Optimal Ship Navigation and Algorithms for Stochastic Obstacle Scenes

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

İbrahim ARI

in partial fulfillment for the
degree of Master of Science

in
Industrial and Systems Engineering

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Industrial and Systems Engineering

APPROVED BY:

Assist. Prof. Dr. Vural Aksakalli .....
(Thesis Advisor)

Assist. Prof. Dr. Hatice Tekiner Moğulkoç .....

Assist. Prof. Dr. Ali Fuat Alkaya .....

This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences of Istanbul Şehir University:

DATE OF APPROVAL: 19 August 2013

SEAL/SIGNATURE:

Declaration of Authorship

I, İbrahim ARI, declare that this thesis titled, 'Optimal Ship Navigation and Algorithms for Stochastic Obstacle Scenes' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 27.12.2013

"Raise your words, not voice. It is rain that grows flowers, not thunder."

Rumi

Optimal Ship Navigation and Algorithms for Stochastic Obstacle Scenes

İbrahim ARI

Abstract

This thesis is comprised of two different but related sections. In the first section, we consider the optimal ship navigation problem wherein the goal is to find the shortest path between two given coordinates in the presence of obstacles subject to safety distance and turn-radius constraints. These obstacles can be debris, rock formations, small islands, ice blocks, other ships, or even an entire coastline. We present a graph-theoretic solution on an appropriately-weighted directed graph representation of the navigation area obtained via 8-adjacency integer lattice discretization and utilization of the A^* algorithm. We explicitly account for the following three conditions as part of the turn-radius constraints: (1) the ship's left and right turn radii are different, (2) ship's speed reduces while turning, and (3) the ship needs to navigate a certain minimum number of lattice edges along a straight line before making any turns. The last constraint ensures that the navigation area can be discretized at any desired resolution. We illustrate our methodology on an ice navigation example involving a 100,000 DWT merchant ship and present a proof-of-concept by simulating the ship's path in a full-mission ship handling simulator at Istanbul Technical University.

In the second section, we consider the stochastic obstacle scene problem wherein an agent needs to traverse a spatial arrangement of possible-obstacles, and the status of the obstacles may be disambiguated en route at a cost. The goal is to find an algorithm that decides what and where to disambiguate en route so that the expected length of the traversal is minimized. We present a polynomial-time method for a graph-theoretical version of the problem when the associated graph is restricted to parallel avenues with fixed policies within the avenues. We show how previously proposed algorithms for the continuous space version can be adapted to a discrete setting. We propose a generalized framework encompassing these algorithms that uses penalty functions to guide the navigation in realtime. Within this framework, we introduce a new algorithm that provides near-optimal results within very short execution times. Our algorithms are illustrated via computational experiments involving synthetic data as well as an actual naval minefield data set.

Keywords: Graph theory, shortest path, ship navigation, probabilistic path planning, stochastic dynamic programming, Markov decision process, Canadian traveler's problem

Optimal Gemi Navigasyonu ve Stokastik Engelli Ortamlar için Algoritmalar

İbrahim ARI

Öz

Bu tez araştırması iki farklı fakat alakalı bölümden oluşmaktadır. Birinci bölümde, optimal gemi navigasyon problemini göz önüne aldık. Bu problemde amaç, engellerin bulunduğu bir alanda emniyet mesafesi ve dönüş-yarıçapı kısıtı altında verilen iki koordinat arasında en kısa yolu bulmaktır. Bahsi geçen engeller enkaz, kaya formasyonları, küçük adalar, buz blokları, diğer gemiler ve hatta tüm kıyı şeridi olabilir. Gemi navigasyon problemi için 8-komşulu tamsayı örgü ayrıklaştırması üzerinde graf-teori tabanlı bir çözüm sunduk ve A* algoritmasından faydalandık. Sunduğumuz çözümde, dönüş-yarıçapı kısıtı için şu üç koşulu açıkça hesaba katılmıştır: (1) gemilerin iskele (sol) ve sancak (sağ) dönüş yarıçapları farklı olması, (2) dönüşlerde geminin hızının düşmesi, (3) dönüş yapmadan önce belirli bir mesafe aynı doğrultuda gitmesidir. Üçüncü kısıt navigasyon alanının istenilen çözünürlükte ayrıklaştırmasına imkan tanır. Optimal (ayrık) yol belirlendikten sonra keskin dönüşleri, gerçek gemi dönüşlerine benzetmek için yumuşatma işlemi yaptık. Bu metodolojimizi 100.000 DWT ticari bir geminin buzlu denizlerdeki navigasyonu örneği üzerinde test ettik ve İstanbul Teknik Üniversitesi'nde bulunan Tam Donanımlı Köprü-üstü (TDK) Simülatöründe konsept-ispatını sunduk.

