

**PROTOCOLS FOR STOCHASTIC SHORTEST PATH
PROBLEMS WITH DYNAMIC LEARNING**

by

Vural Aksakalli

A dissertation submitted to The Johns Hopkins University in conformity with the
requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

March, 2007

© Vural Aksakalli 2007

All rights reserved

Abstract

The research problem considered in this dissertation, in its most broad setting, is a stochastic shortest path problem in the presence of a dynamic learning capability (SDL). Specifically, a spatial arrangement of possible-obstacles needs to be traversed as swiftly as possible, and the status of the obstacles may be disambiguated (at a cost) en route. The central question is to find a protocol that decides what and where to disambiguate so as to minimize the expected length of the traversal. No efficiently computable optimal protocol is known and many similar problems have been proven intractable.

Chapter 1 defines SDL in continuous and discrete settings, and introduces the Random Disambiguation Paths Problem (RDP), a continuous variant of SDL wherein the possible-obstacles are disc-shaped regions in the plane. For any RDP instance, the continuous plane can be approximated by a graph (a lattice), giving rise to the Discrete RDP Problem (DRDP).

Chapter 2 casts SDL as a Markov Decision Process and develops the corresponding Bellman equation. Introduced in this chapter is a new improvement on the well-known AO^* Algorithm, called the BAO^* Algorithm, which employs stronger pruning techniques and substantially shortens the running time needed to find an exact solution to relatively small DRDP instances. The chapter also presents a Partially Observable Markov Decision Process (POMDP) formulation of RDP.

Chapter 3 discusses visibility graphs and introduces the tangent arc graph data structure (TAG). TAG is a new data structure comprised of the topological superimposition of all of the visibility graphs generated by a collection of all subsets of disc-shaped obstacles. Although there are exponentially many such subsets, TAG is polynomially-sized. The chapter then points out that the well-known A^* Algorithm with a slightly stronger admissibility requirement on the heuristic function is equivalent to Dijkstra’s Algorithm under a change of variable.

Chapter 4 introduces the simulated risk disambiguation protocol (SR)—a suboptimal but, effective and efficiently computable algorithm for RDP and DRDP. This protocol initially assumes that all discs are riskily traversable. Then, a chosen “undesirability function” is used (for each such possible traversal) to combine length and risk into a single measure of traversal undesirability. A shortest traversal in this undesirability sense is what the DM traverses until the first ambiguous disc is encountered, at which point a disambiguation is performed and the problem data is updated accordingly. This procedure is iteratively repeated until arrival at the destination.

In Chapter 5, another suboptimal, but very efficiently computable protocol is proposed for RDP and DRDP, called the continually reactivated (CR) disambiguation protocol. The CR protocol is defined as the optimal protocol in an altered RDP setting wherein the discs are continually reactivated, and this CR protocol is efficiently computed in the RDP setting through the use of TAG. The CR protocol is then proved to be optimal for parallel graphs in the discrete SDL setting, and this theorem is extended to yield optimal protocols for a broader class of SDL problems where the DM’s choice is just between parallel avenues under fixed policies within the avenues.

Chapter 6 presents summary, conclusions, and directions for future research.

Advisor: Prof. Donniell E. Fishkind

Second Reader: Prof. Carey E. Priebe

Acknowledgements

First, I would to thank my wife Şükran for her continued support and enormous patience during my Ph.D. studies. For me, trying to maintain a reasonably normal family life with my wife and our little daughter Feyza was more challenging than the program itself (which was already extremely challenging). I was usually there, but I wasn't really "there", and I owe an apology to her for that. Next, to my parents and brother & sister, and my friends for their love and support.

I would like to express my deepest appreciation to my advisor Prof. Donniell Fishkind for his support, guidance, and patience. He has been not only a great mentor, but also a great friend. He helped me see things in a way that I would never be able to see otherwise—I learned so much from him. He patiently spent countless hours with me for research discussion and editing, which resulted in a drastic improvement of this dissertation. I am grateful for everything I learned from him and everything he has done for me.

I would like to express my sincere gratitude to Prof. Carey Priebe, my second reader and PI of the ONR RDP project (Grant N000140610013) that supported me during the last three semesters of my studies. He was always willing to help and he always had great judgement. He provided valuable insight and suggestions during the development of this dissertation, which I very much appreciate. I also would like to thank Dr. Wendy Martinez and Dr. Don Wagner with ONR for their support and funding of the RDP project.

Chapters 3 and 4 are joint work with Prof. Fishkind, Prof. Priebe, Leslie Smith, and Kendall Giles. I would like to thank Kendall for giving me permission to use his graphical interface for the RDP java code.

I would like to thank Prof. Lawrence Carin of Duke University for several valuable discussions regarding state space reduction in RDP, which motivated the development of the *BAO** Algorithm, and Prof. Alfred Hero of University of Michigan for his valuable insight on POMDP aspects of RDP and its variants.

Special thanks goes to Prof. Benjamin Hobbs for serving on my candidacy exam, GBO, and dissertation committees; Prof. Alan Goldman, Prof. Michael Miller, and Prof. Justin Williams for serving on my GBO committee; and Prof. Shih-Ping Han for serving on my dissertation committee.

I want to thank our department chair Prof. Daniel Naiman, Kristin Bechtel, Sandy Kirt, and Amy Berdann for all their help and support during the five years I spent at the department. I would like to thank Beryl Castello for being a good friend and helping me with various $\text{\LaTeX}$ problems.

Many thanks to the faculty of the Applied Mathematics & Statistics department for everything I learned from them, and many thanks to all my friends at the department for making this past five years a fun and memorable one.

Finally, I would like to thank the legendary Johnny Cash, whose music helped me keep awake many late nights and early mornings. Rest in peace Johnny, God Bless.

Contents

Abstract	ii
Acknowledgements	iv
List of Figures	viii
List of Tables	x
1 Introduction	1
1.1 Introduction	1
1.2 Continuous SDL	2
1.3 Discrete SDL	2
1.4 RDP	4
1.5 Discrete RDP	8
1.6 Literature Review	9
1.7 Organization of the Dissertation	11
2 The BAO* Algorithm	13
2.1 Introduction	13
2.2 Markov Decision Process Formulation	14
2.3 The BAO* Algorithm	19
2.3.1 AO Trees	20
2.3.2 The AO* Algorithm	24
2.3.3 BAO*	26
2.4 Computational Experiments	30
2.5 Partially Observable Markov Decision Process Formulation	33
3 Data Structures	37
3.1 Introduction	37
3.2 The Visibility Graph	38
3.3 The Tangent Arc Graph	40

3.4	The A^* and Dijkstra's Algorithms	42
4	The Simulated Risk Disambiguation Protocol	48
4.1	Introduction	48
4.2	The Simulated Risk Disambiguation Protocol	49
4.2.1	Linear Undesirability Functions	52
4.3	Mine Countermeasures Example	53
4.4	Minimizing $El^e p_{\mathcal{D}_\alpha}$ over $\alpha \geq 0$	59
4.5	Adaption to DRDP	64
4.6	Computational Experiments	67
5	The CR Disambiguation Protocol	69
5.1	Introduction	69
5.2	The CR Setting and the CR Protocol	70
5.3	The CR Protocol for RDP	72
5.3.1	Adaptation to DRDP	73
5.3.2	Optimality of CR Protocol for Discrete SDL on Parallel Graphs	74
5.3.3	Adaptation to SDL with Parallel Avenues	77
5.4	Computational Experiments	80
5.4.1	Comparison of the CR Protocol Against the Optimal Policy .	81
5.4.2	Comparison of the CR and SR Protocols on a Mine Counter- measures Example	82
5.4.3	Comparison Against the SR Protocol on Random RDP Instances	83
5.5	Neutralization as a Special Case of CR Setting	87
6	Summary, Conclusions, and Directions for Future Research	89
6.1	Summary and Conclusions	89
6.2	Computing Optimal α in the Simulated Risk Disambiguation Protocol	92
6.3	Approximate Solutions	92
6.4	Multiple, Noisy Sensors and Neutralization	93
6.5	Roving Detections	93
6.6	Asymptotic Analysis via Percolation Theory	93
	Bibliography	94
	Vita	105

List of Figures

1.1	An example of a random disambiguation path. $X = \{x_1, x_2\}$ consists of the centers of the two discs.	8
2.1	Illustration of a simplified DRDP instance with two discs.	22
2.2	Illustration of the partial AO tree corresponding to the DRDP instance illustrated in Figure 2.1.	23
2.3	The BAO^* Algorithm.	29
2.4	An experimental DRDP realization with $ X = 10$	31
2.5	An experimental DRDP realization with $ X = 20$	32
3.1	An example of a visibility graph. The dashed arcs are not visibility graph edges since they are not in $\partial(\cup_{x \in X} R_x)$	39
3.2	An example of a tangent arc graph.	41
3.3	Dijkstra's Algorithm.	44
4.1	Marked point process realization. Gray-scale of discs reflects ρ of detections at respective disc's centers.	54
4.2	All (two) possible realizations of $p_{\mathcal{D}_{2000}}$	55
4.3	All (seven) possible realizations of $p_{\mathcal{D}_{100}}$. The probabilities are the respective probabilities of the path realizations.	56
4.4	$El^e p_{\mathcal{D}_\alpha}$ as a function of α , for $c = 5$	57
4.5	$El^e p_{\mathcal{D}_\alpha}$ for the optimal α , as a function of c	60
4.6	Nine realizations of a marked point process.	62
4.7	Plots of $El^e p_{\mathcal{D}_\alpha}$ against $\alpha = 2, 7, 12, 17, \dots$ for the respective nine realizations in Figure 4.6. Note that in each of these plots, for the largest value of α which is plotted, we have $p_{\mathcal{D}_\alpha} = p_{\mathcal{D}_\infty}$	63
4.8	Edge weights illustration with $ X = 4$	66
5.1	Illustration of an SDL with parallel avenues problem instance with three parallel avenues.	79
5.2	An illustration of simulation environment A.	84

5.3	An illustration of simulation environment B.	85
5.4	An illustration of simulation environment C.	85

List of Tables

2.1	Comparison of SDP, AO*, and BAO* on randomly generated DRDP instances. For $ X = 10$, SDP did not run for either K due to excessive memory requirements.	31
4.1	x,y-coordinates and ρ 's for marked point process realization.	54
4.2	All indifference intervals, the $El^e p_{D_\alpha}$ for values of α in the respective indifference intervals, and the range of disambiguation costs c where this indifference interval is optimal. For example, for any value $0 < c < 4.1013$ the optimal value of α is any $26.77 < \alpha < 30.23$ and, as such, $El^e p_{D_\alpha} = 717.22 + 2.1665c$	59
4.3	Weights of the edges in Figure 4.8.	67
4.4	Comparison of $El\hat{p}_{SR}^*$ to Elp^*	68
5.1	Comparison of Elp_{CR} to Elp^*	82
5.2	Comparison of the CR and SR protocols on a mine countermeasures example for different cost intervals.	83
5.3	Comparison of Elp_{CR} to $El\hat{p}_{SR}^*$	86

Chapter 1

Introduction

1.1 Introduction

The research problem we consider in this dissertation, in its most broad setting, is a stochastic shortest path problem in the presence of a dynamic learning capability. Specifically, a spatial arrangement of possible obstacles needs to be traversed as swiftly as possible, and the status of the obstacles may be disambiguated (at a cost) en route. This problem has practical applications in important probabilistic path planning environments such as robot navigation in stochastic domains [15, 28, 45], mine countermeasures [70, 76], and adaptive traffic routing [27, 31]. No efficiently computable optimal policy is known for this problem, and many similar problems have been proven intractable. In this chapter, the research problem is defined in continuous and discrete settings and a comprehensive literature review is presented.

1.2 Continuous SDL

Suppose a decision maker (DM) needs to swiftly navigate from one given location to another through an arrangement of arbitrarily-shaped regions in the plane which are possible obstacles; at the outset the DM is given the respective probabilities that the regions are truly obstacles and, en route, when situated on a region's boundary, the DM has the option to *disambiguate* the region, i.e., learn at a cost if the region is a true obstacle. We may sometimes assume that there is a limit on the number of available disambiguations. The central question is to find a policy (protocol)¹ that decides what and where to disambiguate en route so as to minimize the expected length of the traversal. We call this problem the *continuous Stochastic Shortest Path Problem With Dynamic Learning Capability (SDL)*, which is a minor modification of the *Stochastic Obstacle Scene Problem (SOSP)* of Papadimitriou and Yannakakis [55].

1.3 Discrete SDL

The discrete analogue of the above problem, which we call *discrete SDL*, is defined as follows: Let $G = (V, E)$ be an undirected graph with designated vertices $s, t \in V$, and suppose there is a function $\ell : E \rightarrow \mathbb{R}_{\geq 0}$ assigning a length to each edge; the goal here is to find a shortest s, t traversal (walk) in G . However, not all of the edges

¹In this dissertation, we do not make any distinctions between a “policy” and a “protocol”, and we use the two terms interchangeably.

may indeed be traversable. In particular, for a given subset $E' \subseteq E$ of edges, called *stochastic edges*, there is a function $\rho : E' \rightarrow [0, 1)$ such that, for each edge $e \in E'$, $\rho(e)$ is the probability that e is not traversable, independent of the other edges. (The edges in $E \setminus E'$ are known a priori to be traversable.) For any edge $e \in E'$, when the traversal is at an endpoint of e , we have the option to disambiguate e —learning whether e is traversable—at a cost $c(e)$ added to the length of the traversal, for some function $c : E' \rightarrow \mathbb{R}_{\geq 0}$. Edges cannot be traversed until it is known that they are traversable, and, the traversability status of each edge is static and will never change over the course of the traversal.

Of course, if the DM follows any particular policy then the traversal is still random (and will unfold depending on the results of the disambiguations, so the traversal will have distribution specified through ρ). The goal, however, is to find an optimal policy in the sense of having shortest expected length. (As in the continuous version, we may sometimes assume that there is a limit K on the number of available disambiguations.) Finding such an optimal policy is the discrete SDL problem, which is known as the *Canadian Traveler Problem (CTP)* in the literature, and has been shown to be intractable [55]. To avoid infinite expected length, it may be helpful to assume the existence of a (possibly very long) s, t path consisting of edges from $\{e \in E' : \rho(e) = 0\} \cup (E \setminus E')$.

Unless specified otherwise, we shall use the acronym “SDL” to collectively refer to both the continuous and discrete versions of the problem.

1.4 RDP

The main focus of this dissertation is on a variant of continuous SDL with *disc-shaped* possible-obstacles, called the *Random Disambiguation Paths Problem (RDP)* for historical reasons [29, 60]. In what follows, we describe the problem environment and formally define the RDP problem.

As in [60], consider a marked point process on a subset of $\mathbb{R}^2$ that generates random detections $X_T, X_F \subseteq \mathbb{R}^2$ (respectively called *true* and *false* detections), and, random marks $\rho_T : X_T \rightarrow (0, 1]$ and $\rho_F : X_F \rightarrow (0, 1]$. When observing a realization of this process, the decision maker (DM) only sees $X := X_T \cup X_F$ and $\rho := \rho_T \cup \rho_F$. We assume that, for all $x \in X$, $\rho(x)$ is the probability—conditioned on the observed values X and ρ —that $x \in X_T$. It shall be assumed that the specific values of X and ρ have been observed and all discussion of probability is conditioned accordingly. We assume that whether or not any one $x \in X$ is in X_T is independent of any other $x' \in X$. For every detection x , the possibly obstacle region R_x is an open disc centered at x with radius $r(x) > 0$, for a given function $r : X \rightarrow \mathbb{R}_{>0}$. Given a starting point $s \in \mathbb{R}^2$ and a destination point $t \in \mathbb{R}^2$, the DM seeks to traverse a continuous s, t curve in

$(\bigcup_{x \in X_T} R_x)^C$ of shortest achievable arclength.

Without any means of discovering which detections in X are true, the DM cannot traverse any s, t curve shorter than the shortest s, t curve in $(\bigcup_{x \in X} R_x)^C$, denoted $q(s, t, X)$. (The curve $q(s, t, X)$ can be computed using the associated *visibility graph* described in Section 3.2). However, the DM has a dynamic capability of disambiguating detections from the boundaries of their associated discs. Specifically, when the DM is on ∂R_x for any $x \in X$, the DM can dynamically discover whether $x \in X_T$ or $x \in X_F$, at a cost $c(x)$ added to the Euclidean length of the curve, for a given cost function $c : X \rightarrow \mathbb{R}_{\geq 0}$. The DM is permitted to proceed through R_x only if $x \in X_F$. There may also be a limit K on the number of available disambiguations. The DM's goal is to minimize the traversal curve's (expected) Euclidean length by efficiently exploiting the disambiguation capability.

A *disambiguation protocol* is a decision rule that specifies the DM's actions under any circumstance and at any possible current location. Specifically, it is a function $\mathcal{D}$ that, to any such s, t, X, ρ, K , it assigns a detection $x \in X$ and a point $y \in \partial R_x$ (we explicitly allow $y = t$, in which case x is not defined). Given a disambiguation protocol $\mathcal{D}$, the *random disambiguation path* $p_{\mathcal{D}}$ is defined as the s, t curve in $(\bigcup_{x \in X_T} R_x)^C$ realized as follows: Suppose $\mathcal{D}$ associates $x \in X$ and $y \in \partial R_x$ to s, t, X, ρ, K . Then the DM is to traverse from s to y along a shortest path avoiding all the discs $R_{x'}$

such that $x' \in X$ is true or ambiguous. If $y = t$ then terminate, otherwise disambiguate detection x (if $K = 0$ then it is required that $y = t$). The DM recursively repeats this entire procedure using y in place of s , updating X and ρ as follows: If the disambiguation has just discovered that $x \in X_T$ then update $\rho(x) := 1$, and if the disambiguation has just discovered that $x \in X_F$ then remove x from X ; either way, decrement K by 1 (if indeed there is a limit on the number of available disambiguations).

The random disambiguation path $p_{\mathcal{D}}$ is an s, t -curve-valued random variable even after X and ρ are observed since the emerging outcomes of the disambiguations dictated by the protocol are still random. In fact, the distribution of $p_{\mathcal{D}}$ is specified through ρ .

The *Random Disambiguation Paths Problem (RDP)* is then defined as choosing the disambiguation protocol—among all possible disambiguation protocols—that minimizes the expected total traversal length.

To illustrate, suppose $K = 2$ and s, t, X, ρ are as shown in Figure 1.1, and consider one particular disambiguation protocol $\mathcal{D}$ which, say, dictates that the next (*i.e.*, first) disambiguation be of detection x_1 at point y_1 . Now suppose, if it is discovered that $x_1 \in X_F$, $\mathcal{D}$ would then dictate that no more disambiguations be performed, and the curve should proceed to t . Also suppose, if it is instead discovered that $x_1 \in X_T$, $\mathcal{D}$

would then dictate that x_2 should be disambiguated next, at point y_2 . Whether x_2 is revealed to be a true or false detection, $\mathcal{D}$ would then dictate that the DM proceeds directly to t , since no more disambiguations are available (currently $K = 0$). There are three possible realizations of the random disambiguation path $p_{\mathcal{D}}$, each pictured in Figure 1.1:

- With $1 - \rho(x_1) = .7$ probability, $p_{\mathcal{D}}$ traverses the points s, y_1, t ;
- With $\rho(x_1)\rho(x_2) = .27$ probability, $p_{\mathcal{D}}$ traverses the points s, y_1, y_2, t employing the curve γ at the traversal conclusion; and,
- With $\rho(x_1)(1 - \rho(x_2)) = .03$ probability, $p_{\mathcal{D}}$ traverses the points s, y_1, y_2, t employing the line segment $\overline{y_2, t}$ at the traversal conclusion.

Note that in between disambiguations the s, t curve traverses the shortest curves avoiding all possibly forbidden risk regions—using the corresponding visibility graphs. If the lengths of these three paths are, respectively, 6, 8, 7, and if the cost of disambiguation was $c = 5$ then then the expected length of $p_{\mathcal{D}}$ is given by $(1 - .3)(6 + 1 * 5) + (.3)(.9)(8 + 2 * 5) + (.3)(1 - .9)(7 + 2 * 5)$.

In the above example, we illustrated one particular disambiguation protocol $\mathcal{D}$; a different choice of protocol may indeed yield a significantly lower expected length.

