. Bullo, and E. Frazzoli. Dynamic multivehicle routing with multiple classes of demands. In *American Control Conference*, 2009. 122

[78] G. Poole and J. Gerriets. Minimum covers for arcs of constant length. *Bulletin of the American Mathematical Society*, 79(2), 1973. 88

[79] H. N. Psaraftis. Dynamic vehicle routing problems. *Vehicle routing: Methods and studies*, 16:223–248, 1988. 120

[80] H. N. Psaraftis. Dynamic vehicle routing: Status and prospects. *Annals of Operations Research*, 61(1):143–164, 1995. 121

[81] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In *Robotics and Automation, Proceedings. 1999 IEEE International Conference on*, volume 1, 1999. 23, 57, 166

[82] K. Savla, C. Nehme, T. Temple, and E. Frazzoli. On efficient cooperative strategies between UAVs and humans in a dynamic environment. In *Proc. of the AIAA Conf. on Guidance, Navigation, and Control*, 2008. 159

[83] Barry Schiff. *Proficient Pilot, Volume 1.* Aviation Supplies & Academics, Inc., second edition, September 2001. 11, 55

[84] R. Sitters. The minimum latency problem is NP-hard for weighted trees. *Integer Programming and Combinatorial Optimization*, pages 230–239, 2006. 70

[85] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985. 122

[86] Lawrence Stone. *Theory of Optimal Search.* Academic Press, 1975. 104

[87] U.S. Army, Washington, D.C. *Map Reading and Land Navigation*, FM 3-25.26 edition, July 2001. 64

[88] V. M. Vishnevskii and O. V. Semenova. Mathematical methods to study the polling systems. *Automation and Remote Control*, 67(2):173–220, 2006. 122

[89] A. Washburn. Dynamic programming and the backpacker's linear search problem. *Journal of Computational and Applied Mathematics*, 60(3):357–365, 1995. 29, 48, 50, 56

[90] R. Weber and G. Weiss. On an index policy for restless bandits. *Journal of applied probability*, 27(3):637–648, 1990. 25

[91] J. E. Wetzel. Fits and covers. *Math. Magazine*, 76:349–363, 2003. 103

[92] P. Whittle. Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability*, 25:287–298, 1988. 24, 35

[93] H. Xu. *Optimal policies for stochastic and dynamic vehicle routing problems.* PhD thesis, Massachusetts Institute of Technology, 1995. 121

[94] V. A Zalgaller. A discussion about a question of bellman. *Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov (POMI)*, 299(Geom. i Topol. 8):54–86, 327–328, 1961. 88, 90, 98, 100

Acronyms

ADP	Approximate Dynamic Programming. 19, 22, 26, 113, 124
AFB	Air Force Base. 160, 161
AFRL	Air Force Research Lab. 26, 158
AMDP	Augmented MDP. 22, 57
ATR	Automated Target Recognition. 159
CMASI	Common Mission Automation Services Interface. 158, 159
CNP	Coastal Navigation Problem. 57
CPP	Cow Path Problem. 3, 15, 16, 25, 27, 30–32, 34, 35, 41, 53, 56, 58, 64, 70, 78, 83, 84, 166
CPS	Continuous Polling System. 122
DSP	Densest Sub-Tree Problem. 73, 75, 169, 171
DSRP	Dynamic Search and Repair Problem. 25, 115–117, 127, 129, 151, 152, 159, 163, 167
DTRP	Dynamic Traveling Repairperson Problem. 3, 17, 19, 25, 116, 119, 120, 122, 141, 151, 152, 154, 167
DTSP	Dynamic Traveling Salesman Problem. 120
DVR	Dynamic Vehicle Routing. 119, 120, 122, 123
GPS	Global Positioning System. 55, 120
GSP	Graph Search Problem. 3, 16, 69, **69**
i.i.d.	independent, identically-distributed. 115, 116, 134, 142, 156
ICE-T	Intelligent Control and Evaluation of Teams. 158
IED	improvised explosive device. 115
LFP	Lost in a Forest Problem. 3, 16, 25, 85, 87, 90, **90**, 92, 94, 96, 103, 104
LSP	Line Search Problem. 3, 16, 27–30, 50, 56

MABP	Multi-Armed Bandit Problem. 23, 24
MDP	Markov Decision Problem. 20, **20**, 21, 24, 31–33, 189
MLTP	Minimum Latency Tour Problem. 3, 16, 19, 25, 65, 69, **69**, 70, 76–78, 152
MPPP	Minimum Price Path Problem. **69**, 70, 73, 80
MRCP	Minimum Ratio Circuit Problem. 71, 72, 77
MRPP	Minimum Ratio Path Problem. 71–73
MWLCP	Minimum Weighted Latency Covering Problem. 68, **68**, 69, 70, 85, 104, **104**, 124, 125, 166
PCSP	Prize Collecting Stroll Problem. 77, 78
PCSTP	Prize Collecting Steiner Tree Problem. 76–78
PDE	partial differential equation. 28, 56
POMDP	Partially-Observable Markov Decision Problem. 21, 22, 57, 118
PPP	Persistent Patrol Problem. 3, 17, 116, 119, 123, 124, 127, 151, 152, 166
RBP	Restless Bandit Problem. 24, 35
TS	Talisman Saber Exercise. 159
TSP	Traveling Salesman Problem. 18, 70, 72, 76, 121, 147
UAV	Unmanned Arial Vehicle. 25, 158–160
VRP	Vehicle Routing Problem. 120, 121
VSCS	Vigilant Spirit Control Station. 158–160

Index