İkinci bölümde, stokastik engelli ortamlar problemini göz önüne aldık. Bu problemde bir ajan, olası-engellerin olduğu bir bölgede hedef bir konuma ulaşmak için yol almalıdır ve yolculuk esnasında olası-engellerin belirsizliği bir ücret karşılığında giderilebilir. Buradaki amaç, gidilen yolu en kısa yapacak şekilde hangi engelin neresinden belirsizlik gidermenin yapılacağını belirleyen bir algoritma bulmaktır. Bu problemin belirli politikalar altındaki paralel caddelerle kısıtlanmış durumu bağlamındaki graf-teori tabanlı versiyonu için polinom-zamanlı bir yöntem sunduk. Bunun yanında, problemin sürekli ortamlardaki versiyonu için sunulan algoritmaların nasıl ayrık ortamlara uyarlanacağını gösterdik. Bu kısımda, navigasyon kılavuzu olarak penaltı fonksiyonlarını kullanan algoritmaları kapsayacak şekilde genel bir çatı önerdik. Bu çatı içerisinde, çok kısa koşma süresinde optimala yakın değerler veren yeni bir algoritmayı tanıttık. Bu algoritmamızı, sentetik veri üzerinde olduğu gibi gerçek deniz mayın tarlası verisi üzerinde de hesaplamasal deneylerle test ettik.

Anahtar Sözcükler: Graf teori, en kısa yol, gemi navigasyonu, rassal yol planlama, stokastik dinamik programlama, Markov karar süreci, Kanadalı gezgin problemi

This thesis is dedicated to my mother Semine and my sister Sema. Their love, encouragement, and endless support throughout my life have made me stronger.

Acknowledgments

I would like to express my deepest appreciation to my thesis adviser Prof. Vural Aksakallı for his support, guidance, and patience. I am grateful for everything I learned from him and everything he has done for me. Prof. Aksakallı was also the PI of the TUBITAK Project Grant No. 111M541 that supported me during the last three semesters of my studies.

Section 1.6 is joint work with Prof. Serdar Kum and Prof. Volkan Aydoğdu from Maritime Faculty at Istanbul Technical University. I would like to thank them for performing the simulations and providing me the output of a full-mission ship handling simulator.

Special thanks goes to Prof. Hatice Tekiner Moğulkoç, and Prof. Ali Fuat Alkaya from Department of Computer Engineering at Marmara University, for serving on my thesis committee.

I want to thank our department chair Prof. Muammer Koç for his invaluable advice and guidance. I would like to express my gratitude to Prof. Özdal Boyraz for the valuable experience I had as a teaching assistant under his supervision.

Many thanks to the faculty members of the Industrial Engineering Department for everything I learned from them, and special thanks to faculty secretary Ms. Seçil Öçal for her help regarding administrative issues. And finally, many thanks to all my friends at the ISE program for making this past two years a fun and memorable one.

Contents

List of Figures

List of Tables

Chapter 1

Optimal Ship Navigation with Safety Distance and Realistic Turn Constraints

1.1 Introduction

Seaborne shipping is a major form of transportation that accounts for about 90% of world's trade. As of 2011, there are more than 100,000 seagoing commercial ships in the world transporting over 8,000 million tons of cargo each year [1]. As the world's population grows and countries increase their participation in international commerce, seaborne shipping continues to expand as a low cost, acceptable risk, and environment friendly form of transportation. In this chapter, we consider a ship navigation problem wherein the objective is find the (time-wise) shortest path from a given starting point s to a termination point t in the presence of obstacles subject to (i) safety distance and (ii) turn-radius constraints. We define an obstacle as any region in any shape or size that the ship needs to avoid in its $s-t$ voyage. These obstacles can be debris, rock formations, small islands, ice blocks, other ships, or even an entire coastline. We assume that the obstacles are static, i.e., they do not move or change shape during the ship's navigation, and we do not take into consideration environmental effects such as winds, waves, or sea currents. Our methodology involves directed 8-adjacency integer lattice discretization of the navigation area and utilization of the A^* algorithm on the resulting graph.