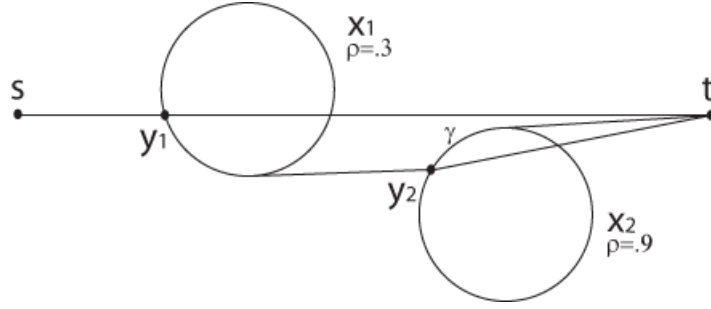


Figure 1.1: An example of a random disambiguation path. $X = \{x_1, x_2\}$ consists of the centers of the two discs.

1.5 Discrete RDP

Optimal disambiguation protocols are not readily computable for all but the most trivial instances of RDP. We therefore consider a discrete approximation which is, for simplicity and convenience, on a subgraph of the integer lattice $\mathbb{Z}^2$. Specifically, it is the graph G whose vertices are all of the pairs of integers i, j such that $1 \leq i \leq i_{\max}$ and $1 \leq j \leq j_{\max}$, where $i_{\max}$ and $j_{\max}$ are given integers. There are edges between all pairs of vertices of the form (i, j) and $(i + 1, j)$, and there are edges between all pairs of vertices of the form (i, j) and $(i, j + 1)$. One vertex in G is designated as the starting point s , another vertex in G is designated as the termination point t , and the decision maker is to walk from s to t in G , only traversing edges that do not intersect any true or ambiguous obstacles. If an edge intersects any ambiguous obstacle, then a disambiguation of the obstacle may be performed from either of the edge's endpoints that is outside of the obstacle. As before, the goal is to develop a policy that minimizes the expected length of the traversal by effective exploitation of

the disambiguation capability. We call the discretized version as the *Discrete RDP* (*DRDP*), which, in effect, is a special case of discrete SDL (and CTP) with statistical dependency among the edges.

1.6 Literature Review

The shortest path problem in deterministic networks—where each arc has a fixed, known length—is widely encountered in theory as well as practice and has been well-studied. Various efficient algorithms exist for solving such problems and a traditional treatment of this subject can be found in [1, 22].

In many practical applications, however, arclengths are not known in advance and it can be more realistic to model them as random variables, which gives rise to *stochastic shortest path problems*. A particular version of this class of problems is the one with a *dynamic learning capability*, where the realization of arclengths is learned as the traversal progresses. A wide variety of such problems have been studied, including arc-wise and time-wise dependencies [21, 24, 30, 49, 73]. Most of these studies were conducted by researchers in the disciplines of transportation science [8, 20, 27, 42], computer science (theoretical computer science and artificial intelligence) [9, 15, 28, 45, 55], and operations research [6, 7, 12, 23, 59, 63]. The work in these three communities was done, by and large, independent of each other.

The continuous SDL problem is a minor modification of the Stochastic Obstacle Scene Problem (SOSP) of Papadimitriou and Yannakakis [55], where possibly intersecting rectilinear blocks may be present in an obstacle scene and whether a specific block exists or not can be learned only by visual contact. The authors also describe a discrete version of this problem called the Canadian Traveler Problem (CTP), which we refer to as discrete SDL in this dissertation. Papadimitriou and Yannakakis prove the intractability of several variants of SOSP and CTP. Several modifications and extensions of CTP are discussed in [9], [15], and [45]. Baglietto *et al.* [7] cast CTP as a stochastic optimal control problem and propose a heuristic method based on neurodynamic programming. Ferguson, Stenz, and Thrun [28] also propose a heuristic for CTP, which is a more elaborate version of the AO^* Algorithm.

CTP is a special case of the Stochastic Shortest Paths with Recourse (SPR) problem of Andreatta and Romeo [6], who present a stochastic dynamic programming formulation for SPR and note its intractability. Polychronopoulos and Tsitsiklis [59] also present a stochastic dynamic programming formulation for SPR and then prove the intractability of several variants. Provan [63] proves that SPR is intractable even if the underlying graph is directed and acyclic.

The underlying difficulty in obtaining a tractable stochastic dynamic programming

formulation of these problems—even in the discrete setting—is that in order for the actions to be considered at any given location, there is a need to know the current ambiguous/true/false status of all of the detections, and the exponentially many such possibilities need to be incorporated accordingly.

1.7 Organization of the Dissertation

The rest of this dissertation is organized as follows:

Chapter 2 casts SDL as a Markov Decision Process and introduces the *BAO** Algorithm for DRDP, which is not polynomial-time, but can be used to optimally solve relatively small instances of the problem. This chapter also formulates RDP as a Partially Observable Markov Decision Process.

Chapter 3 discusses visibility graphs and introduces the tangent arc graph data structure, which plays an important role in the next chapters. This chapter also points out that the well-known *A** Algorithm with a slightly stronger admissibility requirement on the heuristic function is equivalent to Dijkstra’s Algorithm under a change of variable.

Chapter 4 presents the simulated risk disambiguation protocol (SR) for RDP and

DRDP, which is suboptimal, but efficiently computable as well as reasonably effective.

Chapter 5 introduces the CR disambiguation protocol for RDP and DRDP, which is also suboptimal, but much more efficiently computable even than SR, and, as illustrated via simulations, yields solutions that compare favorably to those obtained by the optimal policy and also SR. The chapter proves optimality of the CR protocol for parallel graphs in the discrete setting. This chapter also shows that the situation where there is an obstacle neutralization capability instead of the disambiguation capability can be seen as a special case of the setting that leads to the CR protocol.

Chapter 6 presents summary, conclusions, and directions for future research.

Chapter 2

The BAO* Algorithm

2.1 Introduction

In this chapter, we first cast SDL as a Markov Decision Process and develop the corresponding Bellman equation, whose solution via stochastic dynamic programming can yield the optimal policy. However, the state space is too large to efficiently utilize the dynamic programming paradigm. Next, we define *AO* trees, which can be used to allow partial solutions of the optimality conditions to be selectively searched—without exhaustively back-solving complete stages as in stochastic dynamic programming. In particular, the well-known *AO** Algorithm was devised for searching *AO* trees. Introduced in Section 2.3 is a new improvement on *AO** that we call the *BAO** Algorithm, which employs stronger pruning methods, including the utilization of both upper and lower bounds on path lengths, and significantly speeds up the search process. The

chapter concludes with a Partially Observable Markov Decision Process formulation of RDP.

2.2 Markov Decision Process Formulation

A Markov Decision Process (MDP) is a mathematical framework for modeling sequential decision-making processes in an environment with the Markov property, i.e., the complete past history is summarized in the current state. At discrete time steps called stages, a decision maker (DM) is at a state contained in a state space, at which he chooses an action from a set of actions and earns a reward that depends on the current state and the action chosen. The state arrived at the next stage is probabilistic and the distribution depends only on the current state and the action chosen. The DM's goal is to maximize the expected sum of rewards earned over all stages, sometimes with a discount factor on future rewards. It is assumed that the DM knows his current state at all times. An in-depth discussion of MDPs can be found in [64, 74].

Formally, an MDP is a 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ where

- $\mathcal{S}$ is a set of states: At every stage $k = 0, 1, 2, \dots, \mathcal{K}$ (where $\mathcal{K}$ is the final stage, or $\mathcal{K} = \infty$), the DM is at one of these states;
- $\mathcal{A}$ is a set of actions: At every stage, the DM chooses one of them depending

on what his current state is;

- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the state-transition function: For any $s, s' \in \mathcal{S}$ and $\alpha \in \mathcal{A}$, $\mathcal{T}(s, \alpha, s')$ is the probability of ending up in state s' in the next stage given that the DM is at state s in the current stage and chooses action α ; and,
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function: $\mathcal{R}(s, \alpha)$ represents the immediate reward the DM gains for choosing action α at state s .

The DM's objective is to maximize the expected sum of rewards (possibly discounted by a factor $\lambda \in (0, 1]$, i.e., $E[\sum_{k=0}^{\mathcal{K}} \lambda^k R_k]$ where R_k is the reward received at stage k).

We now simultaneously cast continuous and discrete SDL as a finite-horizon MDP (i.e., $\mathcal{K} < \infty$), formulating the components of $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ as follows.

- **States:** Let N be the number of possible-obstacles in the continuous setting, and the number of stochastic edges in the discrete setting. In order to keep track of the decision maker's current knowledge of the status of possible-obstacles (or stochastic edges), we define the *information vector* $\mathcal{I} \in \{\text{"A"}, \text{"T"}, \text{"F"}\}^N$, such that, for all $i = 1, 2, \dots, N$, the i -th entry of $\mathcal{I}$ is "A", "T", or "F", according as the current status of i -th possible-obstacle (or stochastic edge) is ambiguous, true, or false, respectively.

Let $\mathcal{Y}$ be the union of s , t , and the set of all disambiguation locations, i.e., locations at which possible-obstacles (or stochastic edges) can be disambiguated. (In continuous SDL, these locations are points on the plane and, in discrete SDL, they are vertices in the underlying graph.) If there are certain locations at which multiple possible-obstacles (or stochastic edges) can be disambiguated, these locations are included in $\mathcal{Y}$ with their respective multiplicities.

The MDP state space $\mathcal{S}$ is defined as $\mathcal{Y} \times \{\text{"A"}, \text{"T"}, \text{"F"}\}^N$. The state space thus represents possible locations at which the DM may be at a particular stage, coupled with information that describes the decision maker's knowledge at that stage.

- **Actions:** The set of actions $\mathcal{A}$ is $\mathcal{Y} \setminus \{s\}$, i.e., all the locations where a disambiguation can be performed and the destination.
- **State Transition Function:** Given a state and an action, the state transitioned into is comprised of the location identified in the action and, the information vector of the previous state updated to indicate whether the possible-obstacle (or stochastic edge) identified in the action is true or false; with respective probabilities being specified according to its mark.

- **Rewards:** The reward for a specific action at any particular state is the negative of the shortest path distance between the location identified in the state and the location identified in the action—avoiding all ambiguous and true obstacles (or stochastic edges) as indicated by that state’s information vector. The disambiguation cost is also subtracted if the location identified in the action is not the destination.

The above state space, set of actions, rewards, and state transition function comprise a Markov Decision Process with K stages (or N stages if there is no limit on the number of available disambiguations).

We now present the Bellman equation corresponding to the above MDP formulation, which can be solved via standard stochastic dynamic programming (SDP) for relatively small instances in the discrete setting. The reader is referred to [10, 11, 51] for a general discussion of dynamic programming and [67] for an introduction to stochastic dynamic programming. The notation used in the Bellman equation is as follows.

- For $\mathbf{s} = (y, \mathcal{I}) \in \mathcal{S}$ and stage $k \leq K$, the *value function* $V_k^* : \mathcal{S} \rightarrow \mathbb{R}$ is defined as the negative of the shortest expected y, t path length under an optimal policy when the status of the obstacle field (or the underlying graph) is $\mathcal{I}$, and there

are k disambiguations left.

- For any $y, y' \in \mathcal{Y}$ and information vector $\mathcal{I}$, $q(y, y', \mathcal{I})$ is defined as the length of the shortest y, y' path while avoiding all the true and ambiguous obstacles (or stochastic edges) as indicated by $\mathcal{I}$.
- For any $y \in \mathcal{Y}$, $\mathcal{I}_y$ is defined as the component of $\mathcal{I}$ corresponding to the possible-obstacle (or stochastic edge) associated with y .
- For any $y \in \mathcal{Y}$, $\rho(y)$ is defined as the mark of the possible-obstacle (or stochastic edge) associated with y .
- For information vector $\mathcal{I}$ and $y \in \mathcal{Y}$, $T_{\mathcal{I},y}$ and $F_{\mathcal{I},y}$ are defined as the information vectors whose components are the same as $\mathcal{I}$ except at the component corresponding to y , which is set to “T” and “F”, respectively.

For $k = 1, \dots, K$, and $\mathbf{s} = (y, \mathcal{I}) \in \mathcal{S}$, the Bellman equation is as follows:

$$V_k^*(\mathbf{s}) = \max_{\substack{y' \in \mathcal{Y} \text{ s.t.} \\ y'=t \text{ or } \mathcal{I}_{y'}=\text{“A”}}} \left\{ -q(y, y', \mathcal{I}) - \delta_{y' \neq t} \cdot c + \rho(y')V_{k-1}^*(y', T_{\mathcal{I},y'}) \right. \\ \left. + (1 - \rho(y'))V_{k-1}^*(y', F_{\mathcal{I},y'}) \right\} \quad (2.1)$$

where δ is the indicator function. The optimal solution to SDL is then given by $-V_K^*((s, (\text{“A”}, \dots, \text{“A”})))$. Note that an SDP solution entails exhaustively back-solving complete stages from stage 1 up to stage K , where stage 0 values $V_0^*((y, \mathcal{I}))$ are given

by $-q(y, t, \mathcal{I})$.

For continuous SDL, the above SDP approach is not practically applicable due to the infinitely many states and actions—yet they provide valuable insight into the structure of this class of problems and illustrate their difficulty. Due to exponentially many states, the SDP approach is not practical for discrete SDL either, as illustrated in the computational experiments of Section 2.4.

2.3 The BAO* Algorithm

In this section, we first define *AO* trees, which can be used to represent a given DRDP instance as a rooted tree with exponentially many nodes—the branches of the tree correspond to sequential decisions that can be made and their probabilistic outcomes. We next describe the *AO** Algorithm for searching *AO* trees, and introduce the *BAO** Algorithm [2], which stands for *AO** with bounds. The *BAO** Algorithm improves upon *AO** by utilizing stronger pruning techniques and speeds up the search process significantly.

2.3.1 AO Trees

An *AO* tree is defined as a rooted tree $T = (\mathbf{N}, \mathbf{A})$ with a function $\ell : \mathbf{A} \rightarrow \mathbb{R}_{\geq 0}$ assigning a length to each arc, and a function $p : \mathbf{A} \rightarrow [0, 1]$ assigning a probability to each arc. The node set $\mathbf{N}$ is partitioned into a set of AND nodes, denoted by $\mathbf{N}_A$, and a set of OR nodes, denoted by $\mathbf{N}_O$. All arcs emanating from OR nodes have probability one. AO trees are the AND-OR graphs commonly found in the literature, see e.g., [52].

Denote by $S(n)$ the successors of the node $n \in \mathbf{N}$ in the AO tree. A function $g : \mathbf{N} \rightarrow \mathbb{R}_{\geq 0}$ is said to be *consistent* on a collection of nodes $\mathbf{C} \in \mathbf{N}$ provided that, for all $n \in \mathbf{C}$, the following three conditions are satisfied:

- if $n \in \mathbf{N}_A$, $g(n) = \sum_{n' \in S(n)} [p(n, n') \cdot (\ell(n, n') + g(n'))]$,
- if $n \in \mathbf{N}_O$, $g(n) = \min_{n' \in S(n)} \{\ell(n, n') + g(n')\}$, and,
- if $n \in \mathbf{N}$ is a leaf node, $g(n)$ is zero.

The function $f : \mathbf{N} \rightarrow \mathbb{R}_{\geq 0}$ that is consistent on $\mathbf{N}$ is called the *cost-to-go function* and $f(n)$ is called the *true label*¹ of node n . Typically, ℓ and p are given explicitly, and f is implicitly defined via ℓ and p . Given an *AO* tree, the goal is to compute the true label of the root node, which denotes the optimal value of an underlying decision problem.

¹In our context, f will denote the negative of the SDP value function V^* defined in Section 2.2.

A DRDP instance can be solved by computing the true label of the root node of an appropriately chosen *AO* tree. Specifically, associated with each node n is a state $\mathbf{s}_n = (y_n, \mathcal{I}_n)$ from the state space $\mathcal{S} = \mathcal{Y} \times \{\text{"A"}, \text{"T"}, \text{"F"}\}^{|X|}$. The root r is an OR node with $y_r = s$ and $\mathcal{I}_r = (\text{"A"}, \dots, \text{"A"})$, which is the first generation in the tree. All subsequent odd generations consist of OR nodes corresponding to possible disambiguation locations, and even generations consist of AND nodes which are either leaf nodes denoting direct traversal to destination, or, have two successors each of whom corresponds to true and false disambiguation outcomes. For any arc $a = (n, n') \in \mathbf{A}$,

- If $n \in \mathbf{N}_O$, $\ell(a)$ is set to $q(y_n, y_{n'}, \mathcal{I}_n)$ and $p(a)$ is set to one.
- If $n \in \mathbf{N}_A$, $\ell(a)$ is set to the disambiguation cost c and, $p(a)$ is set to $\rho(y_n)$ if n' corresponds to a true disambiguation outcome and $1 - \rho(y_n)$ otherwise.

Note that the same state could be associated with multiple nodes in the *AO* tree. This construction is essentially a mapping of all the actions the DM can choose and all the disambiguation outcomes that can occur. In particular, this construction ensures that $f(n)$ is the negative of $V^*(s_n)$ for any node $n \in \mathbf{N}_O$.

As an example, consider the DRDP instance in Figure 2.1 with two discs centered at $(3, 2)$ and $(6, 2)$ respectively, with $s = (1, 2)$ and $t = (8, 2)$. For simplicity, we assume that each disc can be disambiguated only at the point to its left, i.e., the first disc

can be disambiguated at $y_1 = (2, 2)$ and the second at $y_2 = (5, 2)$.

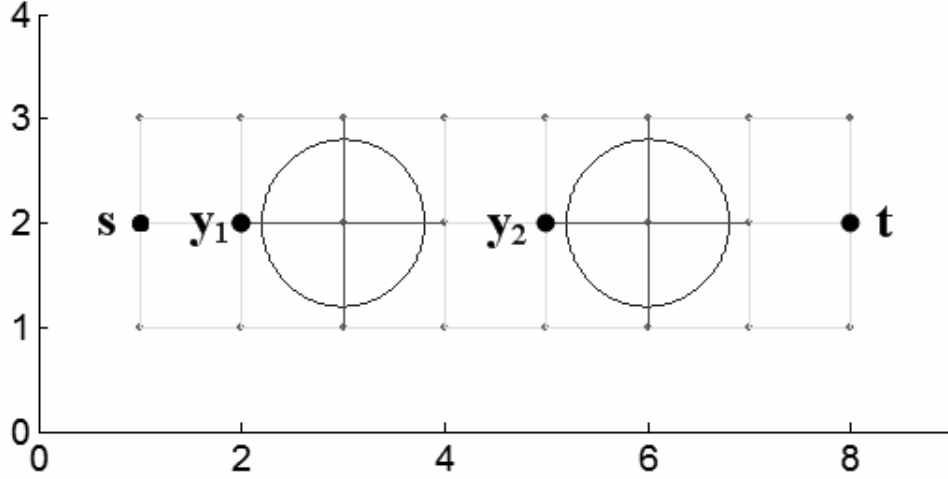


Figure 2.1: Illustration of a simplified DRDP instance with two discs.

A partial *AO* tree corresponding to this instance, for $K = 2$, is illustrated in Figure 2.2, where the successors of the node $(y_2, ("A", "A"))$ are not shown. AND nodes are depicted by circles and OR nodes by squares. The numbers next to each arc denote its length and thick-bordered circles represent leaf nodes.

In an *AO* tree representation of DRDP, the optimal disambiguation protocol is a collection $\mathcal{C}$ of all the arcs whose lengths are explicitly included in the calculation of $f(r)$. Specifically, we first define the function $m : \mathbf{N}_O \rightarrow \mathbf{N}_A$ such that $m(n) := \arg \min_{n' \in S(n)} \{\ell(n, n') + f(n')\}$ for any $n \in \mathbf{N}_O$. We identify the collection $\mathcal{C}$ recursively as follows:

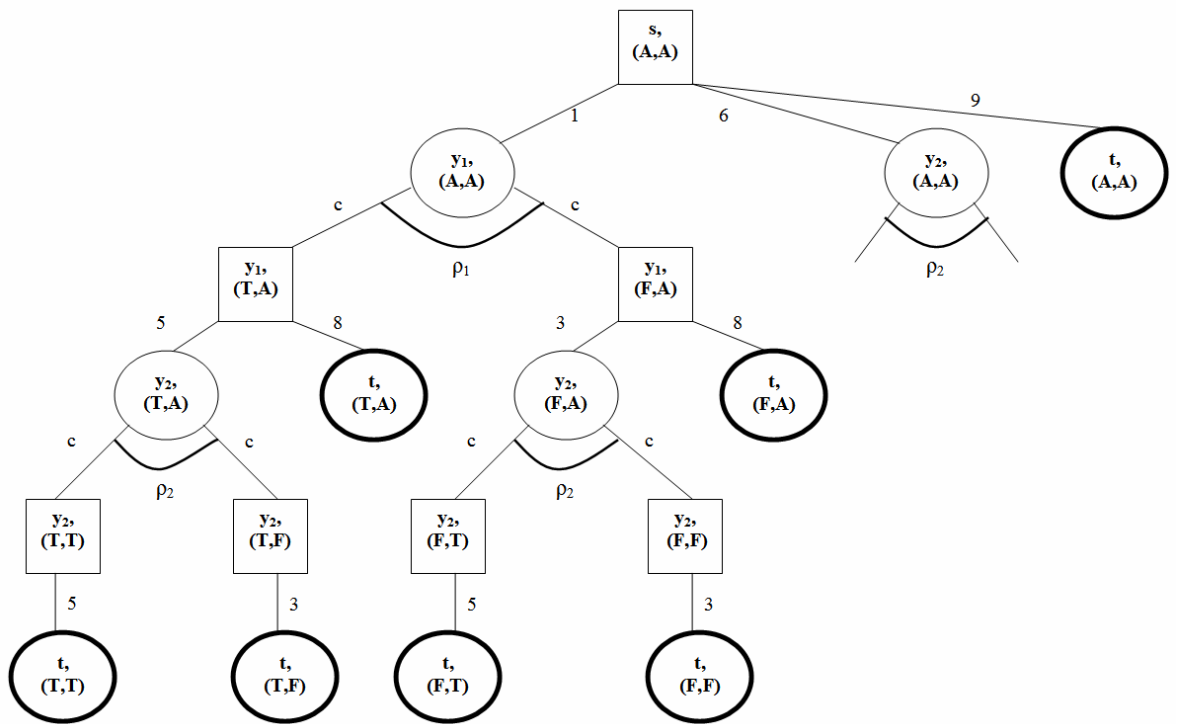


Figure 2.2: Illustration of the partial *AO* tree corresponding to the DRDP instance illustrated in Figure 2.1.

Step 1. Set $\mathcal{C} := \emptyset$ and $n_m := r$.

Step 2. If $m(n_m)$ is a leaf node, augment $\mathcal{C}$ by $(n_m, m(n_m))$. Otherwise, augment $\mathcal{C}$ by $(n_m, m(n_m))$, $(m(n_m), n')$, and $(m(n_m), n'')$ where n' and n'' are the successors of $m(n_m)$.

Step 3a. Set $n_m := n'$. If n_m is not a leaf node, go to Step 2.

Step 3b. Set $n_m := n''$. If n_m is not a leaf node, go to Step 2.

The benefit of the *AO* tree representation is that, as shown in Section 2.3.2, it allows for selectively evaluating the value function in a top-down fashion rather than back-computing all of them for every stage as in the dynamic programming paradigm.

2.3.2 The AO* Algorithm

Theoretically, an optimal solution to a problem represented by an *AO* tree can be determined by computing $f(n)$ for all $n \in \mathbf{N}$ in a bottom-up fashion. However, the exponential number of nodes in the *AO* tree representation of DRDP makes this approach prohibitively expensive. On the other hand, not all the nodes' true labels need to be calculated to determine the true label of the root node. We define *searching* an *AO* tree as identifying the nodes that are of interest in determining the true label of the root node.

The classical AO^* Algorithm for searching AO trees [18, 48, 52] improves upon the brute force approach by utilizing *admissible* bounds $h : \mathbf{N} \rightarrow \mathbb{R}_{\geq 0}$, called *heuristic* labels, which are lower bounds that are guaranteed not to overestimate the true label of any node. These lower bounds guide the search in a top-down fashion so that only a small portion of the complete AO tree is examined. The AO^* Algorithm and its variants have been used by researchers to successfully solve various decision problems [8, 28, 52]. The AO^* Algorithm is described below.

The AO^* Algorithm grows a *solution tree* $S = (\mathbf{N}', \mathbf{A}')$, which is a subtree of the complete AO tree and a representation of partial solutions of the optimality conditions. S initially consists of only the root node r , and is gradually augmented by two alternating steps, *expansion* and *propagation*, until $f(r)$ is computed. A node $n' \in \mathbf{N}'$ is said to be *terminal* if $f(n')$ has been calculated. In the expansion step, the non-terminal leaf node with the lowest h value, called the expansion node and denoted by n'_e , is found and its successors are added to S . The successors are then assigned heuristic labels. In the propagation step, $h(n'_e)$ is recalculated using the labels of its successors—true labels for successors that are leaf nodes in the complete AO tree and heuristic labels otherwise—and the new label is propagated up S until a node is reached whose heuristic label is not affected. Terminal status of nodes are also updated accordingly during the propagation step.

In DRDP, an admissible lower bound on $f(n)$ for any $n \in \mathbb{N}$ is available in the form of the shortest path length from y_n to destination while avoiding only the true stochastic edges as indicated by $\mathcal{I}_n$ —which allows for the use of the AO^* Algorithm to solve relatively small instances of the problem.

2.3.3 BAO^*

An important characteristic of the DRDP problem is that the expected length of the optimal protocol is always upper bounded by the zero-risk s, t traversal. In this section, we present an improvement on AO^* that incorporates this additional information and significantly speeds up the search, which we call BAO^* for AO^* with bounds. The idea of utilizing upper bound information in AO^* is briefly mentioned in [8], but no specific algorithms are presented.

The BAO^* Algorithm intelligently incorporates the lower and upper bound information described above and examines only a very small fraction of the complete AO tree to compute the true label of the root node. BAO^* is not polynomial-time, but it can substantially shorten the running time needed to find an exact solution to small instances. As illustrated in the computational experiments section, BAO^* has significant computational savings over SDP and AO^* . BAO^* pseudocode is presented in Figure 2.3.

Key features and benefits of BAO^* are as follows:

1. BAO^* only expands OR nodes. Successor AND nodes and an AND node representing direct traversal to the destination are automatically generated, but only when the heuristic s, t traversal lengths of these nodes are less than the conventional zero-risk s, t path.
2. BAO^* lower bounds the heuristic label of true OR nodes by the heuristic labels of their parents as it moves down the solution tree during the expansion step; providing better estimates.
3. If there is only one disambiguation left at the current expansion node, the true label of the expansion node is calculated and the corresponding AND node is marked as terminal. This feature eliminates the overhead cost of individually considering the successors of this AND node in future expansions.
4. As the updated labels are propagated up the solution tree, new node labels are used to dynamically delete the nodes whose heuristic s, t traversal lengths exceed the length of the zero-risk s, t path.

A variant of the AO^* Algorithm is discussed in [28], which propagates heuristic labels across to neighbors in the complete AO tree during the propagation process. However, the exponentially many nodes in DRDP representation renders this approach

Algorithm BAO***Input:** DRDP instance.**Output:** Minimum expected traversal length from s to t .

Begin

1. Set $zeroRiskDist := q(s, t, ("A", \dots, "A"))$.
2. Generate node $root$ with state $(s, ("A", \dots, "A"))$.
3. While ($root$ is not marked as terminal)

Begin

% Identify the node to be expanded:

- 3.1. Set $totalPathDist := 0$.
- 3.2. Set $expandNode := root$.
- 3.3. While ($expandNode \neq \text{null}$)

Begin

3.3.1. If $expandNode$ is a non-leaf AND node, lower bound the heuristic label of its true successor by $h(expandNode)$.

3.3.2. Set $totalPathDist += expandNode$'s distance from its parent in the parent's information state.

3.3.3. Set $expandNode := expandNode$'s best non-terminal successor.

End while

% Expand best partial solution:

3.4. Temporarily generate the AND node $(t, expandNode$'s information state). If its total s, t traversal length is less than $zeroRiskDist$, add it to $expandNode$'s successors list.

3.5. If there are any disambiguations left, temporarily generate AND nodes corresponding to each disambiguation location of ambiguous stochastic edges that are reachable in $expandNode$'s information state. Also generate true and false OR successors for them.

3.6. Compute the heuristic labels of these OR nodes, h_{true} and h_{false} , respectively, and set heuristic label of the parent AND node to $h_{weighted} = \rho \cdot h_{true} + (1 - \rho) \cdot h_{false}$, where ρ is the mark of the disc associated with this node.

3.7. For each of the AND nodes, compute a heuristic s, t traversal length as $h_{total} := totalPathDist + (\text{distance to } expandNode \text{ in } expandNode\text{'s information state}) + h_{weighted}$. If $h_{total} < zeroRiskDist$, add these AND nodes to $expandNode$'s successors list and also their two successors. Otherwise, delete them. If there is only one disambiguation left, generate the AND node's two successors and compute their true labels. Also mark this AND node as terminal.

%Propagate new labels and update the solution tree:

4. Set *continueProp* := true.
5. While (*continueProp*) and (*expandNode* != null) do

Begin

 - 5.1. If *expandNode* is an OR node and it does not have any successors, mark it terminal. Otherwise, search the successors of *expandNode*'s parent and find the node that yields the lowest heuristic label. Update the parent's best successor node and the parent's heuristic label.
 - 5.2. If *expandNode* is a non-leaf AND node, set $h(\textit{expandNode})$ to the weighted average of the heuristic labels of its two successors. Update the best non-terminal successor of the parent of *expandNode*. If the parent node does not have any non-terminal successors, mark it terminal.
 - 5.3. If the new $h(\textit{expandNode})$ plus the distance between *expandNode* and the root is not less than *zeroRiskDist*, then delete *expandNode* and all of its successors.
 - 5.4. Set *continueProp* := false if the heuristic label of *expandNode* has not changed.
 - 5.5. Set *expandNode* := *expandNode*'s parent.

End while

End while

End

Figure 2.3: The *BAO** Algorithm.

impractical and we do not incorporate this logic in BAO^* .

2.4 Computational Experiments

In this section, we compare the performances of SDP, AO^* , and BAO^* on relatively small, but non-trivial DRDP instances. For all of the experiments, $i_{\max}$ is taken as 20 and $j_{\max}$ is taken as 15 on the integer lattice, with $s = (10, 15)$ and $t = (10, 1)$. Each possible-obstacle is a disc with radius 2.5, with their centers sampled from a uniform distribution on the pairs of real numbers in $[4, 17] \times [4, 12]$; in particular, this ensures that there is always a permissible path from s to t . Marks of the discs are sampled from a uniform distribution on the unit interval. Throughout the simulations, the disambiguation cost c is taken as 3.0. The simulation environment is illustrated in Figure 2.4 for $|X| = 10$ where the dark-colored edges represent stochastic edges.

Table 2.1 compares the performances of SDP, AO^* , and BAO^* for 20 experiment realizations for each $|X|, K$ combination listed. The third, fourth, and fifth columns in the table show the average execution times of SDP, AO^* , and BAO^* respectively, on a 3.5 gigahertz personal computer with 1 gigabyte memory. SDP did not run for $|X| = 10$ for either K due to insufficient memory. The next column displays the mean number of value function evaluations in SDP. The last two columns show the mean number of nodes expanded by AO^* and BAO^* , respectively.

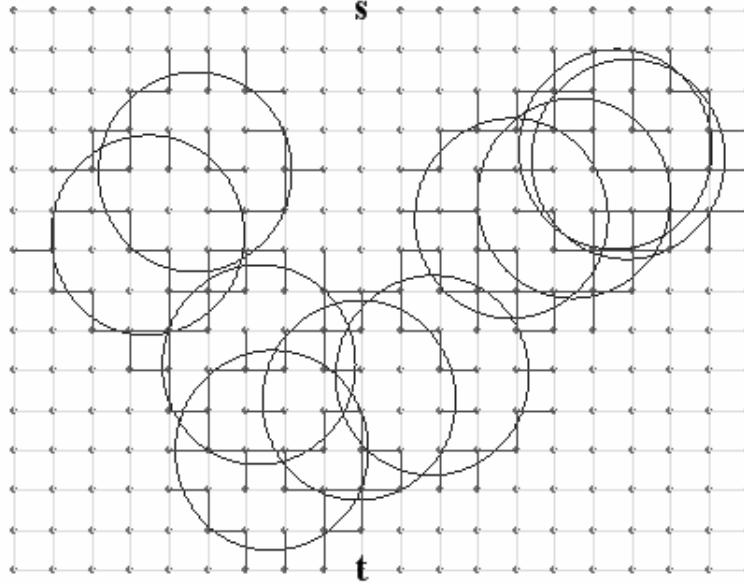


Figure 2.4: An experimental DRDP realization with $|X| = 10$.

		Mean execution time			Mean # of V^* calc's in SDP	Mean # of expansions	
		SDP	AO^*	BAO^*		AO^*	BAO^*
$ X = 7$	$K = 1$	19.0s	7.1s	5.4s	1677	119	31
	$K = 2$	4.2m	22.1m	1.9m	10,841	12,653	322
$ X = 10$	$K = 1$	—	9.0s	7.3s	—	128	41
	$K = 2$	—	33.1m	2.8m	—	15,052	386

Table 2.1: Comparison of SDP, AO^* , and BAO^* on randomly generated DRDP instances. For $|X| = 10$, SDP did not run for either K due to excessive memory requirements.

As shown in Table 2.1, mean AO^* and BAO^* runtime were less than that of SDP for $|X| = 7, K = 1$. However, since the same state can reside in multiple nodes in the AO tree when $K > 1$, the number of AO^* expansions can exceed the number of value function evaluations in SDP, causing AO^* runtime surpass SDP runtime, as

illustrated for $|X| = 7, K = 2$. On the other hand, for $K = 2$, the mean number of node expansions in BAO^* was only about 2.5% of the mean number of expansions in AO^* , for both $|X| = 7$ and $|X| = 10$. For $|X| = 10$, SDP did not run even for $K = 1$ due to excessive memory requirements, whereas BAO^* mean runtime for $K = 2$ was just 2.8 minutes, which was only 8.5% of the mean AO^* runtime, illustrating the efficiency of the BAO^* Algorithm in solving small DRDP instances.

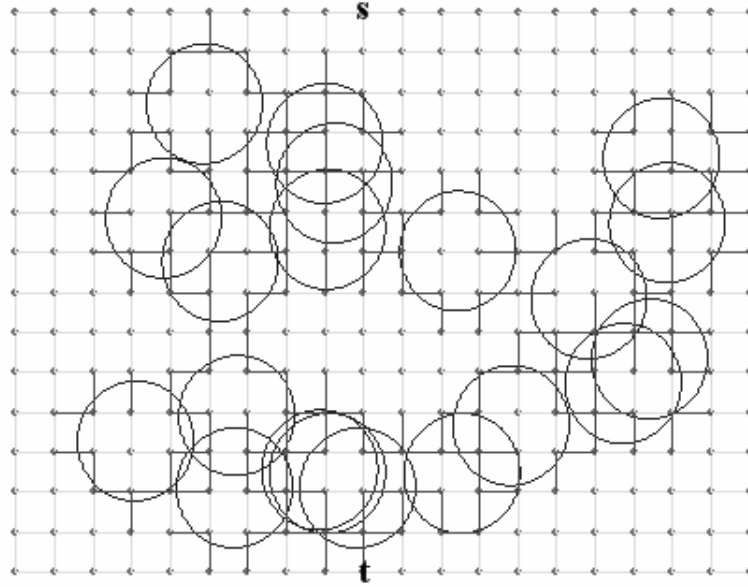


Figure 2.5: An experimental DRDP realization with $|X| = 20$.

To further illustrate the efficiency of BAO^* on relatively small DRDP instances, we conducted a second set of simulations where $i_{\max} = 20$ and $j_{\max} = 15$, with $s = (10, 15)$ and $t = (10, 1)$, as above. This time, disc radii are set to 1.5 and the disc centers are sampled from a uniform distribution on the pairs of real numbers in

$[3, 18] \times [3, 13]$. Number of discs is taken as 20 and the cost of disambiguation is taken as 1.5. Disc marks are sampled from a uniform distribution on the unit interval. The simulation environment is illustrated in Figure 2.5. For 10 experiment realizations in this environment with $K = 2$, mean BAO^* execution time was just 7.4 minutes and mean number of node expansions was 1,957. On the other hand, mean AO^* execution time was 2.7 hours whereas mean number of node expansions was 43,309. We did not perform additional simulations with $K = 3$ as one such run of BAO^* took almost 20 hours.

2.5 Partially Observable Markov Decision Process Formulation

A Partially Observable Markov Decision Process (POMDP) is a generalization of the MDP framework where the decision maker (DM) no longer has direct access to his current state, but instead, has the ability to make observations via imperfect sensors. MDPs can handle uncertainty regarding the outcomes of the DM's actions, whereas POMDPs can also cope with uncertainty arising from lack of complete state knowledge as well as uncertainty due to imperfect sensing [40, 72]. MDPs can be solved in polynomial-time [50, 77], whereas POMDPs have been shown to be intractable and are difficult to solve in general [54].

A POMDP is denoted as a 6-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, O \rangle$ where

- $\mathcal{S}, \mathcal{A}, \mathcal{T}$, and $\mathcal{R}$ denote a Markov Decision Process;
- Ω is a set of observations the DM can make; and,
- $O : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \Omega \rightarrow [0, 1]$ is the observation function: For each current state, action, and resulting state, it specifies a probability distribution over possible observations. Specifically, $O(s, \alpha, s', o)$ is the probability of observing o when the DM is at state s , chooses action α , and ends up in state s' .

As shown below, RDP can be cast as a POMDP by trimming the set of information vectors to $\{\text{"T"}, \text{"F"}\}^{|X|}$ and, folding the ambiguity of detections into ambiguity of the information vector, hence the partial observability of the state. Current POMDP technology is not sufficient to efficiently solve RDP in this manner [69], but, this formulation keeps open more options for attacking the problem in the future. In our formulation, we shall assume that cost of disambiguation is zero and there is no limit on the number of available disambiguations. The motivation for this assumption is that the reward for each state/ action pair needs to be specified a priori and, the DM may have to re-disambiguate certain detections in order to traverse the shortest path used to calculate this reward. We formulate the components of $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, O \rangle$ as follows²:

²This formulation can be generalized to continuous and discrete SDL in a relatively straightforward manner.

- **States:** The POMDP information vector is defined as $\mathcal{I}' \in \{\text{"T"}, \text{"F"}\}^{|X|}$ such that, for all $i = 1, 2, \dots, |X|$, the i -th entry of $\mathcal{I}'$ is “T” or “F”, according as the actual status of i -th detection is true or false. For each disambiguation location $y \in \mathcal{Y}$, we introduce two points: y_{out} that is an exact copy of y (called an *out-point*), and y_{in} that is y infinitesimally perturbed from its location into the interior of the associated disc (called an *in-point*)— y_{in} cannot be arrived at unless the associated detection is false. Let $\mathcal{Y}'$ be the union of s , t , and, the in- and out-points associated with each disambiguation location. The POMDP state space is defined as $\mathcal{Y}' \times \mathcal{I}'$. We shall refer to the location component of a given state as the *state point*.
- **Actions:** The set of actions is $\mathcal{Y}' \setminus \{s\}$.
- **State Transition Function:** Given a state and an action, the state point of the state transitioned into is the point as identified by the action. The information vector of state transitioned into is always the same as the information vector of the current state.
- **Rewards:** At any state whose state point is an out-point, the reward for choosing the corresponding in-point as the action is 0 if the detection associated with them is false, and $-\infty$ otherwise. For any other state/ action pair, the reward is the negative of the shortest path distance between the state point and the action avoiding all the discs (regardless of the actual status of their associated

detections) except the ones that the DM is currently inside of.

- **Observations:** The set of observations Ω is $\{\text{true}, \text{false}\}$.
- **Observation Probabilities:** Since state transitions are always deterministic, the observations only need to be specified for state/ action pairs, which are true or false according as the actual status of the detection associated with the actions.

Thanks to the assumption that the cost of disambiguation is zero and there is no limit on the number of available disambiguations, the DM can traverse through discs that have been previously disambiguated by re-disambiguating them whenever they need to be re-traversed.

Notice that in the above formulation, state transitions and observations are deterministic and, all the ambiguity is folded into the information vector. The information vector, on the other hand, represents the DM's current knowledge of the obstacle field, whose probability distribution (called the *belief state* in POMDP terminology) is specified by the disc marks.

Chapter 3

Data Structures

3.1 Introduction

In this chapter, we first discuss visibility graphs, which can be used to compute deterministic shortest paths through arrangements of definite-obstacles in the plane. Then, we introduce the tangent arc graph (TAG) data structure, which is the topological superimposition of all of the visibility graphs generated by the collection of all subsets of obstacles. Even though there are exponentially many such subsets, the TAG is polynomially-sized. The TAG plays a fundamental role in the protocols developed for RDP in the next two chapters. We then point out that the well-known A^* Algorithm with a slightly stronger admissibility requirement on the heuristic function is equivalent to Dijkstra's Algorithm under a change of variable. Such a heuristic function is available for tangent arc graphs with the various edge weights our protocols

use, and this will speed up finding shortest paths as part of computing these protocols.