A novel aspect of our research is that we account for the following three real-world ship navigation phenomena as part of the turn-radius constraints: (1) the ship's port (left) and starboard (right) turn radii are different, (2) ship's speed reduces while turning, and (3) the ship needs to navigate a certain minimum number of lattice edges along a straight line before altering course. The third constraint ensures that the navigation

area can be discretized at any resolution needed. These three constraints together will be referred to as *the ship-turn constraints*. In addition, we fully parameterize turn-radius and safety distance constraints in the following sense: (i) different (left and right) turn-radii and turn-speed can be specified based on the particular characteristics of the ship, and (ii) safety distances can be defined at the obstacle level based on the nature of the obstacle. Implementation of our methodology requires non-trivial modifications to the underlying graph in order to preserve optimality. These modifications include making certain number of copies of each vertex, defining an appropriate neighborhood structure, and assigning edge lengths accordingly. Once the optimal (discrete) path is determined, we smoothen it to emulate the actual (continuous) navigation of the ship. We illustrate our methodology on an ice navigation example in a full-mission ship handling simulator with a 100,000 DWT full-load-condition tanker ship and present a proof-of-concept by simulating the ship's actual path. Here, DWT stands for deadweight tonnes, which is a measure of how much the tanker can carry safely including cargo, fuel, fresh water, and passengers [2].

There exists a vast amount of literature on deterministic shortest paths and ship navigation—both in continuous and discrete settings. For the most part, continuous-space studies on ship navigation involve complex differential equations and/or calculus of variation with curvature constraints. These types of approaches typically do not scale well in the presence of a large number of arbitrarily-shaped obstacles and ship-turn constraints. Existing discrete-space studies, on the other hand, impose overly simplistic turn constraints while ignoring safety distance requirements. To our knowledge, ours is the first study in the literature that accounts for safety distance and the ship-turn constraints as described above in a graph-theoretical framework that also allows for full parametrization of these constraints.

The rest of this chapter is organized as follows: Section 1.2 provides an overview of previous work on ship navigation (both in continuous and discrete settings) and reviews existing studies on turn constraints. Section 1.3 formally defines the optimal ship navigation problem. Section 1.4 presents our navigation methodology in detail including the lattice discretization, modeling of the safety distance and turn constraints, and smoothing of the optimal path. Section 1.5 demonstrates our approach on an ice navigation example and Section 1.6 provides the output of a full-mission ship handling simulator. Summary, conclusions, and directions for future research are presented in Section 1.7.

1.2 Previous Work

In many real-world applications, a challenging yet critical task is to find shortest paths for wheeled vehicles, aircrafts, and ships in different terrains subject to various operational constraints. Hence, path planning problems and deterministic shortest paths in general

have been studied extensively; particularly within the fields of transportation science, operations research, artificial intelligence, and robotics. Arguably, the most commonly used shortest path methods in the literature are the Dijkstra's Algorithm [3] and the A^* Algorithm [4, 5].

Dubins' car [6] is an example of a forward-moving wheeled vehicle with a maximum turn angle. (Note that this turn angle is with respect to the vehicle's present direction and that a maximum turn angle constraint can be converted to an equivalent minimum turn-radius constraint.) Path planning problems for Dubins' car were primarily studied within the context of non-holonomic path planning [7]. These types of problems are typically modeled as partial differential equation systems in continuous space and solved by numerical methods. This solution approach, however, is not effective in general due to the difficulty in incorporation of real-world physical constraints [8]. The rapidly-exploring random tree (RRT) method of LaValle [9] is an effective continuous-space algorithm for Dubins' car that can handle a wide range of environmental constraints. However, the path obtained by the RRT method is not necessarily optimal as it is based on generation of random way-points over the course of navigation [10]. To our knowledge, there are currently no available non-holonomic models that can be utilized for application of the RRT method for ship navigation. In addition, incorporating asymmetric left and right turn constraints as well as decreased speeds during turns into RRT seem to be rather difficult.