3.2 The Visibility Graph

The visibility graph is a data structure that can be used to find deterministic shortest paths through arrangements of definite-obstacles in the plane. In particular, visibility graphs can be used to compute $q(s, t, X)$, the shortest zero-risk path from s to t associated with X in RDP. Our visibility graph is an adaptation of the visibility graph from [47] and [66], more similar to the generalized visibility graph in [65]. The visibility graph is also used in the formation of the tangent arc graph, another important data structure presented in the next section.

Let s, t and X be specified. For any distinct points $a, b \in \{s, t\} \cup \partial(\cup_{x \in X} R_x)$, we call the closed line segment $\overline{a, b}$ a *tangent segment* provided that (1) for all $r \in \{a, b\} \setminus \{s, t\}$, $\overline{a, b}$ is tangential to $\partial(\cup_{x \in X} R_x)$ at r and (2) the relative interior of $\overline{a, b}$ is contained in the interior of $[(\cup_{x \in X} R_x) \cup \{s, t\}]^C$.

The *visibility graph* associated with s, t, X is defined as follows. Its vertex set consists of s, t , all points of $\partial(\cup_{x \in X} R_x)$ which intersect a tangent segment, and all points of $\partial(\cup_{x \in X} R_x)$ at which two or more ∂R_x 's intersect. The edge set of the visibility graph consists of all tangent segments and all connected components of $\partial(\cup_{x \in X} R_x)$

after the vertices of the visibility graph are removed (the latter edges are segments of arc from circles). The graph theoretic endpoints of these edges are their line and arc endpoints, respectively, and each edge is weighted with its arclength. An example of a visibility graph is pictured in Figure 3.1.

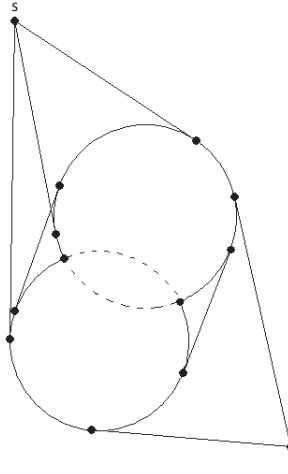


Figure 3.1: An example of a visibility graph. The dashed arcs are not visibility graph edges since they are not in $\partial(\cup_{x \in X} R_x)$.

It is a well-known (and true) folk theorem that $q(s, t, X)$ is the shortest s, t path in the visibility graph associated with s, t, X . Since every pair of nonidentical ∂R_x 's have at most 4 mutually tangential lines and two points of intersection, the number of vertices and edges in the visibility graph are each $O(|X|^2)$. Thus Dijkstra's algorithm with a heap implementation [1] applied to this visibility graph yields $q(s, t, X)$ in $O(|X|^2 \log |X|)$ operations. For each pair of nonidentical ∂R_x 's, $O(|X|)$ ∂R_x 's need to be checked for intersection with the tangential lines associated with this pair, thus

the naive construction of the visibility graph can be accomplished in $O(|X|^3)$ assignment, arithmetic, and trigonometric operations.

3.3 The Tangent Arc Graph

The tangent arc graph (TAG) is a new data structure comprised of the topological superimposition of all the visibility graphs generated by the collection of all subsets of disc-shaped obstacles. Although there are exponentially many such subsets (hence exponentially many such generated visibility graphs), the TAG is polynomially-sized, as described below.

Given s, t, X , and ρ , for any distinct points $a, b \in \{s, t\} \cup (\bigcup_{x \in X} \partial R_x)$, we say the closed line segment $\overline{a, b}$ is a *general tangent segment* provided that, for all $r \in \{a, b\} \setminus \{s, t\}$, $\overline{a, b}$ is tangential to ∂R_x for some $x \in X$. The TAG is defined as follows: The vertex set of $\text{TAG}_{s,t,X}$ consists of s, t , all points of intersection between any general tangent segment and any ∂R_x (over all $x \in X$), and all points of intersection between two or more ∂R_x 's. The edge set of $\text{TAG}_{s,t,X}$ consists of all connected components of all general tangent segments after the vertices of $\text{TAG}_{s,t,X}$ are removed, and all connected components of $\bigcup_{x \in X} \partial R_x$ after the vertices of $\text{TAG}_{s,t,X}$ are removed.

An example of a $\text{TAG}_{s,t,X}$ is pictured in Figure 3.2. In this example, there are 11

general tangent segments; in particular, one of these is the line segment $\overline{s,t}$, which intersects boundaries of discs in two places. These two intersections become vertices in $\text{TAG}_{s,t,X}$ and, upon removal from $\overline{s,t}$, three $\text{TAG}_{s,t,X}$ edges are created in $\overline{s,t}$'s place. In total, this $\text{TAG}_{s,t,X}$ has 21 vertices and 37 edges; 16 of these edges are line segments and 21 of these edges are segments of arc from circles.

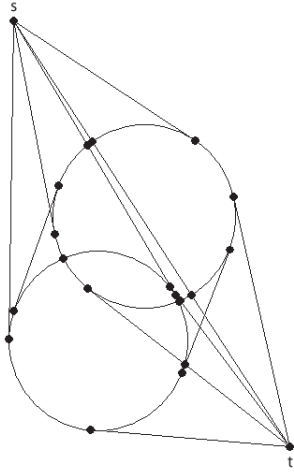


Figure 3.2: An example of a tangent arc graph.

Note that the graph $\text{TAG}_{s,t,X}$ is a topological superimposition of all the visibility graphs generated by s, t, Y over all $Y \subseteq X$. There are $O(|X|^2)$ general tangent segments, each intersecting $O(|X|)$ regions of the form $R_x : x \in X$, so we have $O(|X|^3)$ vertices and $O(|X|^3)$ edges in $\text{TAG}_{s,t,X}$. Thus, shortest paths in TAG may be found in $O(|X|^3 \log |X|)$ operations by using a heap implementation of Dijkstra's Algorithm. Since the $O(|X|)$ regions intersecting the general tangent segments can be sorted in $O(|X| \log |X|)$, the number of operations to set up TAG is $O(|X|^3 \log |X|)$.

An important feature of TAG is that any shortest path in the plane restricted to avoid any subset of the original obstacles will automatically be in TAG regardless of which subset of obstacles are to be avoided. Consequently if, rather than obstacles being untraversable, there is an additive "penalty" (added to traversal length) for traversing the respective obstacles, then a shortest path in this setting will also be in TAG.

3.4 The A^* and Dijkstra's Algorithms

The protocols we introduce for RDP in the next two chapters involve finding deterministic shortest paths in tangent arc graphs with various weights on the edges, which can be computed by the well-known (heap implementation of) Dijkstra's Algorithm. When it can be used, the A^* Algorithm [34, 35, 52] is an alternative to Dijkstra's Algorithm, which has the same worst-case complexity, but empirically runs faster. In A^* , the guidance of a *heuristic function*, which roughly reflects the distance from the respective vertices to the destination, helps to guide the algorithm to make a more goal-oriented search than the search performed in Dijkstra's Algorithm. In the most general case, A^* is not guaranteed to terminate with an optimal solution, but if the heuristic function is *admissible*, i.e., never overestimates the actual shortest distance, then A^* indeed terminates with an optimal solution (see, e.g., [52]).

In the setting of the tangent arc graph, which is embedded in the plane and where edge weights are never less than the Euclidean distance between the edges' endpoints, an admissible heuristic function is the Euclidean distance from every vertex to t (“as the crow flies,” in the absence of any and all obstacles). Moreover, this heuristic function is *valid* in a fashion defined later in this section. In this case, the A^* Algorithm can be coded much more simply as re-marking of vertices is never necessary. In fact, we point out below that in this case A^* is equivalent to Dijkstra’s Algorithm under a change of variable, and this provides a neat and quick way to see that A^* would always terminate with an optimal solution. This observation that under validity of the heuristic function A^* is Dijkstra’s Algorithm with a change of variable is not new, but is not explicitly discussed in the literature. To our knowledge, the only reference available is an exercise problem in [1].

Specifically, given a network $G = (\mathbf{N}, \mathbf{A})$, a starting node $s \in \mathbf{N}$, a destination node $t \in \mathbf{N}$, and a nonnegative length function $\ell : \mathbf{A} \rightarrow \mathbb{R}_{\geq 0}$, we seek an s, t path P of shortest length $\ell(P) := \sum_{e \in \mathbf{A}(P)} \ell(e)$. Dijkstra’s Algorithm grows a set F of *marked* nodes and also tunes distance labels $d : \mathbf{N} \rightarrow \mathbb{R}_{\geq 0}$ while maintaining the invariant that, for all $v \in \mathbf{N}$, $d(v)$ is the shortest s, v path distance when these paths are restricted to the nodes $F \cup \{v\}$ and, furthermore, if $v \in F$, then such a shortest restricted path is also a global shortest s, v path. For all $v \in \mathbf{N}$, $\text{PRED}(v)$ is the immediate

predecessor of v on the current s, v path with length $d(v)$ and, at the conclusion of the algorithm, the shortest s, t path is $s, \dots, \text{PRED}(\text{PRED}(t)), \text{PRED}(t), t$ and has length $d(t)$. Pseudocode for Dijkstra's Algorithm is presented in Figure 3.3.

Algorithm Dijkstra

Input: Network $G = (\mathbf{N}, \mathbf{A})$, start node $s \in \mathbf{N}$ and termination node $t \in \mathbf{N}$, arc lengths $\ell : \mathbf{A} \rightarrow \mathbb{R}_{\geq 0}$.

Output: Shortest path from s to t , if it exists.

Begin

 Initialize $F := \emptyset$

 For all $v \in \mathbf{N}$, initialize $d(v) := 0$ or ∞ , according as $v = s$ or not

 While $t \notin F$

 Select $x \in \arg \min_{v \in F^c} d(v)$

 Put x in F

 For all $w \in F^c : (x, w) \in \mathbf{A}$

 If $d(w) > d(x) + \ell(x, w)$

 Set $d(w) := d(x) + \ell(x, w)$

 Set $\text{PRED}(w) := x$

 End if

 End for

 End while

End

Figure 3.3: Dijkstra's Algorithm.

A *heuristic function* $h : \mathbf{N} \rightarrow \mathbb{R}$ (which is meant to be a rough approximation of the shortest path distance from the respective nodes to t) is said to be *valid* if $h(t) = 0$ and, for all $(u, v) \in \mathbf{A}$, $h(u) \leq h(v) + \ell(u, v)$. This latter inequality should be seen as a triangle inequality, and if h is valid then, for all $v \in \mathbf{N}$, $h(v)$ turns out to be a lower bound on the shortest v, t path distance. This can be seen as follows: Let $P = v_0, v_1, \dots, v_k$ be a shortest v, t path. Then, by validity of h ,

$h(v) = h(v_0) = \sum_{i=0}^{k-1} (h(v_i) - h(v_{i+1})) \leq \sum_{i=1}^{k-1} \ell(v_i, v_{i+1}) = \ell(P)$, which is what we wanted to show.

Given a valid heuristic function h , the A^* Algorithm has the identical pseudocode as Dijkstra's algorithm above, just substituting the line "Select $x \in \arg \min_{v \in F^c} d(v)$ " with the line "Select $x \in \arg \min_{v \in F^c} (d(v) + h(v))$ ". We next show that A^* is precisely Dijkstras Algorithm under a change of variable.

Suppose we are given a heuristic function h . For each arc $(u, v) \in A$, we define the *reduced length* $\ell^h(u, v) := \ell(u, v) + h(v) - h(u)$. Clearly, h is valid if and only if all reduced lengths are nonnegative (and $h(t) = 0$). For any s, t path P we have $\ell^h(P) = \ell(P) + h(t) - h(s)$, thus finding a shortest s, t path relative to ℓ^h is equivalent to finding a shortest s, t path relative to ℓ ; when found, just adjust the length of the shortest s, t path by the constant $h(t) - h(s)$. Indeed, supposing h is valid, running Dijkstra's algorithm¹ with ℓ^h in place of ℓ can be a very good idea if h is a "good" heuristic; specifically, if for each arc (u, v) along the shortest s, t path we have $h(u) \approx h(v) + \ell(u, v)$ then, equivalently, $\ell^h(u, v) \approx 0$ and thus the shortest s, t path distance relative to ℓ^h is very small, which will mean that t will then be quickly placed in F .

¹Note that here we do not need to restrict ℓ to be nonnegative, since Dijkstra's Algorithm will work relative to ℓ^h when ℓ^h is nonnegative. We just require that there be no negative length cycles in G .

While running Dijkstra's Algorithm to find a shortest s, t path relative to ℓ^h , let us dynamically denote, for all $v \in \mathbf{N}$, $f(v) :=$ relative-to- ℓ shortest s, v path distance when these paths are restricted to $F \cup \{v\}$; as mentioned before, $d(v) = f(v) + h(v) - h(s)$. Observe that, since $h(s)$ is fixed, the objective function of the arg min in this run of Dijkstra's Algorithm can be viewed as being $f(v) + h(v)$. Then, three lines later when this run of Dijkstra's Algorithm checks if $d(w) > d(x) + \ell^h(x, w)$ (with the consequence of setting LHS equal to RHS), Dijkstra's Algorithm can be seen as checking if $f(w) + h(w) - h(s) > f(x) + h(x) - h(s) + \ell(x, w) + h(w) - h(x)$, in other words it is checking if $f(w) > f(x) + \ell(x, w)$ (with the effective consequence of setting LHS equal to RHS). We thus can see that f is—at all times and for all vertices—precisely what d would have been during a run of the A* algorithm with length function ℓ and heuristic h , and the order that this run of Dijkstra's Algorithm places nodes in F is the same as the order in the run of the A* Algorithm with length function ℓ and heuristic h . We summarize this finding in the following theorem:

Theorem 1. *Suppose $G = (\mathbf{N}, \mathbf{A})$ is a network, $s, t \in \mathbf{N}$, and $\ell : \mathbf{A} \rightarrow \mathbb{R}$ is such that there are no negative length cycles in G , and suppose that $h : \mathbf{N} \rightarrow \mathbb{R}$ is a valid heuristic. If we run the A* Algorithm for this problem, then, at all times and for all $v \in \mathbf{N}$, the distance label $d(v)$ in this run of the A* Algorithm will always be the relative-to- ℓ shortest s, v path distance among paths restricted to the nodes $F \cup \{v\}$, and it will also hold that the order that nodes are placed in F is the same order as the*

nodes would have been placed in F running Dijkstra's Algorithm using the lengths ℓ^h .

In particular, A^* will terminate with the correct shortest s, t path and s, t path-length.

Chapter 4

The Simulated Risk

Disambiguation Protocol

4.1 Introduction

This chapter introduces the simulated risk (SR) disambiguation protocol; a suboptimal but efficiently computable and effective algorithm for RDP and DRDP [29]. This protocol temporarily pretends (for the sole purpose of choosing the next disambiguation location) that all discs are riskily traversable. Then, a chosen “undesirability function” is used (for each such possible traversal) to combine length and risk into a single measure of traversal undesirability. A shortest traversal in this undesirability sense is what the DM will then traverse until the first ambiguous disc is encountered, at which point a disambiguation is performed, the problem data is updated accord-

ingly, and this procedure is iteratively repeated until arrival at the destination.

We discuss how to efficiently implement SR protocols based on linear undesirability functions, and describe how the tangent arc graph can be used to simplify the implementation of the SR protocol for RDP. We then adapt the basic strategy of the SR protocol to DRDP and present computational experiments illustrating the effectiveness of the SR protocol.

4.2 The Simulated Risk Disambiguation Protocol

In our framework, the decision maker (DM) will never enter discs of the form $R_x : x \in X$ while they are possibly forbidden, and thus actual risk will never be experienced. However, for the purpose of deciding the next disambiguation location, the simulated-risk disambiguation protocol temporarily pretends (simulates) that the possible-obstacles are riskily traversable.

Under this simulation of risk, for any s, t curve p (allowing intersection with $\bigcup_{x \in X} R_x$), its Euclidean length $\ell^e p$ is defined in the usual way, and its *risk length* is defined as

$$\ell^r p := -\log \prod_{x \in X: p \cap R_x \neq \emptyset} (1 - \rho(x)).$$

This negative logarithm of the probability that p is permissibly traversable is a measure of the risk in traversing p —if the DM were willing to take on risk. An *undesirability function* is any function $g : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ which is monotonically nondecreasing in its arguments. That is to say, for all $u_1, u_2, v_1, v_2 \in \mathbb{R}_{\geq 0}$ such that $u_1 \leq u_2$ and $v_1 \leq v_2$, it holds that $g(u_1, v_1) \leq g(u_2, v_2)$. The number $g(\ell^e p, \ell^r p)$ is thought of as a measure of the undesirability of p in the sense that, if the DM were required to traverse from s to t under the simulation of risk and without a disambiguation capability, the DM would want to traverse the s, t curve $\phi_g := \arg \min_{s, t \text{ curves } p} g(\ell^e p, \ell^r p)$. For this s, t curve ϕ_g , let $y \in \mathbb{R}^2$ be the last point of ϕ_g before ϕ_g intersects $\bigcup_{x \in X} R_x$, and say $x' \in X$ is the detection whose associated region $R_{x'}$ the curve ϕ_g was entering at y . (If there is no intersection between ϕ_g and $\bigcup_{x \in X} R_x$ then $y := t$.) Back in our setting (where there is a disambiguation capability and the DM may not experience risk), the simulated risk disambiguation protocol $\mathcal{D}_g$ is defined as assigning this x' and y to s, t, X, ρ, K (provided that $K > 0$).

Thus, the simulated risk random disambiguation path $p_{\mathcal{D}_g}$ follows a shortest s, t curve (in the sense of g , and under the simulation that potentially forbidden disks are riskily traversable) until it encounters an ambiguous region (which, in actuality, it can not enter without disambiguation), at which point a disambiguation is performed, and the whole process is repeated using the current location in place of s (and the updated information on ρ, X , and K).

Note that for the particular undesirability function $g(u, v) = u + \infty_{v>0}$ (where $\infty_{v>0}$ denotes ∞ or 0 , according as $v > 0$ or $v = 0$) it holds that $p_{\mathcal{D}_g} = q(s, t, X)$, which is the s, t curve the DM would traverse if the DM did not have the disambiguation capability (and the DM were still not permitted to take any risk). In Section 4.4 we will discuss how, in practice, the DM would select an undesirability function g to use. We advocate choosing—from among a specific family of undesirability functions—the undesirability function whose associated random disambiguation path has minimum expected length. As long as this family of undesirability functions the DM chooses from also includes the function $g(u, v) = u + \infty_{v>0}$ (which is indeed¹ the case for the strategy we advocate in Section 4.4) the DM is then guaranteed to do no worse (in expectation) than the s, t curve the DM would follow if the DM did not have a disambiguation capability (and the DM was not permitted any risk).

Given an undesirability function g , in order to evaluate $\mathcal{D}_g$ and to realize $p_{\mathcal{D}_g}$ one must be able to compute $\phi_g := \arg \min_{s, t \text{ curves } p} g(\ell^e p, \ell^r p)$. Although there are uncountably infinitely many s, t curves over which to minimize, the monotonicity of g can be used to show that ϕ_g must be a path in the tangent arc graph $\text{TAG}_{s, t, X}$ defined in Section 3.3, so that ϕ_g solves the finite optimization problem $\min_{s, t \text{ paths } p \text{ in } \text{TAG}_{s, t, X}} g(\ell^e p, \ell^r p)$. Specifically, if ϕ_g is the shortest s, t curve in the sense of a function g , then for $Y = \{x \in$

¹The disambiguation protocol $\mathcal{D}_g$ associated with $g(u, v) = u + \infty_{v>0}$ is precisely the protocol $\mathcal{D}_\alpha$ discussed later in this chapter with $\alpha = \infty$.

$X : \phi_g \cap R_x = \emptyset\}$ we have that ϕ_g is a path in the visibility graph associated with s, t, Y . Thus, in particular, ϕ_g is a path in $\text{TAG}_{s,t,X}$.

4.2.1 Linear Undesirability Functions

The simplest undesirability functions are the linear ones, where $g(u, v) = u + \alpha \cdot v$ for some fixed parameter $\alpha \geq 0$; in this case we abbreviate the disambiguation protocol $\mathcal{D}_g$ to $\mathcal{D}_\alpha$. We next show that for any fixed $\alpha \geq 0$, $\mathcal{D}_\alpha$ is efficiently computable and $p_{\mathcal{D}_\alpha}$ is efficiently realizable.