Ship navigation in the presence of obstacles is inherently a continuous-space problem. However, incorporation of realistic operational constraints in a continuous setting is a challenging task. Therefore, previous researches on this topic primarily focused on discretization of the navigation area in various ways. A major advantage of a discretization approach is that it allows for utilization of well-established and extremely rich machinery of graph theory and network flows. For instance, the work of Fagerholt et al. [11] on ship navigation utilizes a visibility graph discretization where the graph is constructed only partially during the solution process for improved efficiency. The study by Lee et al. [12], on the other hand, uses pre-specified way points for discretization and employs a modified depth-first search algorithm. Neither of these studies consider any turn or safety distance constraints.

Regarding turn constraints for ships, their incorporation in a continuous setting requires nonlinear maneuvering equations [13–15] and is difficult in general [16]. In graph-theoretical settings, generic maximum turn angle constraints in the literature seem to be limited to symmetric one-edge ahead turn constraints. However, such a limitation implies that resolution of the navigation area is essentially dictated by the turn-radius, which in turn, eliminates any possibility of working with a finer resolution for improved accuracy. These one-edge ahead turn constraints were modeled in various ways such as (1) vertex replication [17–21], (2) modification of the Dijkstra's Algorithm [22, 23], and (3) transformation of the original graph [24–29]. A comparison of the vertex replication

method and modified Dijkstra's Algorithm for road networks can be found in Vanhove and Fack [30]. Our methodology is based on the vertex replication technique where we split each vertex into copies labeled by the direction the ship is coming from as well as the distance traveled. Thus, immediate navigation history is incorporated into the present location, which in turn enables enforcement of the ship-turn constraints as defined in Section 1.1.

A research area closely related to ours is mission planning for routing of military aircraft and unmanned aerial vehicles [31, 32]. Some of these studies also consider minimum turn radius constraints [25, 33–35]. Such mission planning studies, however, are not readily adaptable to ship navigation problems due to the fact that ships differ considerably from aircrafts and aerial vehicles with respect to their technological and operational constraints. In particular, aerial mission planning research often takes into account limited fuel storage constraints, which fundamentally changes the structure of the underlying problem and makes it computationally intractable [25]. On the other hand, such fuel capacity constraints do not typically exist in ship navigation.

1.3 The Optimal Ship Navigation Problem

In this section, we formally define the optimal ship navigation problem in the presence of obstacles, or the OSN problem in short. Without loss of generality, we assume that the navigation area is rectangular. In addition, we only consider polygon-shaped obstacles. This should not be seen as a limitation since a polygon approximation can be used to represent any geometric shape at any level of accuracy [36]. We also assume that these polygons are non-self-intersecting as self-intersecting polygons can easily be transformed into non-self-intersecting ones without changing the geometric shape of the obstacle that the polygon represents. Nonetheless, we allow for non-convex and overlapping polygons. The terms polygon and obstacle shall be used synonymously in the rest of this work. The notation in Table 1.1 characterizes our ship navigation model. In the model, one mile refers to one nautical mile, which corresponds to 1.151 (land) miles or 1.852 kilometers. Ship's speed is expressed in knots, with one knot representing a speed of one nautical mile per hour.

For any obstacle $p \in P$, its buffer zone B_p is defined as the region whose boundary is comprised of points that are d_p meters away from the closest point on the obstacle's boundary. In our model, we require that the ship does not enter the buffer zone of any obstacle, which we call the *safety distance constraint*. The ship's left and right turning diameters are defined as $\ell \times c_l$ and $\ell \times c_r$ respectively where ℓ is the ship's length between perpendiculars (LBP) in meters. LBP refers to the length of the ship along the water line from the forward-surface of the stem to the after-surface of the sternpost. This method for specifying a ship's turn radius is typical in seaborne navigation [37]. The

ship's turning angle at any point (with respect to its present direction) is constrained to be upper bounded by that of the curvature of the circle with respective left and right turn diameters.

TABLE 1.1: Notation characterizing our ship navigation model.