For each edge $f = \{a, b\}$ in $\text{TAG}_{s,t,X}$, we assign its Euclidean length $\ell^e f$ in the usual manner, and we define its risk length as

$$\ell^r f := - \sum_{x \in X} \delta_{f \cap R_x \neq \emptyset} \left(\frac{\delta_{a \in \partial R_x}}{2} + \frac{\delta_{b \in \partial R_x}}{2} \right) \log(1 - \rho(x))$$

where δ is the indicator function; this definition is consistent with that of risk length for an s, t curve since, for any s, t path p in $\text{TAG}_{s,t,X,\rho}$, it holds that $\ell^r p = \sum_{\text{edges } f \in p} \ell^r f$ (provided that p never revisits any region R_x twice). Thus ϕ_g may be found by running Dijkstra's Algorithm on $\text{TAG}_{s,t,X}$ using the lengths $\ell^e f + \alpha \cdot \ell^r f$ for each edge f in $\text{TAG}_{s,t,X}$. The running time for Dijkstra's algorithm with a heap implementation is $O(|X|^3 \log |X|)$, so $\mathcal{D}_\alpha$ can be computed in $O(|X|^3 \log |X|)$ operations, and $p_{\mathcal{D}_\alpha}$ is

thus realizable in $O(|X|^3 \log |X|)$ operations since K is a constant. (In particular, if there was no limit K on the number of disambiguations permitted, then $p_{\mathcal{D}_\alpha}$ is realizable in $O(|X|^4 \log |X|)$ operations.)

4.3 Mine Countermeasures Example

Minefield detection and localization is an important problem currently receiving much attention in the scientific and engineering literature; see, for instance, [70] and the references cited there. Witherspoon *et al.* [76] depict the operational concept for minefield reconnaissance via an unmanned aerial vehicle. Multispectral imagery of an area of interest is processed and a mine detection algorithm identifies locations of potential mines (see [39]), the collection of these points constituting our point process realization. The marks are posterior probabilities that the respective detections represent actual mines, as rendered by a post-processing classification rule [53], [57], [56], [61], [62].

The following marked point process realization, shown in Figure 4.1, is referred to in [53] and [62], and has 39 potential mines whose x - and y -coordinates are listed in Table 4.1. The associated marks ρ in the table were generated by the post-classification rule in [53]. Each disc R_x has radius 50, s is the point $(0, 800)$, and t is the point $(0, 100)$. Suppose a maximum of $K = 4$ disambiguations may be performed.

x-coordinates	y-coordinates	ρ	x-coordinates	y-coordinates	ρ	x-coordinates	y-coordinates	ρ
321.17	158.27	.59017	54.23	201.12	.54178	158.17	516.48	.43525
215.13	428.31	.61890	-145.67	703.06	.61714	-151.01	572.15	.56076
221.12	557.31	.64047	-166.36	299.42	.49173	296.16	163.31	.11649
163.31	186.14	.65636	28.31	205.03	.15269	-79.26	709.99	.56085
100.40	376.47	.51487	-105.75	262.20	.25748	185.31	182.18	.65266
116.39	110.84	.44124	-128.60	274.12	.62001	-61.19	345.12	.17183
-91.27	664.45	.16675	-82.87	248.29	.58308	105.47	509.80	.85147
-19.93	568.04	.59937	-310.23	402.92	.65428	-320.73	532.23	.33092
-35.11	242.61	.10330	-169.99	438.90	.64163	95.39	248.12	.18868
-78.75	396.14	.07310	-245.28	372.05	.52154	-166.45	180.33	.61082
-134.53	769.27	.19386	-258.45	641.03	.65670	111.60	640.10	.56529
-219.32	313.68	.57449	-455.72	742.57	.63987	-157.10	441.96	.64444
-242.22	321.51	.65655	-237.86	546.19	.13793	-269.98	379.65	.52802

Table 4.1: x,y-coordinates and ρ 's for marked point process realization.

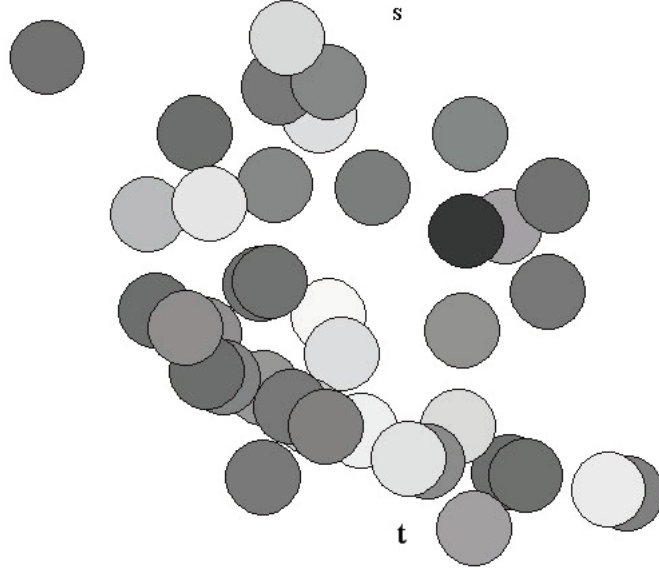


Figure 4.1: Marked point process realization. Gray-scale of discs reflects ρ of detections at respective disc's centers.

For $\alpha = 2000$, the first disambiguation is at the point $(-10.42, 286.09)$ and, regardless of the result, no more disambiguations are performed; thus, there are only two possible realizations of $p_{D_{2000}}$, and they are pictured in Figure 4.2, along with their respective Euclidean lengths and their respective probabilities. In particular,

$\text{El}^e p_{\mathcal{D}_{2000}} = .89670 \cdot (707.97 + 1 \cdot c) + .10330 \cdot (1116.19 + 1 \cdot c) = 750.14 + c$, where c is the disambiguation cost.

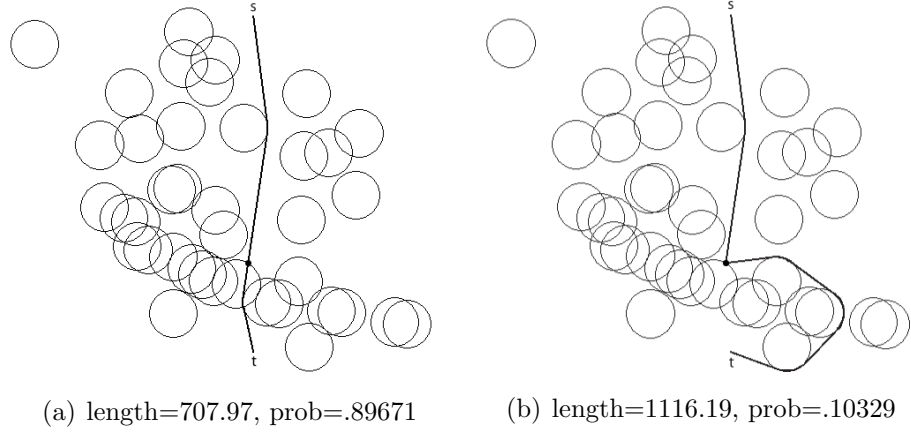


Figure 4.2: All (two) possible realizations of $p_{\mathcal{D}_{2000}}$.

For $\alpha = 100$, all seven possible realizations of $p_{\mathcal{D}_{100}}$ are pictured in Figure 4.3, labelled with their respective Euclidean lengths and probabilities. In particular, we can compute $\text{El}^e p_{\mathcal{D}_{100}} = .89671 \cdot (707.97 + 1 \cdot c) + .040105 \cdot (714.90 + 3 \cdot c) + .038472 \cdot (859.37 + 4 \cdot c) + .012796 \cdot (831.04 + 3 \cdot c) + .0089469 \cdot (1188.77 + 4 \cdot c) + .0019532 \cdot (1185.40 + 4 \cdot c) + .0010226 \cdot (958.43 + 4 \cdot c) = 721.14 + 1.2570 c$.

Note that the coefficient of c in $\text{El}^e p_{\mathcal{D}_{\alpha}}$ is the expected number of disambiguations. Also observe that $\text{El}^e p_{\mathcal{D}_{100}} < \text{El}^e p_{\mathcal{D}_{2000}}$ (*i.e.*, $\alpha = 100$ is a better choice than $\alpha = 2000$) precisely when $c < 112.80$.

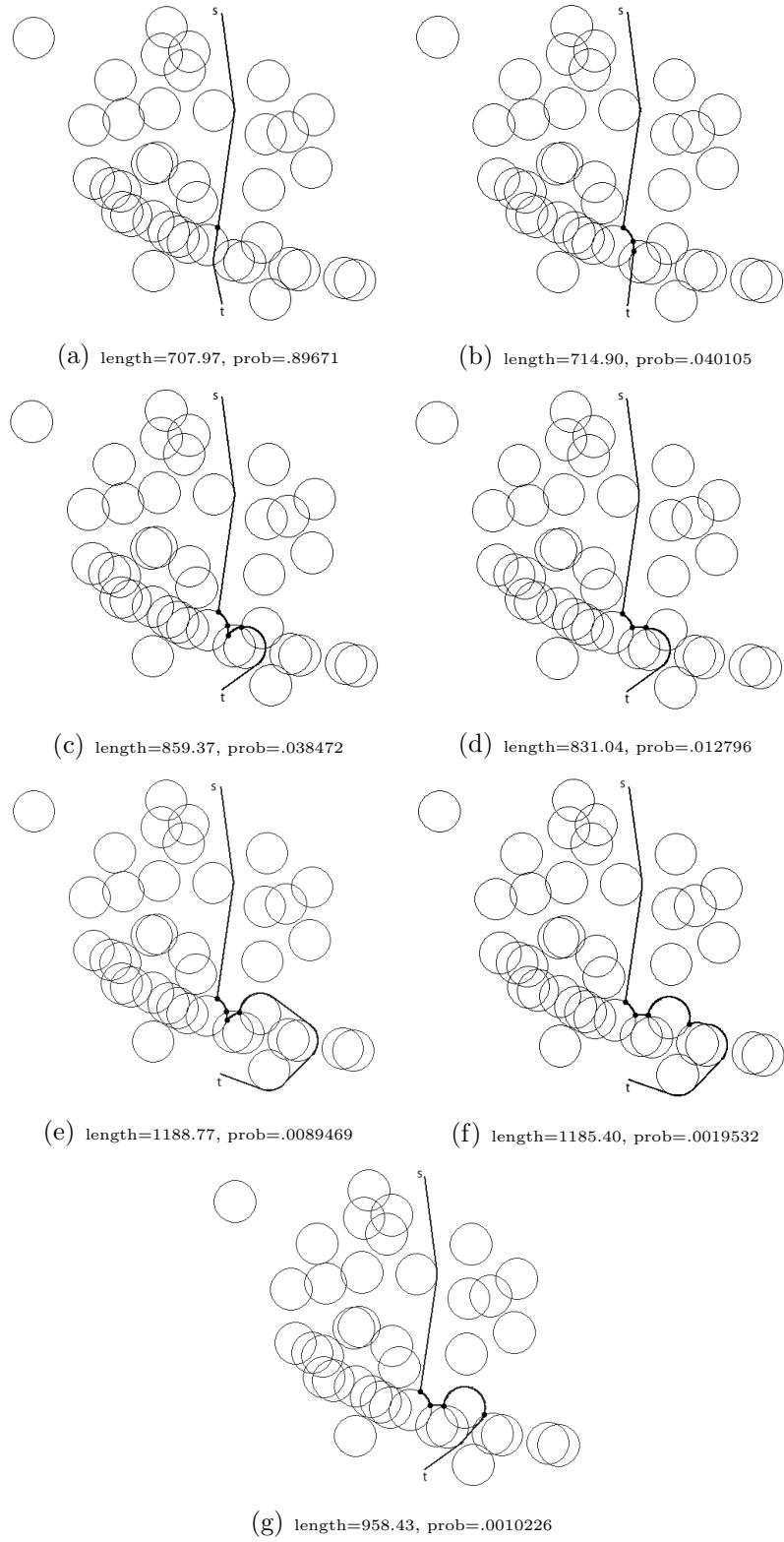


Figure 4.3: All (seven) possible realizations of $p_{\mathcal{D}_{100}}$. The probabilities are the respective probabilities of the path realizations.

In practice, the DM will want to select a nonnegative value for α of minimum $\text{El}^e p_{\mathcal{D}_\alpha}$. In Figure 4.4, we plot $\text{El}^e p_{\mathcal{D}_\alpha}$ as a function of α for the specific cost $c = 5$; here $\arg \min_{\alpha \geq 0} \text{El}^e p_{\mathcal{D}_\alpha}$ is seen to be the interval $(30.23, 55.09)$. It turns out that $p_{\mathcal{D}_\alpha}$ is, in fact, identical to $p_{\mathcal{D}_{50}}$ for all values of α in $(30.23, 55.09)$; a maximal interval I where $p_{\mathcal{D}_\alpha}$ is identical for all values of α in I will be called an *indifference interval*². Here, in total, there are 11 indifference intervals, listed in Table 4.2 with their respective values of $\text{El}^e p_{\mathcal{D}_\alpha}$ and the range of costs c where that interval is precisely $\arg \min_{\alpha \geq 0} \text{El}^e p_{\mathcal{D}_\alpha}$.

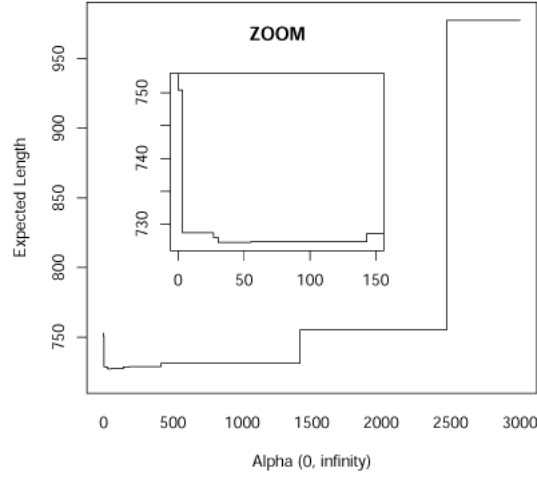


Figure 4.4: $\text{El}^e p_{\mathcal{D}_\alpha}$ as a function of α , for $c = 5$.

In other words, suppose the DM was presented—in practice—the specific X and

²Note that the specific value of c is not relevant to $\mathcal{D}_\alpha$ nor $p_{\mathcal{D}_\alpha}$, so c has no influence in the establishment of indifference intervals. However, c will affect $\ell^e p_{\mathcal{D}_\alpha}$, $\text{El}^e p_{\mathcal{D}_\alpha}$, and $\arg \min_{\alpha \geq 0} \text{El}^e p_{\mathcal{D}_\alpha}$, so c will influence our choice of α .

ρ given in Table 4.1, with $s = (0, 800)$, $t = (0, 100)$, $K = 4$, and some disambiguation cost $c \geq 0$. If the DM would choose to traverse from s to t via a random disambiguation path based on a simulated risk disambiguation protocol using a linear undesirability function, then the particular value of the parameter α the DM should select depends on the particular disambiguation cost c . For example, if $c \in (0.0000, 4.1013)$ then, by comparing (for the various indifference intervals) $\text{El}^\ell p_{\mathcal{D}_\alpha}$ in the second column of Table 4.2 (as linear functions in c), it is clear that, for $c \in (0.0000, 4.1013)$, the fourth indifference interval listed there is best, and the DM should use any $\alpha \in (26.77, 30.23)$ in practice. If $c \in (4.1013, 15.9145)$ then by similar reasoning the DM should use any $\alpha \in (30.23, 55.09)$; see the third column in Table 4.2. Also in a similar fashion, for each possible value of $c \geq 0$, consider such α minimizing $\text{El}^\ell p_{\mathcal{D}_\alpha}$; the values of $\min_{\alpha \geq 0} \text{El}^\ell p_{\mathcal{D}_\alpha}$ are plotted as a (piece-wise linear) function of cost c in Figure 4.5.

Now, observe that for all $c < 228.1639$ it holds that $\min_{\alpha \geq 0} \text{El}^\ell p_{\mathcal{D}_\alpha} < \text{El}^\ell p_{\mathcal{D}_\infty} = \ell^e q(s, t, X)$, which means that, for disambiguation costs less than 23.34% of $\ell^e q(s, t, X) = 977.54$, the optimal simulated-risk disambiguation path based on a linear undesirability function yields a strict (expected) improvement over the curve $q(s, t, X)$ which would be traversed if the disambiguation capability was not available at all and risk was not permitted. Next, in Section 4.4, we address the issue of how, in general, to select an optimal or near-optimal value for α .

Indifference interval I	$\text{El}^e p_{\mathcal{D}_\alpha}$ for $\alpha \in I$	Range of c such that $I = \arg \min_{\alpha \geq 0} \text{El}^e p_{\mathcal{D}_\alpha}$
(0.00, 0.05)	$735.02 + 3.5587 c$	$\emptyset$
(0.05, 3.23)	$734.89 + 3.1033 c$	$\emptyset$
(3.23, 26.77)	$717.92 + 2.1665 c$	$\emptyset$
(26.77, 30.23)	$717.22 + 2.1665 c$	(0.0000, 4.1013)
(30.23, 55.09)	$720.89 + 1.2698 c$	(4.1013, 15.9145)
(55.09, 143.21)	$721.14 + 1.2570 c$	$\emptyset$
(143.21, 186.72)	$722.92 + 1.1423 c$	(15.9145, 50.4720)
(186.72, 413.71)	$723.07 + 1.1393 c$	(50.4720, 76.0167)
(413.71, 1414.41)	$725.81 + 1.1033 c$	(76.0167, 228.1639)
(1414.41, 2472.44)	$750.14 + 1.0000 c$	$\emptyset$
(2472.44, ∞)	977.54	(228.1639, ∞)

Table 4.2: All indifference intervals, the $\text{El}^e p_{\mathcal{D}_\alpha}$ for values of α in the respective indifference intervals, and the range of disambiguation costs c where this indifference interval is optimal. For example, for any value $0 < c < 4.1013$ the optimal value of α is any $26.77 < \alpha < 30.23$ and, as such, $\text{El}^e p_{\mathcal{D}_\alpha} = 717.22 + 2.1665 c$.

4.4 Minimizing $\text{El}^e p_{\mathcal{D}_\alpha}$ over $\alpha \geq 0$

Given s, t, X, ρ, K and assuming the DM will use a linear undesirability function in the establishment of a simulated risk disambiguation protocol, the DM still needs to determine the value of the parameter α to use; once the value of α is chosen then the DM can efficiently realize $p_{\mathcal{D}_\alpha}$, as described in Section 4.2.1. Thus, what is needed is a practical way to minimize $\text{El}^e p_{\mathcal{D}_\alpha}$ over $\alpha \geq 0$, exactly or approximately.

As a first step, suppose it is desired to evaluate $\text{El}^e p_{\mathcal{D}_\alpha}$ for just one particular value of α . This may be accomplished by considering all possible outcomes of the disambigua-

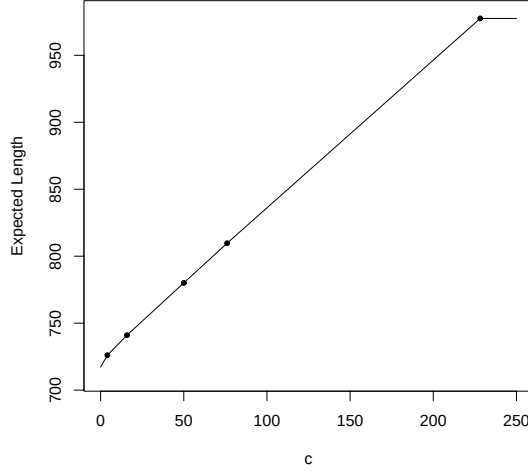


Figure 4.5: $\text{El}^e p_{\mathcal{D}_\alpha}$ for the optimal α , as a function of c .

tions dictated by $\mathcal{D}_\alpha$ and encountered by $p_{\mathcal{D}_\alpha}$ (which can be done via straightforward recursion), and then weighting the lengths of the possible s, t curves that $p_{\mathcal{D}_\alpha}$ can assume by their respective probabilities. Indeed, we used this procedure to compute $\text{El}^e p_{\mathcal{D}_\alpha}$ in Section 4.3.

The number of different s, t curves that can be realized by $p_{\mathcal{D}_\alpha}$ is bounded by 2^K . Note that this is just an upper bound; in the example of Section 4.3, where K was 4, the number of different s, t curves that could be realized by $p_{\mathcal{D}_\alpha}$ for α in the 11 different indifference intervals were, respectively, 13, 12, 9, 9, 8, 7, 6, 5, 3, 2, and 1. In particular, since K is fixed, $\text{El}^e p_{\mathcal{D}_\alpha}$ can be efficiently evaluated for a particular α ; the time required is $O(|X|^3 \log |X|)$, with a multiplicative constant depending on K .

When K is small, it can be practical to evaluate $\text{El}^e p_{\mathcal{D}_\alpha}$ for a particular α in this manner.

Therefore, it may be practical to compute $\text{El}^e p_{\mathcal{D}_\alpha}$ for a mesh of different values of α , and to then adopt the best value α' from the mesh as our parameter, hoping that $\text{El}^e p_{\mathcal{D}_{\alpha'}} \approx \min_{\alpha: \alpha \geq 0} \text{El}^e p_{\mathcal{D}_\alpha}$.

To illustrate, we obtained eleven realizations of a particular marked spatial point process on $[0, 55] \times [0, 220] \subseteq \mathbf{R}^2$ where the number of detections is $\text{Poisson}(70)$ and the marks are $\text{Uniform}(0, 1)$. We adopted the starting point $s = (-11, 110)$, destination point $t = (66, 110)$, disc radius $r = 10$, disambiguation cost $c = 1$, and the number of available disambiguations $K = 4$. Two of these eleven point process realizations had nearly unobstructed s, t paths and were therefore trivial; the other nine realizations are pictured in Figure 4.6. The discs in Figure 4.6 are gray-scaled to reflect the marks of the associated detections; discs are darker and lighter according as the marks are closer to 1 and 0.

In each of these nine nontrivial realizations, we computed $\text{El}^e p_{\mathcal{D}_\alpha}$ for $\alpha = 2, 7, 12, 17, \dots$ until the values of α are large enough so that no disambiguations are performed, i.e. until the values of α are large enough so that $p_{\mathcal{D}_\alpha} = p_{\mathcal{D}_\infty}$. Figure 4.7 shows the plots of $\text{El}^e p_{\mathcal{D}_\alpha}$ against α for each of these nine realizations. From among the mesh of

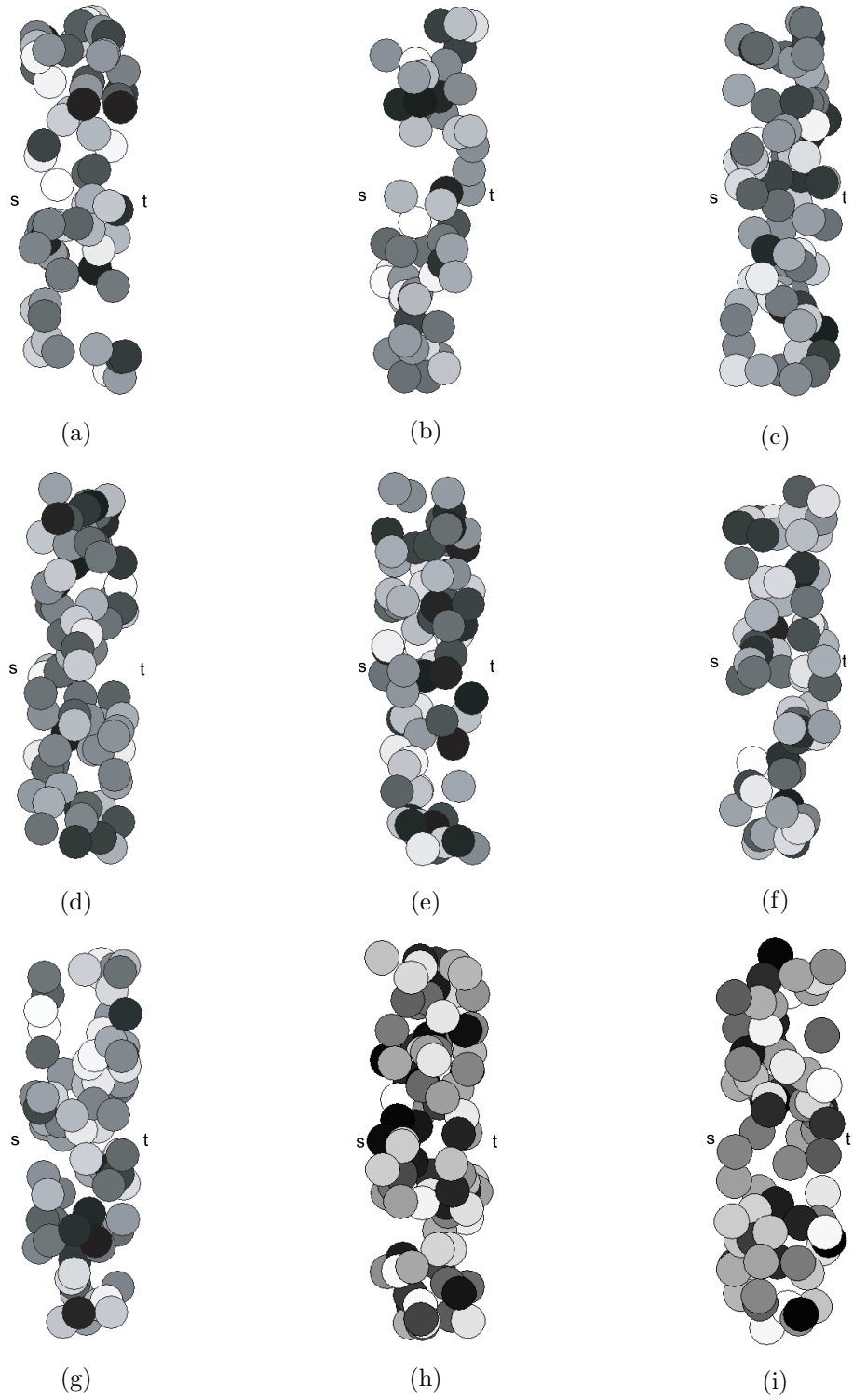


Figure 4.6: Nine realizations of a marked point process.

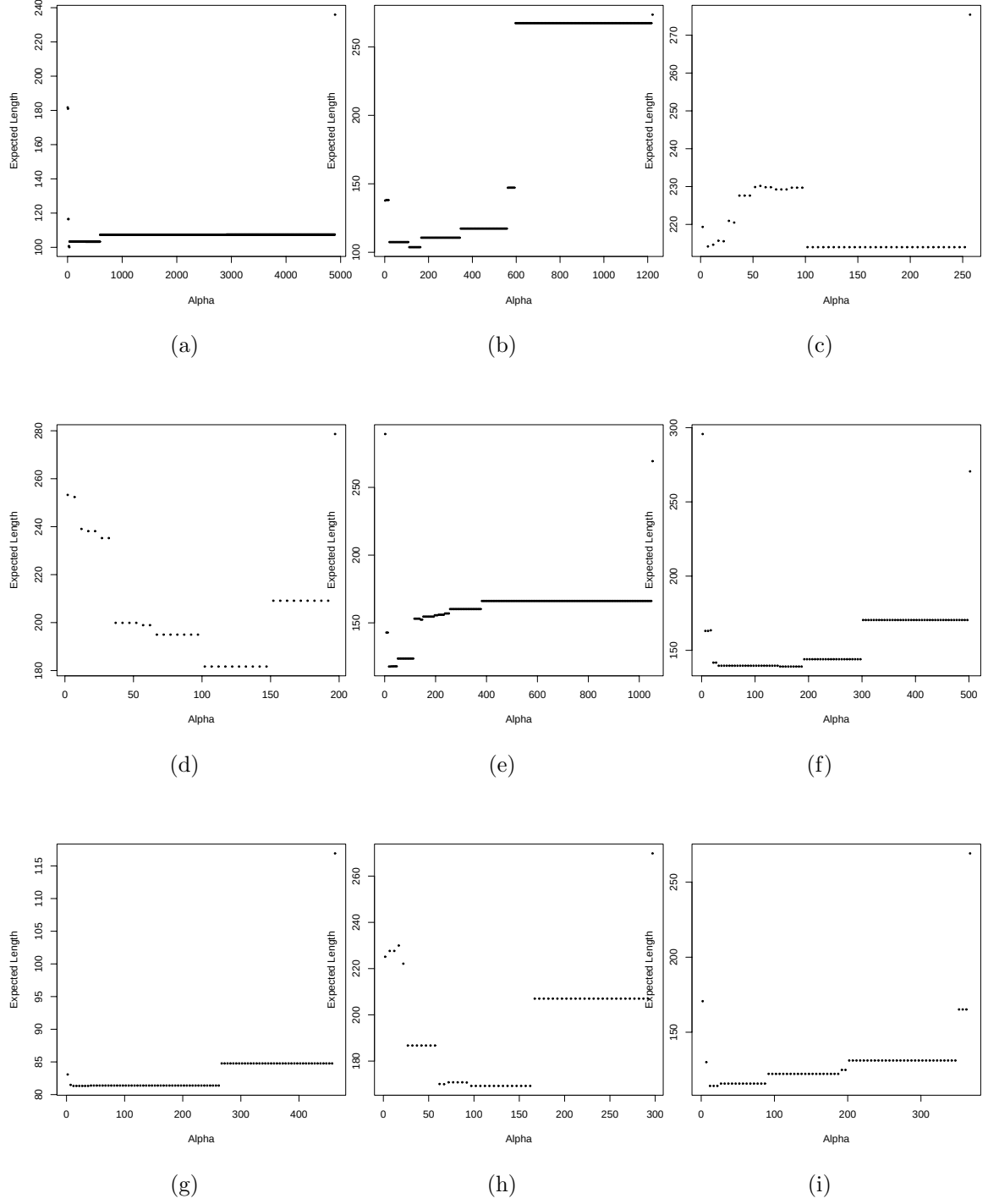


Figure 4.7: Plots of $\mathbb{E} \ell^e p_{\mathcal{D}_\alpha}$ against $\alpha = 2, 7, 12, 17, \dots$ for the respective nine realizations in Figure 4.6. Note that in each of these plots, for the largest value of α which is plotted, we have $p_{\mathcal{D}_\alpha} = p_{\mathcal{D}_\infty}$.

values $\alpha = 2, 7, 12, 17, \dots$, we found that the value of α of minimum $\text{El}^e p_{\mathcal{D}_\alpha}$ are, for the respective nine realizations, 32, 162, 252, 147, 22, 187, 37, 162, and 22. These respective nine values would be the ones to choose for the parameter α in (linear undesirability function based) simulated risk disambiguation protocols if the DM encounters these respective nine realizations.

In general, since $\text{El}^e p_{\mathcal{D}_\alpha}$ can be efficiently evaluated for any single value of α , the usual numerical optimization methods for real functions of the real line are applicable in minimizing $\text{El}^e p_{\mathcal{D}_\alpha}$ over $\alpha \geq 0$.

For large values of K where exact evaluation of $\text{El}^e p_{\mathcal{D}_\alpha}$ may not be practical (even for a single value of α), Monte-Carlo simulations of $p_{\mathcal{D}_\alpha}$ can yield approximate values of $\text{El}^e p_{\mathcal{D}_\alpha}$.

4.5 Adaption to DRDP

We now adapt the SR disambiguation protocol to DRDP [3]. Under the simulation of risk, for any s, t walk W in $G = (V, E)$, the Euclidean length of W , $\ell^e(W)$, is the number of edges in W (since each edge in our lattice has Euclidean length 1). The

risk length of W is defined as

$$\ell^r(W) := -\log \prod_{R_i: R_i \cap W \neq \emptyset} (1 - \rho_i).$$

This negative logarithm of the probability that W is permissibly traversable is a measure of the risk in traversing W . For the monotonically nondecreasing undesirability function $g : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$, $g(\ell^e(W), \ell^r(W))$ is thought of as a measure of the undesirability of W in the sense that, if the DM was required to traverse from s to t in G under the simulation of risk and without a disambiguation capability, the DM would select the walk $\phi_g := \arg \min_{s, d \text{ walks } W} g(\ell^e(W), \ell^r(W))$.

The *adapted SR disambiguation protocol* associated with g has the DM traverse ϕ_g from s until its first ambiguous edge, say, e is encountered at, say, vertex v , and say e intersects region R_i . At this point (since the DM may not take on risk in actuality), disambiguate the obstacle region R_i and repeat this procedure using v as the new s , and then either remove R_i or set $\rho_i := 1$ according as R_i was just discovered to be a false or true obstacle. If at some point the limit K on the number of disambiguations has been reached, then the shortest unambiguously permissible path is then taken to t .

The simplest undesirability functions are, again, the linear ones, where $g(u, v) := u + \alpha \cdot v$ for some given constant $\alpha > 0$, and it is to these undesirability functions

that we restrict our attention in this section. To find ϕ_g in this particular case, we just need to find a deterministic shortest s, t path in G (via Dijkstra's algorithm, say) where each edge in G is weighted³ with

$$w(e) := 1 - \alpha \cdot \frac{1}{2} \sum_{i=1}^{|X|} \# \text{comp}(e \setminus R_i) \cdot \delta_{e \cap R_i \neq \emptyset} \cdot \log(1 - \rho_i), \quad (4.1)$$

where δ is the indicator function (taking value 1 or 0 according as its subscripted expression is true or false) and $\# \text{comp}(\cdot)$ is the number of connected components of its argument. See illustration in Figure 4.8 with $|X| = 4$. Corresponding edge weights are shown in Table 4.3.

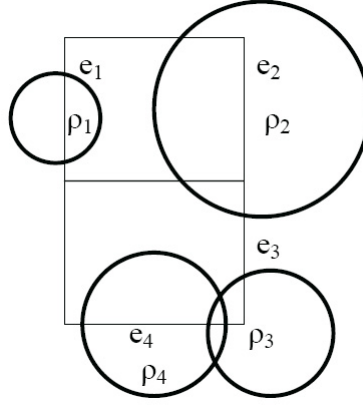


Figure 4.8: Edge weights illustration with $|X| = 4$.

³We assume that s and t are not inside any disc, and that ϕ_g never revisits a possible-obstacle.

Edge	Edge weight
e_1	$1 - \alpha \log(1 - \rho_1)$
e_2	1
e_3	$1 - \frac{\alpha}{2} (\log(1 - \rho_2) + \log(1 - \rho_3))$
e_4	$1 - \frac{\alpha}{2} (\log(1 - \rho_3) + 2\log(1 - \rho_4))$

Table 4.3: Weights of the edges in Figure 4.8.

4.6 Computational Experiments

Because of its continuous setting, optimal protocols are not readily computable for all but the most trivial instances of RDP. On the other hand, we can use the *BAO** Algorithm to compute an optimal protocol for relatively small instances of the problem in the discrete setting for comparison purposes.

This section compares the performance of the adapted SR protocol to the optimal policy on DRDP instances where the optimal policy can be obtained by the *BAO** Algorithm. The simulations are performed in the environment illustrated in Figure 2.5. Specifically, i_{max} and j_{max} are taken as 20 and 15, respectively, with $s = (10, 15)$ and $t = (10, 1)$. Each disc has a radius of 1.5 and the disc centers are sampled from a uniform distribution on the pairs of real numbers in $[3, 18] \times [3, 13]$; in particular, this ensures that there is always a permissible path from s to t . The number of discs is taken as 20 and the cost of disambiguation is taken as 1.5. Disc marks are sampled from a uniform distribution on the unit interval. The values of α minimizing $El^e p_\alpha$ are computed numerically by evaluating $El^e p_\alpha$ for a mesh of α values—starting at

$\alpha = 2$ and incrementing successively by 5 units until α is large enough that no disambiguations are performed. We will denote $El\hat{p}_{SR}^*$ the value of the SR protocol with the best α found by meshing.

Table 4.4 compares $El\hat{p}_{SR}^*$ to Elp^* (the expected Euclidean length of the overall optimal policy) for 50 experiment realizations for $K = 1$ and $K = 2$, respectively, for which the the optimal policy was to perform at least one disambiguation. In the table, the second column shows the percentage of simulations where $El\hat{p}_{SR}^* = Elp^*$, in which case the SR protocol contains the overall optimal policy, and the next column displays the median of the percentage errors, $\frac{El\hat{p}_{SR}^* - Elp^*}{Elp^*} * 100$. Mean BAO^* execution time was 9.6 minutes, whereas mean SR protocol execution time was just .32 seconds. As Table 4.4 illustrates, solutions found by the SR protocol compare favorably to the optimal solutions; the SR protocol found the optimal solution in 49% of the simulations, whereas the median percentage error was only 0.54%.

	% where $El\hat{p}_{SR}^* = Elp^*$	Median % errors $\frac{El\hat{p}_{SR}^* - Elp^*}{Elp^*} * 100$
K = 1	52	0.0
K = 2	46	1.08

Table 4.4: Comparison of $El\hat{p}_{SR}^*$ to Elp^* .

Chapter 5

The CR Disambiguation Protocol

5.1 Introduction

In this chapter, we propose another suboptimal, but effective and efficiently computable algorithm for RDP and DRDP, which we call the CR disambiguation protocol [4]. This protocol is based on a different setting called the continually reactivated (CR) setting, which is an alteration to RDP where the true/false status of any detection is not static but, rather, changes over time. There is a clear polynomial-time optimal algorithm for the CR setting through the use of TAG. We adapt this algorithm to RDP and DRDP, calling it the CR disambiguation protocol. The CR protocol is proved to be optimal for parallel graphs in discrete SDL, and this result is extended to yield optimal policies for a broader class of SDL problems where the DM's choice is just between parallel avenues under fixed policies within the avenues.

Simulations then illustrate the effectiveness of the CR protocol; CR is much more efficiently computable even than SR protocol, and it also yields results that are comparable to that of SR, making it particularly suitable for real-time applications. The CR protocol is also employed for DRDP instances that are small enough to be solved optimally by the BAO* Algorithm; the quality of the policies obtained by the CR protocol compare favorably to the optimal policies.

Finally, the situation where there is an obstacle neutralization capability instead of the disambiguation capability is seen as a special case of the continually reactivated setting that leads to the CR protocol.

5.2 The CR Setting and the CR Protocol

The setting for CR differs from RDP in only one way: the true/false status of any detection is not static but, rather, changes over time (in contrast to RDP, where each detection's true/false status is static, and once a detection is disambiguated it will never again be ambiguous). We shall call this the *continually reactivated* (CR) setting.

In this setting, at any moment in time and for all $x \in X$, the probability that x is true is $\rho(x)$, the probability that x is false is $1 - \rho(x)$, but the status of x at different times are statistically independent of each other. Thus, even if at one moment of

time an obstacle $x \in X$ is revealed by a disambiguation to be true (in which case R_x is not immediately traversable), a later disambiguation may reveal x to have subsequently changed its status to false, and a yet later disambiguation might reveal yet another change in the status of x . If a disambiguation of detection x reveals at some moment that x is false, the curve may immediately enter R_x , in which case x will stay false until the curve next exits R_x , thus rendering R_x traversable on this occasion.

Let us consider what features an optimal CR policy will have. Under an optimal CR policy, suppose the disambiguation of a detection x is performed at some point and time, and consider the situation if the result of the disambiguation is that x is (currently) true. To then execute another disambiguation would be every bit as valuable as it was to originally execute the disambiguation, since the reactivation of the obstacle has brought us back to that previous situation from the perspective of what to do next. Thus, without loss of generality, the optimal policy would have us execute and re-execute disambiguations until such time as x is false (which will happen with probability 1 if $\rho(x) < 1$). The number of such disambiguations has a geometric distribution with probability of success $1 - \rho(x)$, and thus the expected number of disambiguations needed is $\frac{1}{1-\rho(x)}$, and the associated expected cost (added to the s, t traversal's length) is $\frac{c(x)}{1-\rho(x)}$. (If $\rho(x) = 1$, we adopt the convention that $\frac{c(x)}{1-\rho(x)} := \infty$, even if $c(x) = 0$.)

Thus, an optimal policy can be found by first finding a deterministic shortest s, t curve γ , shortest in the sense of arclength supplemented by, for each $x \in X$, an added $\frac{1}{2} \frac{c(x)}{1-\rho(x)}$ for each time the boundary ∂R_x is crossed by the curve. Then it would be an optimal policy to traverse γ from s to t , along the way repeatedly disambiguating any $x \in X$ when γ needs to enter R_x until such time as x is false. As discussed in Section 4.2.1, γ can be computed in $O(|X|^3 \log |X|)$ operations using TAG. We will call the above optimal policy the *CR protocol*, and it will be adapted in Section 5.3 for use in RDP.