Notation	Description
A	The (rectangular) navigation area
A_x	Length of the navigation area in nautical miles
A_y	Width of the navigation area in nautical miles
P	The set of (polygon-shaped) obstacles indexed by p
R_p	Region associated with obstacle p
B_p	Buffer zone associated with obstacle p
d_p	Safety distance in meters associated with obstacle p
s	Starting point
t	Termination point
ℓ	Ship's length between perpendiculars (LBP) in meters
c_r	Ship's right turn coefficient
c_l	Ship's left turn coefficient
k	Ship's speed in knots while straight navigation
k'	Ship's speed in knots while turning such that $k' < k$

The *Optimal Ship Navigation (OSN) Problem* is then defined as follows: Given a set of obstacles P inside the navigation area A, a starting point s and a destination point t, find the time-wise shortest $s-t$ path for a ship with straight (non-turn) speed k, turn speed k', LBP ℓ, left-turn coefficient c_l, and right-turn coefficient c_r subject to safety distance and ship-turn constraints. Note that due to reduced turning speeds, time-wise shortest paths are likely to be different than Euclidean-distance shortest paths.

1.4 Methodology

In this section, we first present an algorithm that computes safety buffer zones around each obstacle. We then describe our lattice discretization and explain implementation of the ship-turn constraints. Next, we illustrate how the optimal (discrete) path can be smoothed in order to emulate the actual (continuous) navigation of the ship.

1.4.1 Safety Distance Constraints

As mentioned earlier, this chapter assumes that (polygon-shaped) obstacles inside the navigation area are static. In reality, however, obstacles such as ice blocks or anchored ships may move slightly due to environmental conditions such as winds, waves, or sea currents. Unexpected changes in these conditions may also cause a certain deviation in the ship's planned course. In addition, it might be the case that vicinity of an obstacle is unsafe due to the nature of the obstacle. For instance, under-water sections of icebergs are sometimes dangerous for close-by ship navigation. Due to these reasons,

even though obstacles are assumed to be static, we allow for a buffer zone around each obstacle. These zones are defined as the regions whose boundary is comprised of points that are d_p meters away from the closest point on obstacle p's boundary.

Our methodology for computing buffer zones is based on defining a sequence of uniformly-spaced vertices for each pair of edges in the original polygon such that the safety distance constraint is satisfied. The outcome of the process is another polygon that satisfies the safety distance constraint at all points, which we call the *buffer zone polygon* and denote by B_p for obstacle p. A step angle, denoted by δ, is used to determine the number of vertices to be defined for each pair of edges. The step angle δ is taken as 3 degrees in our implementation. As an example, suppose the angle between one pair of edges for an obstacle is 120 degrees, meaning that the angle between these edges' normal vectors is 60 degrees. For the resulting buffer zone polygon, we define uniformly-spaced $60/3 = 20$ vertices in total corresponding to this edge pair. Provided in Figure 1.1 is a pseudo-code of our algorithm for buffer zone polygon calculation and illustrated in Figure 1.2 are two examples. In the rest of this chapter, we will only be concerned with buffer zone polygons as what the ship needs to avoid in its $s-t$ voyage are these buffer zones—not the actual obstacles.

1.4.2 Lattice Discretization

As discussed earlier, identification of the shortest path in the OSN problem in a continuous setting is a rather difficult task. Therefore, we consider a discrete approximation of the continuous setting on a subgraph of the 8-adjacency integer lattice. In particular, this discretization is the directed graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ whose vertices are all of the pairs of integers i, j such that $1 \leq i \leq x_{\max}$ and $1 \leq j \leq y_{\max}$, where $x_{\max}$ and $y_{\max}$ are given integers. There are directed edges between all pairs of the following four types of vertices: (1) (i, j) and $(i+1, j)$ with unit length, (2) (i, j) and $(i, j+1)$ with unit length, (3) (i, j) and $(i+1, j+1)$ with length $\sqrt{2}$ units, and, (4) $(i+1, j)$ and $(i, j+1)$ with length $\sqrt{2}$ units. One vertex in G is designated as the starting point s, another vertex in G is designated as the termination point t. The ship is to traverse from s to t in G, only using edges whose start and end vertices are both outside of buffer polygons of any obstacle in P. Consistent with our lattice discretization, we only consider 45-degree turns in this work.