5.3 The CR Protocol for RDP

Even in RDP, where the true/false status of each possible-obstacle is static, it is still possible to implement the CR protocol. Specifically, given the RDP parameters s, t, X, ρ , and K , the CR protocol would have the DM compute and traverse the shortest s, t path γ in TAG relative to the weights specified above until such point as γ penetrates R_x for some $x \in X$ —let us denote by y the point where γ first intersects ∂R_x —and the protocol would then have us disambiguate x at the point y . Unlike in the CR setting, the result of this disambiguation will not change over time, so the DM updates the parameters of this particular realization accordingly; if x is revealed to be a true obstacle then update ρ by setting $\rho(x) := 1$, and if x is revealed to be a false obstacle then update X by removing x from X . Now the CR protocol can

once again be queried from y relative to the updated parameters, with y in place of s . (Note that if x is indeed revealed to be a true obstacle then $\frac{c(x)}{1-\rho(x)}$ would henceforth be infinite, thus the γ computed relative to the updated parameters will not intersect R_x , and x will not again be disambiguated.)

The CR protocol can be used in this manner in RDP to determine, given any current location and current values of X and ρ , which possible-obstacle to disambiguate next and where to carry out that disambiguation. Upon learning the results of this disambiguation, the DM can then find the next possible-obstacle to disambiguate afterwards by re-querying the CR protocol on the updated parameters, and so on iteratively until t is reached. The computational cost of computing the CR protocol is $O(|X|^3 \log |X|)$ operations for each of the $O(|X|)$ protocol queries (one query for each disambiguation) for a total of $O(|X|^4 \log |X|)$ operations.

5.3.1 Adaptation to DRDP

Given a DRDP problem defined on $G = (V, E)$, the discrete CR setting can be analogously defined, wherein each stochastic edge $e \in E' \subseteq E$ is continually reactivated; the probability at any moment that e is not traversable is $\rho(e)$, independent of other moments in time. (If a disambiguation reveals that e is currently traversable, then e may be traversed immediately.) The CR protocol analogue would be an optimal policy in

the discrete CR setting by identical reasoning as in Section 5; a shortest s, t path γ should be found in G relative to the weights $\ell(e) + \frac{c(e)}{1-\rho(e)}$ for each $e \in E$ (if $e \in E \setminus E'$ then $\frac{c(e)}{1-\rho(e)}$ is understood to be 0), and γ should be traversed from s to t , repeatedly disambiguating en route the edges that need to be traversed until they are traversable.

Indeed, the CR protocol is also applicable to DRDP. Specifically, it would call for γ to be computed as in the previous paragraph, and then γ is traversed until the first ambiguous edge $e \in E'$ is reached. Say the endpoints of e are a and b , with a being the first endpoint encountered along γ . The edge e is disambiguated from a ; if e is not traversable then e is removed from E and the process is repeated with a in place of s , but if e is traversable then e is placed in E' and traversed to b , then the process is repeated with b in place of s . When t is reached this iterative process is complete.

5.3.2 Optimality of CR Protocol for Discrete SDL on Parallel Graphs

We will call a graph $G = (V, E)$ *parallel* if $V = \{s, t\}$, and all edges in E have both s and t as endpoints. The main result in this section is the following theorem.

Theorem 2. *The CR protocol is an optimal policy for discrete SDL problems on parallel graphs.*

Without loss of generality the policies we need to consider in this case have a very particular form; a policy consists of any ordering on E (we may assume $E' = E$) wherein the edges are disambiguated in this order until a traversable edge is found, at which point that edge is traversed. We shall assume that if an edge is disambiguated and found to be traversable, then it will be traversed immediately. (Let us say that the ordering *generates* this policy.) Unfortunately, there are $|E|!$ such orderings, hence $|E|!$ such policies—too many to compute each of their expected lengths for the purposes of finding the optimal policy.

By Theorem 2, we only need to order the edges by $h(e) := \ell(e) + \frac{c(e)}{1-\rho(e)}$ for all $e \in E$ (this ordering can be done in $|E| \log |E|$ operations), and the policy generated by this order is in fact the CR protocol, and is optimal by Theorem 2. To prove Theorem 2, we first state and prove the following lemma.

Lemma 3. *For the discrete SDL on a parallel graph, consider any particular ordering of the edges $e_1, e_2, e_3, \dots, e_{|E|}$ with respective probabilities $\rho_1, \rho_2, \rho_3, \dots, \rho_{|E|}$, lengths $\ell_1, \ell_2, \ell_3, \dots, \ell_{|E|}$, and disambiguation costs $c_1, c_2, c_3, \dots, c_{|E|}$. Let η denote the expected length of the generated traversal and, for any $k \in \{1, 2, \dots, |E| - 1\}$, let η_k denote the expected length of the traversal generated by this same ordering except with the transposition of the k th and $k + 1$ th edges. Then $\eta_k - \eta = (\prod_{j < k} \rho_j)(h_{k+1} - h_k)(1 -$*

$$\rho_k)(1 - \rho_{k+1}).$$

Proof of Lemma 3: If $\rho_j > 0$ for all j then the expected length of a traversal is infinite. Otherwise, independence of traversability for different edges and the definition of the relevant policy yield $\eta = \sum_{i=1}^{|\mathbb{E}|} (\prod_{j<i} \rho_j)(c_i + (1 - \rho_i)\ell_i) = \sum_{i=1}^{|\mathbb{E}|} (\prod_{j<i} \rho_j)(1 - \rho_i)h_i$. Note that this expression for η and the analogous expression for η_k differ only for summands $i = k, k+1$, so we subtract $\eta_k - \eta = (\prod_{j<k} \rho_j)(1 - \rho_{k+1})h_{k+1} + (\prod_{j<k} \rho_j)\rho_{k+1}(1 - \rho_k)h_k - (\prod_{j<k} \rho_j)(1 - \rho_k)h_k - (\prod_{j<k} \rho_j)\rho_k(1 - \rho_{k+1})h_{k+1}$, which simplifies to $\eta_k - \eta = (\prod_{j<k} \rho_j)(h_{k+1} - h_k)(1 - \rho_k)(1 - \rho_{k+1})$, and Lemma 3 is shown. $\square$

Proof of Theorem 2: Suppose the edges are ordered $e_1, e_2, e_3, \dots, e_{|\mathbb{E}|}$ so as to generate an optimal policy. Define $\hat{k} = \min\{k : \rho_k = 0\}$, which exists if η is finite. By optimality, we have for all $k = 1, 2, \dots, |\mathbb{E}| - 1$ that $\eta_k - \eta = (\prod_{j<k} \rho_j)(h_{k+1} - h_k)(1 - \rho_k)(1 - \rho_{k+1}) \geq 0$, which implies that $h_{k+1} \geq h_k$ for all $k \leq \hat{k}$, and also implies $\eta_k = \eta$ for all $k > \hat{k}$. Thus, repeated transpositions among the edges $e_{\hat{k}+1}, e_{\hat{k}+2}, \dots, e_{|\mathbb{E}|}$ will not affect the expected length of the traversal, hence the CR protocol (in which edges are ordered by h_k) is optimal. $\square$

5.3.3 Adaptation to SDL with Parallel Avenues

Consider a special case of SDL where the obstacle scene or the underlying graph consists of non-overlapping parallel avenues $p_1, p_2, \dots, p_n$ between s and t : sets of possible-obstacles or stochastic edges that can be disambiguated along each avenue are disjoint. Suppose that for each of these avenues, there exists a policy that specifies the actions of the DM under any circumstance and at any possible location within the avenue, and that if an avenue is found to be untraversable, it will never be taken again; the DM will return to s and try another avenue; the DM will repeat this process until the destination is reached. We shall call this problem *SDL with parallel avenues*, and denote it by $\{p_1, p_2, \dots, p_n\}$. The following theorem shows how the CR protocol can be adapted to efficiently find an optimal protocol for this problem.

Theorem 4. *In SDL with parallel avenues $p_1, p_2, \dots, p_n$, let a_i denote the expected traversal length of p_i conditioned on reaching t , b_i denote the expected traversal length of p_i conditioned on traversing back to s , and ρ_i denote the probability that p_i is untraversable. The optimal policy for $\{p_1, p_2, \dots, p_n\}$ is generated via ordering the avenues by $h'_i := a_i - b_i + \frac{b_i}{1-\rho_i}$. That is, the optimal policy is to traverse the avenues in increasing h'_i , where if the current avenue is found to be untraversable at some point, the DM traverses back to s and starts traversing the next avenue—until arrival at the destination.*

Proof: Clearly, if $\rho_i > 0$ for all i , then the expected length of a traversal is infinite. Otherwise, the expected length of the traversal generated by a particular ordering of avenues $p_1, p_2, \dots, p_n$ is given by $\eta' = \sum_{i=1}^n (\prod_{j < i} \rho_j) (\rho_i b_i + (1 - \rho_i) a_i) = \sum_{i=1}^n (\prod_{j < i} \rho_j) (1 - \rho_i) h'_i$. Let η'_k denote the expected length of the traversal generated by this same ordering except with the transposition of the k th and $k + 1$ th avenues for any $k \in \{1, 2, \dots, n - 1\}$. As in Lemma 3, $\eta'_k - \eta = (\prod_{j < k} \rho_j) (h'_{k+1} - h'_k) (1 - \rho_k) (1 - \rho_{k+1})$. Suppose the avenues are ordered $p_1, p_2, p_3, \dots, p_n$ so as to generate an optimal policy. By optimality, $\eta'_k - \eta \geq 0$ for all $k = 1, 2, \dots, n - 1$, which implies that, for $\hat{k} = \min\{k : \rho_k = 0\}$, $h'_{k+1} \geq h'_k$ for all $k \leq \hat{k}$, and $\eta'_k = \eta'$ for all $k > \hat{k}$. Therefore, repeated transpositions among the avenues $p_{\hat{k}+1}, p_{\hat{k}+2}, \dots, p_n$ will not affect the expected length of the traversal, hence the protocol in which avenues are ordered by h'_k is optimal. $\square$

As an application of Theorem 4, we shall now compute the optimal protocol for the SDL with parallel avenues problem instance illustrated in Figure 5.1. The underlying graph in this instance has three parallel avenues: Avenue p_1 that consists of edges e_1 and e_2 , avenue p_2 that consists of the edge e_3 , and avenue p_3 that consists of edges e_4, e_5, e_6 , and e_7 . Cost of disambiguation for each edge is taken as 1 and mark of each edge is taken as 0.3. Edge lengths are shown next to the edges in the figure.

We shall assume that a depth-first traversal strategy is adopted within each avenue,

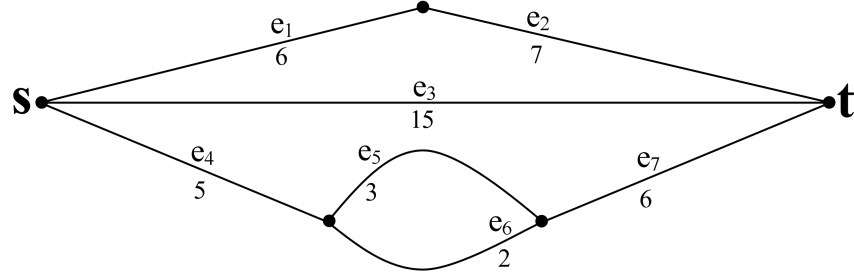


Figure 5.1: Illustration of an SDL with parallel avenues problem instance with three parallel avenues.

which implies that the two parallel edges e_5 and e_6 is an instance of the discrete SDL problem on parallel graphs. Application of the CR protocol yields that, if e_4 is disambiguated and found to be traversable, first e_6 will be disambiguated, as it has a lower h_i value. For each avenue, a_i, b_i, ρ_i , and h'_i are calculated as follows:

- $\rho_1 = 1 - (.7)(.7) = .51$

$$a_1 = 1 + 6 + 1 + 7 = 15$$

$$b_1 = \frac{(.3)}{(.51)}(1) + \frac{(.3)(.7)}{(.51)}(1 + 2(6) + 1) = 6.353$$

$$h'_1 = 15 - 6.353 + \frac{6.353}{.49} = 21.61$$

- $\rho_2 = .3$

$$a_2 = 1 + 15 = 16$$

$$b_2 = 1$$

$$h'_2 = 16 - 1 + \frac{1}{.7} = 16.43$$

- $\rho_3 = 1 - (.7)[1 - (.3)(.3)](.7) = .5541$

$$a_3 = 1 + 5 + 1 + (.3)1 + \frac{(.3)}{(.51)}(2) + \frac{(.3)(.7)}{(.51)}(3) + 1 + 6 = 16.71$$

$$\begin{aligned}
b_3 &= \frac{(.3)}{(.5541)}(1) + \frac{(.7)(.3)(.3)}{(.5541)}(1 + 2(5) + 1 + (.3)1) \\
&\quad + \frac{(.7)[1-(.3)(.3)](.3)}{(.5541)}(1 + 2(5) + 1.3 + 2\left(\frac{(.3)}{(.51)}(2) + \frac{(.3)(.7)}{(.51)}(3)\right) + 1) = 8.72 \\
h'_3 &= 16.71 - 8.72 + \frac{8.72}{.4459} = 27.55
\end{aligned}$$

Thus, the optimal policy is to first try avenue p_2 , which is edge e_3 (since it has the lowest h' value), then avenue p_1 , and finally avenue p_3 .

5.4 Computational Experiments

In this section, we empirically compare the CR protocol against the optimal policy and the SR disambiguation protocol. Specifically, we present computational experiments that consist of three sets of simulations:

- I. Comparison of the CR protocol against the optimal policy on DRDP instances where the optimal policy is obtained by the BAO^* Algorithm,
- II. Comparison of the CR and SR protocols as applied to the mine countermeasures example presented in Section 4.3, and,
- III. Comparison of the CR and SR protocols on randomly generated RDP instances in three different simulation environments.

5.4.1 Comparison of the CR Protocol Against the Optimal Policy

We first compare the performance of the CR protocol against the optimal policy on relatively small DRDP instances illustrated in Figure 2.5, for which the optimal policy can be obtained by the *BAO** Algorithm. Specifically, i_{max} and j_{max} are taken as 20 and 15, respectively, with $s = (10, 15)$ and $t = (10, 1)$. Each disc has radius 1.5, and the disc centers are sampled from a uniform distribution on the pairs of real numbers in $[3, 18] \times [3, 13]$; in particular, this ensures that there is always a permissible path from s to t . Number of discs is taken as 20 and the cost of disambiguation is taken as 1.5. Disc marks are sampled from a uniform distribution on the unit interval.

Let Elp_{CR} denote the value of the CR disambiguation protocol, and Elp^* the expected length of the overall optimal policy. Table 5.1 compares Elp_{CR} to Elp^* for $K = 1$ and $K = 2$, respectively. For each K , 50 instances were generated randomly for which the the optimal policy was to perform at least one disambiguation. In the table, the second column shows the percentage of simulations where $Elp_{CR} = Elp^*$, and the third column displays the median of the percentage errors $\frac{Elp_{CR} - Elp^*}{Elp^*} * 100$. Mean *BAO** execution time was 9.6 minutes whereas mean CR protocol execution time was .05 seconds. As Table 5.1 illustrates, solutions found by the CR protocol compare favorably to the optimal solutions. The CR protocol found the optimal solution in 37% of the simulations, whereas the median percentage error was only 1.25%.

	% where $Elp_{CR} = Elp^*$	Median % errors $\frac{Elp_{CR} - Elp^*}{Elp^*} * 100$
$K = 1$	42	0.98
$K = 2$	32	1.51

Table 5.1: Comparison of Elp_{CR} to Elp^* .

5.4.2 Comparison of the CR and SR Protocols on a Mine Countermeasures Example

In Section 4.3, we discuss how the SR protocol can be efficiently employed on the mine countermeasures example shown in Figure 4.1, and we report the SR values for different intervals of the disambiguation cost c in Table 4.2. We employed the CR protocol on the same instance starting from $c = 0$ and incrementing .1 units until $c = 228.2$; for c values greater than 228.2, both protocols dictated traversing the conventional zero-risk path. Table 5.2 compares the CR and SR protocols for different intervals of c : The second and third columns show the values of the CR and SR protocols for the 11 cost intervals shown in the first column. Let $El\hat{p}_{SR}^*$ denote the value of the SR protocol. The next column shows the difference between the values of the two protocols, $Elp_{CR} - Elp_{SR}^*$, and the last column shows the maximum percentage errors $\frac{Elp_{CR} - Elp_{SR}^*}{Elp_{SR}^*} * 100$ for each interval. As seen in the table, values of both protocols are identical in the following 5 intervals: (1.2, 4.1), (49.6, 50.4), (69, 76.0), and, (116.5, ∞). Maximum percentage error is in the (0.0, 1.1) interval at 2.6%, but, for the other 5 intervals where SR values are better, maximum percentage

error was less than 1%; illustrating the effectiveness of the CR protocol on this mine countermeasures example.

Cost interval I	$\mathbb{E}\ell p_{CR}$ for $c \in I$	$\mathbb{E}\ell p_{SR}^*$ for $c \in I$	$\mathbb{E}\ell p_{CR} - \mathbb{E}\ell p_{SR}^*$ for $c \in I$	Max. difference % $\frac{\mathbb{E}\ell p_{CR} - \mathbb{E}\ell p_{SR}^*}{\mathbb{E}\ell p_{SR}^*} * 100$ in I
(0.0, 1.1)	734.90 + 3.1033c	717.22 + 2.1665c	17.68 + 0.9368c	2.600
(1.2, 4.1)	717.92 + 2.1665c	717.92 + 2.1665c	0	0
(4.2, 15.9)	721.60 + 1.2698c	720.89 + 1.2698c	0.71	0.096
(16.0, 19.6)	721.60 + 1.2698c	722.92 + 1.1423c	-1.32 + 0.1275c	0.158
(19.7, 49.5)	720.89 + 1.2698c	722.92 + 1.1423c	-2.03 + 0.1275c	0.549
(49.6, 50.4)	722.92 + 1.1423c	722.92 + 1.1423c	0	0
(50.5, 68.9)	722.92 + 1.1423c	723.07 + 1.1393c	-0.15 + 0.0030c	0.007
(69, 76.0)	723.07 + 1.1393c	723.07 + 1.1393c	0	0
(76.1, 116.4)	723.07 + 1.1393c	725.81 + 1.1033c	-2.74 + 0.0360c	0.002
(116.5, 228.1)	725.81 + 1.1033c	725.81 + 1.1033c	0	0
(228.2, ∞)	977.54	977.54	0	0

Table 5.2: Comparison of the CR and SR protocols on a mine countermeasures example for different cost intervals.

5.4.3 Comparison Against the SR Protocol on Random RDP Instances

We now compare the CR and SR protocols on randomly generated RDP instances in the following three different simulation environments:

Environment A: Each disc has a radius of 4.5, with $s = (50, 20)$, $t = (50, 1)$. The disc centers are sampled from a uniform distribution on the pairs of real numbers in $[6, 95] \times [6, 15]$.

Environment B: Each disc has a radius of 6.5, with $s = (50, 75)$, $t = (50, 1)$. The disc centers are sampled from a uniform distribution on the pairs of real numbers in $[8, 93] \times [8, 68]$.

Environment C: Each disc has a radius of 6.5, with $s = (50, 40)$, $t = (50, 1)$.

For the disc centers, first an angle ω is sampled from a uniform distribution on $[1.15\pi, 1.85\pi]$. Then, x - and y -coordinates of disc centers are taken as $50 + 50 \cos(\omega) + (3.25)\text{Normal}(0, 1)$ and $70 + 50 \sin(\omega) + (3.25)\text{Normal}(0, 1)$, respectively.

In each of these environments, number of discs is taken as 50, disc marks are sampled from a uniform distribution on the unit interval, and the cost of disambiguation is taken as 5. The environments are illustrated in Figures 5.2, 5.3, and 5.4, respectively.

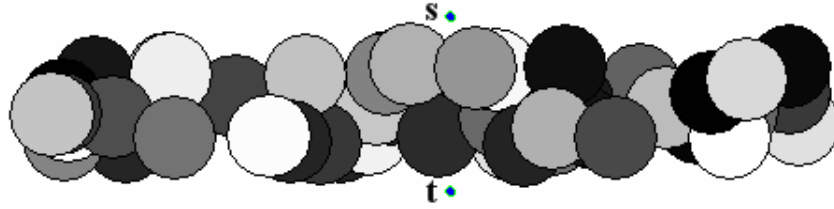


Figure 5.2: An illustration of simulation environment A.

In the SR disambiguation protocol, denote by p_α the s, t path traversed for a fixed $\alpha \geq 0$. Similar to what is done in Section 4.4, values of α minimizing $\text{El}p_\alpha$ are computed numerically by evaluating $\text{El}p_\alpha$ for a mesh of α values—starting at $\alpha = 2$ and incrementing successively by 5 units until $\alpha > 2000$ or α is large enough that no disambiguations are performed.

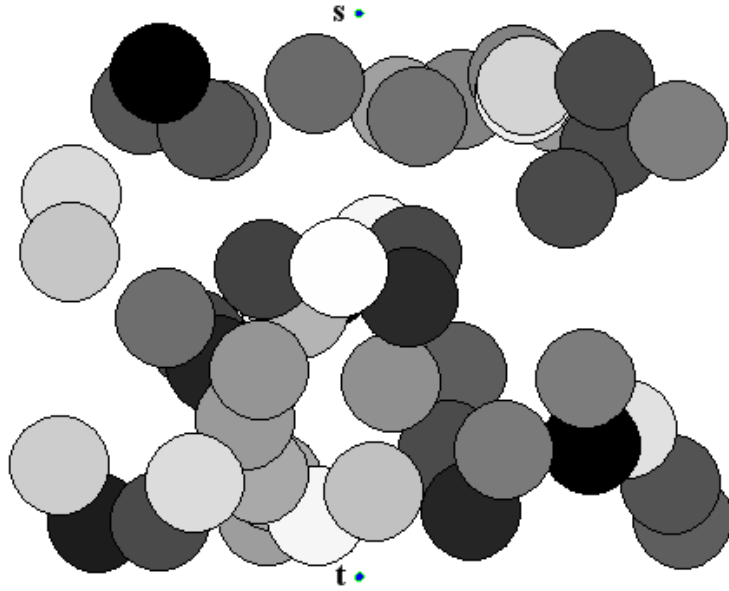


Figure 5.3: An illustration of simulation environment B.

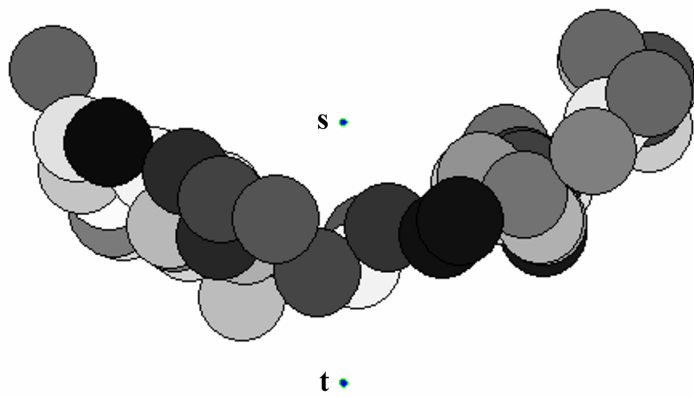


Figure 5.4: An illustration of simulation environment C.

Table 5.3 compares Elp_{CR} to $El\hat{p}_{SR}^*$ for 100 experiment realizations in each of the three simulation environments for $K = 3$. The second column shows the percentage of simulations where $Elp_{CR} = El\hat{p}_{SR}^*$ and the third column shows the median of the percentage errors, $\frac{Elp_{CR} - El\hat{p}_{SR}^*}{El\hat{p}_{SR}^*} * 100$.

	% where $Elp_{CR} = El\hat{p}_{SR}^*$	Median % errors $\frac{Elp_{CR} - El\hat{p}_{SR}^*}{El\hat{p}_{SR}^*} * 100$
Environment A	22	2.32
Environment B	22	1.71
Environment C	29	1.56

Table 5.3: Comparison of Elp_{CR} to $El\hat{p}_{SR}^*$.

As Table 5.3 shows, the CR protocol yields solutions that are quite comparable to those obtained via the SR protocol. Among all the simulations we performed, the values of the CR and SR protocols were the same 24.33% of the time. CR protocol solutions were better than the SR protocol solutions only in 5 of the simulations, but the overall median error percentage was just 1.86%. In addition, the value of the CR protocol was 34.68% less than the length of the conventional zero-risk path on the average.

Across the simulations, mean runtime ratio of the CR and SR protocols—which, in effect, is driven by the mesh size—was 26.56. In particular, maximum CR protocol execution time was 1 minute 17 seconds in our simulations, which were run on

a 3.5 gigahertz personal computer, whereas SR protocol execution time sometimes exceeded an hour. Other numerical optimization methods can yield shorter execution times for finding $\alpha^* \in \arg \min_{\alpha \geq 0} \text{El}p_\alpha$. However, this optimization problem is rather challenging, and is the focus of our current research. The CR protocol, on the other hand, is parameter-free, and yields good solutions in very short execution times—making it an excellent choice for real-time applications of RDP.

5.5 Neutralization as a Special Case of CR Setting

In this section we point out that a special case of the CR setting (for which the CR protocol is an optimal policy) is the different setting where a neutralization capability is available in place of the disambiguation capability that this dissertation is principally concerned with.

Specifically, suppose that for all $x \in X$, when the s, t traversal is on the boundary ∂R_x , we have the option to *neutralize* obstacle x thus removing x from X , but at a cost $c'(x)$ added to the length of the s, t traversal for a given cost function $c' : X \rightarrow \mathbb{R}_{\geq 0}$. We can model this scenario as the CR setting where the probabilities ρ are all zero; just like the neutralization of x will definitely remove the obstacle R_x which couldn't be previously traversed, so too in the disambiguation setting we have that R_x can't be traversed prior to disambiguation—even as $\rho(x) = 0$ —but will defi-

nately be traversable after the disambiguation expenditure which is thus, effectively, the neutralization expenditure.

In particular, the problem of how to best use the neutralization capability to achieve the shortest s, t traversal is solved in $O(|X|^3 \log |X|)$ operations by applying the CR protocol to the CR Setting such that, for all $x \in X$, $c(x)$ is set to $c'(x)$ and $\rho(x)$ is set to 0.

Chapter 6

Summary, Conclusions, and Directions for Future Research

6.1 Summary and Conclusions

In this research, we studied stochastic shortest path problems in the presence of a dynamic learning capability (SDL). Specifically, a spatial arrangement of regions that are possibly obstacles needs to be speedily and safely traversed, and there is a dynamic capability of disambiguating each possible-obstacle en route at a cost; the traversal may enter the associated region only if it is revealed to be a false obstacle. The problem of identifying an optimal policy for where and when to execute disambiguations so as to minimize the expected length of the traversal has been proven intractable in many broad settings and no efficiently computable optimal policy is known.

We first defined the research problem in continuous and discrete settings and provided a comprehensive literature review. We specifically focused on a continuous variant of the problem with disc-shaped possible-obstacles, called the Random Disambiguation Paths Problem (RDP), and its discrete approximation, called Discrete RDP (DRDP).

We next cast SDL as a Markov Decision Process. The state space, however, is too large for an efficient solution via stochastic dynamic programming (SDP). We proposed an exact algorithm for DRDP, called the *BAO** Algorithm, which is not polynomial-time, but significantly shortens the execution time required for an optimal solution to relatively small instances. The *BAO** Algorithm is an improvement on the *AO** Algorithm that employs stronger pruning methods, including utilization of upper and lower bounds on path lengths. We presented computational experiments which indicate that *BAO** has significant computational savings over SDP and *AO**. We also formulated RDP as a Partially Observable Markov Decision Process.

We then presented a new data structure, the tangent arc graph (TAG), which is comprised of the topological superimposition of all of the visibility graphs generated by the collection of all subsets of obstacles. Although there are exponentially many such subsets, we showed that TAG is polynomially-sized. The TAG played an important role in our algorithms for RDP. We also pointed out that the well-known *A** Algo-

rithm with a slightly stronger admissibility requirement on the heuristic function is equivalent to Dijkstra’s Algorithm under a change of variable.

We proposed the simulated risk disambiguation protocol (SR) for RDP and DRDP, which is suboptimal, but efficiently computable as well as reasonably effective. We provided computational experiments where we compared SR to the optimal policy for small DRDP instances, illustrating the efficiency and effectiveness of the SR protocol.

Next, we proposed the CR disambiguation protocol for RDP and DRDP, which is also suboptimal, but effective and much more efficiently computable even than the SR protocol. We performed extensive simulations which suggest that the CR protocol yields results that compare favorably to those obtained by the optimal policy (on randomly generated DRDP instances) and also the SR protocol (on randomly generated RDP instances).

In what follows, we briefly describe several directions for future research.

6.2 Computing Optimal α in the Simulated Risk Disambiguation Protocol

As illustrated in our numerical experiments, the simulated risk disambiguation protocol is an efficient algorithm for RDP and DRDP. However, $\hat{\alpha}^*$ is currently computed via meshing, which is computationally expensive. Future research can identify more efficient numerical methods to compute α^* , either optimally, or sub-optimally.

6.3 Approximate Solutions

Chapter 2 presents optimal algorithms for discrete SDL (DRDP, in particular), which can only be used to solve relatively small instances of the problem. Future research can investigate approximation schemes for the stochastic dynamic programming [7, 13, 25, 26] or the Markov decision process frameworks [19, 41, 71], as well as algorithms that exploit problem structure to reduce the size of the state space [16, 32, 75].

6.4 Multiple, Noisy Sensors and Neutralization

A more realistic variant of SDL is the one that considers deployment of multiple sensors with different accuracy rates & ranges at different costs, with respective utilization limits (see, e.g., [14, 37]). Additionally, a limited neutralization capability may be assumed. This variant can be formulated in a Partially observable Markov decision process (POMDP) framework. Due to the potentially large state-space, however, approximation schemes may need to be considered [17, 36, 44, 43, 46, 58, 68].

6.5 Roving Detections

Another interesting variant of SDL is the dynamic version, wherein potential-obstacles rove in the obstacle field, i.e., location distribution of potential-obstacles is a function of time. Algorithms developed for stochastic time-dependent networks and networks with dependent arclengths may potentially be adapted for this variant [21, 24, 30, 31, 33, 49, 73].

6.6 Asymptotic Analysis via Percolation Theory

Asymptotic performance analysis of our suboptimal algorithms can be performed via percolation theory [5], similar to what was done in [38].

Bibliography

- [1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.
- [2] V. Aksakalli. The BAO^* algorithm for stochastic shortest path problems with dynamic learning. *Submitted for publication*, 2007.
- [3] V. Aksakalli, D.E. Fishkind, and C.E. Priebe. Adaptation of the simulated risk disambiguation protocol to a discrete setting. *Proceedings of Workshop on POMDP, Classification and Regression: Relationships and Joint Utilization, Cumbria, England*, 2006.
- [4] V. Aksakalli, D.E. Fishkind, and C.E. Priebe. The CR disambiguation protocol. *Submitted for publication*, 2007.
- [5] K.S. Alexander. The RSW theorem for continuum percolation and the CLT for euclidean minimal spanning trees. *Annals of Applied Probability*, 6:466–494, 1996.

- [6] G. Andreatta and L. Romeo. Stochastic shortest paths with recourse. *Networks*, 18(3):193–204, 1988.
- [7] M. Baglietto, G. Battistelli, F. Vitali, and R. Zoppoli. Shortest path problems on stochastic graphs: a neuro dynamic programming approach. *Proceedings of the 42nd IEEE Conference on Decision and Control*, 2003.
- [8] J.L. Bander and C.C. White III. A heuristic search approach for a nonstationary stochastic shortest path problem with terminal cost. *Transportation Science*, 36(2):218–230, 2002.
- [9] A. Bar-Noy and B. Schieber. The canadian traveller problem. *SODA '91: Proceedings of the Second Annual ACM-SIAM Symposium on Discrete algorithms*, 1991.
- [10] R. Bellman. *Dynamic Programming*. Dover Publications, Mineola, NY, 2003.
- [11] D.P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice Hall, Englewood Cliffs, NJ, 1987.
- [12] D.P. Bertsekas and J.N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.
- [13] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, MA, 1996.

- [14] D. Blatt and A.O. Hero. Optimal sensor scheduling via classification reduction of policy search. *ICAPS Workshop on POMDP, Classification and Regression: Relationships and Joint Utilization*, 2006.
- [15] D.M. Blei and L.P. Kaelbling. Shortest paths in a dynamic uncertain domain. *IJCAI Workshop on Adaptive Spatial Representations of Dynamic Environments*, 1999.
- [16] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1):49–107, 2000.
- [17] A.R. Cassandra. *Exact and approximate algorithms for partially observable Markov decision processes*. PhD thesis, Brown University, Providence, RI, 1998.
- [18] C.R. Chang and J.R. Slagle. An admissible and optimal algorithm for searching and/or graphs. *Artificial Intelligence*, 2:117–128, 1971.
- [19] H.S. Chang and S.I. Marcus. Approximate receding horizon approach for Markov decision processes: average reward case. *Journal Of Mathematical Analysis And Applications*, 286(2):636–651, 2003.
- [20] T.S. Chang, L.K. Nozick, and M.A. Turnquist. Multiobjective path finding in stochastic dynamic networks, with application to routing hazardous materials shipments. *Transportation Science*, 39(3):383–399, 2005.

- [21] G.A. Corea and V.G. Kulkarni. Shortest paths in stochastic networks with arc lengths having discrete-distributions. *Networks*, 23(3):175–183, 1993.
- [22] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1992.
- [23] J.S. Croucher. A note on the stochastic shortest-route problem. *Naval Research Logistics*, 25(4):729–732, 1978.
- [24] Y.Y. Fan, R.E. Kalaba, and J.E. Moore. Shortest paths in stochastic networks with correlated link costs. *Computers and Mathematics With Applications*, 49(9–10):1549–1564, 2005.
- [25] D.P. De Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.
- [26] D.P. De Farias and B. Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics Of Operations Research*, 29(3):462–478, 2004.
- [27] J. Fawcett and P. Robinson. Adaptive routing for road traffic. *IEEE Computer Graphics Applications*, 20(3), 2000.
- [28] D. Ferguson, A. Stenz, and S. Thrun. Pao* for planning with hidden state. *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, 2004.

- [29] D.E. Fishkind, C.E. Priebe, K. Giles, L.N. Smith, and V. Aksakalli. Disambiguation protocols based on risk simulation. *IEEE Transactions on Systems, Man, and Cybernetics*, Part A, to appear, 2006.
- [30] S. Gao and I. Chabini. Best routing policy problem in stochastic time-dependent networks. *Transportation Research Record*, pages 188–196, 2002.
- [31] S. Gao and I. Chabini. Optimal routing policy problems in stochastic time-dependent networks. *Transportation Research Part B-Methodological*, 40(2):93–122, 2006.
- [32] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal Of Artificial Intelligence Research*, 19:399–468, 2003.
- [33] R. Hall. The fastest path through a network with random time-dependent travel times. *Transportation Science*, 20(3):182–188, 1986.
- [34] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost graphs. *IEEE Transactions on SSC*, 4:100–107, 1968.
- [35] P.E. Hart, N.J. Nilsson, and B. Raphael. Correction to “a formal basis for the heuristic determination of minimum cost graphs”. *SIGART Newsletter*, 37:28–29, 1972.

- [36] M. Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal Of Artificial Intelligence Research*, 13:33–94, 2000.
- [37] L. He, S. Ji, and L. Carin. Application of partially observable Markov decision processes to robot navigation in a minefield. *ICAPS Workshop on POMDP, Classification and Regression: Relationships and Joint Utilization*, 2006.
- [38] A.O. Hero and O. Michel. Asymptotic theory of greedy approximations to minimal k -point random graphs. *IEEE Transactions on Information Theory*, 45:1921–1939, 1999.
- [39] Q.A. Holmes, C.R. Schwartz, J.A. Wright J.H. Seldin, and L.J. Wieter. Adaptive multispectral CFAR detection of land mines. *Proceedings of the SPIE*, 2496:421–432, 1995.
- [40] L.P. Kaelbling, M.L. Littman, and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.
- [41] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232, 2002.
- [42] S. Kim, M.E. Lewis, and C.C. White. Optimal vehicle routing with real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems*, 6(2):178–188, 2005.

- [43] H. Li, X. Liao, and L. Carin. Region-based value iteration for partially observable Markov decision processes. *Proceedings of the 2006 International Conference on Machine Learning*, 2006.
- [44] H. Li, X. Liao, and L. Carin. Incremental least squares policy iteration for POMDPs. *American Association for Artificial Intelligence*, to appear, 2006.
- [45] M. Likhachev, G. Gordon, and S. Thrun. planning for Markov decision processes with sparse stochasticity. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2005.
- [46] M.L. Littman, A.R. Cassandra, and L.P. Kaelbling. Learning policies for partially observable environments: Scaling up. *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.
- [47] T. Lozano-Perez and M.A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22:560–570, 1975.
- [48] A. Martelli and U. Montanari. Optimizing decision trees through heuristically guided search. *Communications of ACM*, 21:1025–10039, 1978.
- [49] E. Miller-Hooks. Adaptive least-expected time paths in stochastic, time-varying transportation and data networks. *Networks*, 37(1):35–52, 2001.

- [50] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender. Complexity of finite-horizon Markov decision process problems. *Journal of the Association for Computing Machinery*, 47(4):681–720, 2000.
- [51] G. L. Nemhauser. *Dynamic Programming*. John Wiley and Sons, Inc, NY, 1967.
- [52] N.J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann, Palo Alto, CA, 1980.
- [53] T. Olson, J.S. Pang, and C.E. Priebe. A likelihood-mpec approach to target classification. *Mathematical Programming*, 96:1–31, 2002.
- [54] C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.
- [55] C.H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127–150, 1991.
- [56] C.D. Piatko, C.P. Diehl, P. McNamee, C. Resch, and I.J. Wang. Stochastic search and graph techniques for mcm path planning. *Proceedings of the SPIE*, 4742:583–592, 2002.
- [57] C.D. Piatko, C.E. Priebe, L.J. Cowen, I.J. Wang, and P. McNamee. Path planning for mine countermeasures command and control. *Proceedings of the SPIE*, 4394:836–843, 2001.

- [58] J. Pineau, G. Gordon, and S. Thurn. Point-based value iteration: An anytime algorithm for POMDPs. *Proceedings of the International Joint Conference on Artificial Intelligence*, 2003.
- [59] G.H. Polychronopoulos and J.N. Tsitsiklis. Stochastic shortest path problems with recourse. *Networks*, 27(2):133–143, 1996.
- [60] C.E. Priebe, D.E. Fishkind, L. Abrams, and C.D. Piatko. Random disambiguation paths for traversing a mapped hazard field. *Naval Research Logistics*, 52:285–292, 2005.
- [61] C.E. Priebe, D.Q. Naiman, and L. Cope. Importance sampling for spatial scan analysis: computing scan statistic p-values for marked point processes. *Computational Statistics and Data Analysis*, 35(4), 2001.
- [62] C.E. Priebe, J.S. Pang, and T. Olson. Optimizing sensor fusion for classification performance. *Proceedings of the CISST 1999 International Conference*, pages 397–403, 1999.
- [63] J.S. Provan. A polynomial-time algorithm to find shortest paths with recourse. *Networks*, 41(2):115–125, 2003.
- [64] M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc, NY, 2005.
- [65] N.S.V. Rao. Robot navigation in unknown generalized polygon terrains using

- vision sensors. *IEEE Transactions on Systems, Man, and Cybernetics*, 25:947–962, 1995.
- [66] N.S.V. Rao and S.S. Iyengar. Autonomous robot navigation in unknown terrains: incidental learning and environmental exploration. *IEEE Transactions on Systems, Man, and Cybernetics*, 20:1443–1449, 1990.
- [67] S.M. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, CA, 1983.
- [68] N. Roy, G. Gordon, and S. Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.
- [69] J. Schulte and S. Thrun. A heuristic search algorithm for acting optimally in Markov decision processes with deterministic hidden state. 2001.
- [70] D.L. Smith. Detection technologies for mines and minelike targets. *Proceedings of the SPIE*, 2496:404–408, 1995.
- [71] T. Smith and R. Simmons. Focused real-time dynamic programming for MDPs: squeezing more out of a heuristic. *Proceedings of AAAI*, 2006.
- [72] E. Sondik. *The optimal control of partially observable Markov decision processes*. PhD thesis, Stanford University, Stanford, CA, 1971.

- [73] S.T. Waller and A.K. Ziliaskopoulos. On the online shortest path problem with limited arc cost dependencies. *Networks*, 40(4):216–227, 2002.
- [74] D.J. White. *Markov Decision Processes*. John Wiley and Sons, Inc, NY, 1993.
- [75] D. Wingate and K.D. Seppi. Prioritization methods for accelerating MDP solvers. *Journal Of Machine Learning Research*, 6:851–881, 2005.
- [76] N.H. Witherspoon, J.H. Holloway, K.S. Davis, R.W. Miller, and A.C. Dubey. The coastal battlefield reconnaissance and analysis (COBRA) program for minefield detection. *Proceedings of the SPIE*, 2496:500–508, 1995.
- [77] Y. Ye. A new complexity result on solving the Markov decision problem. *Mathematics of Operations Research*, 30(3):733–749, 2005.