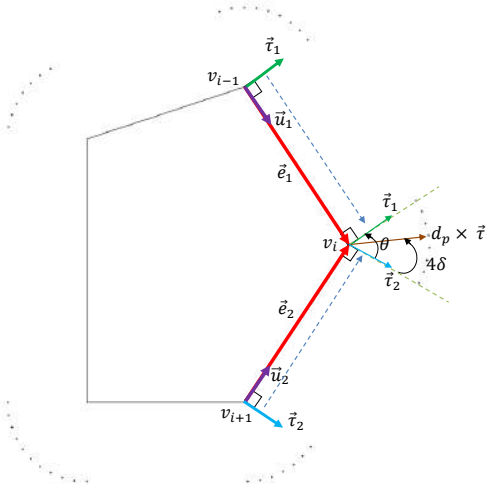
Unit (i.e., non-diagonal) edge length in the lattice, denoted by α, is determined by a one-mile-resolution-factor f such that $\alpha = 1/f$. For instance, $f = 20$ implies that one (nautical) mile corresponds to 20 unit edges with $\alpha = 1/20 = 0.05$ miles, or about 93 meters. Diagonal edge length is then computed as $\sqrt{2}\alpha \simeq 131$ meters. Now, suppose that the navigation area is $A_x = 12$ miles by $A_y = 8$ miles. With $f = 20$, $x_{\max}$ and $y_{\max}$ can be computed as $x_{\max} = f \times A_x = 240$ and $y_{\max} = f \times A_y = 160$. This setup shall be used in our subsequent illustrations and examples.

Buffer Zone Polygon Calculation Algorithm for Obstacle p
Input: Set of clock-wise ordered vertices W_p indexed by $w_i = 1, \ldots, |W_p|$, safety distance $d_p \in \mathbb{R}^+$, step angle δ.
Output: Set of vertices W_{B_p} corresponding to the buffer-zone polygon B_p.
Begin
Initialize $W_{B_p} := \emptyset$
Set $n := |W_p|$
Define $w_0 := w_n$ and $w_{n+1} := w_1$ (boundary conditions)
For $i := 1$ to n do
Set $\tilde{w}_0 := w_i$, $\tilde{w}_1 := w_{i-1}$ and $\tilde{w}_2 := w_{i+1}$
Set $\vec{e}_1 := \tilde{w}_0 - \tilde{w}_1$ and $\vec{e}_2 := \tilde{w}_0 - \tilde{w}_2$
Compute unit vectors $\vec{u}_1$ and $\vec{u}_2$ for $\vec{e}_1$ and $\vec{e}_2$, respectively
Compute direction vectors $\overrightarrow{\tau_1}$ and $\overrightarrow{\tau_2}$ that are rotations of $\vec{u}_1$ and $\vec{u}_2$ by angle $\pi/2$ and
$-\pi/2$, respectively
Set $\theta :=$ angle between $\vec{\tau_1}$ and $\vec{\tau_2}$
If $\theta < \pi$ and $\theta > \delta$
Set $\phi := \theta$
Set $\vec{\tau} := \vec{\tau_2}$
While $\phi \geq 0$
Set $\vec{\tau} :=$ Vector $\vec{\tau}$ rotated counter-clockwise by δ degrees.
Set $\tilde{w} := \tilde{w}_0 + d_p \times \overrightarrow{\tau}$
Put $\tilde{w}$ in W_{B_p}
Set $\phi := \phi - \delta$
End while
Else ($\theta \geq \pi$ or $\theta \leq \delta$)
Set $\tilde{w} := \tilde{w}_0 + d_p \times (\vec{\tau_1} + \vec{\tau_2})$
Put $\tilde{w}$ in W_{B_p}
End if
End for
End

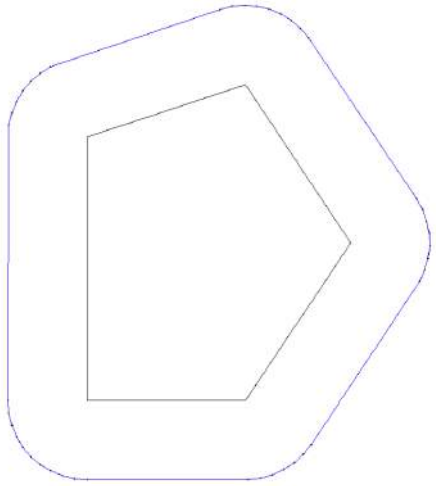
FIGURE 1.1: Buffer zone polygon calculation algorithm.

1.4.3 Ship-Turn Constraints

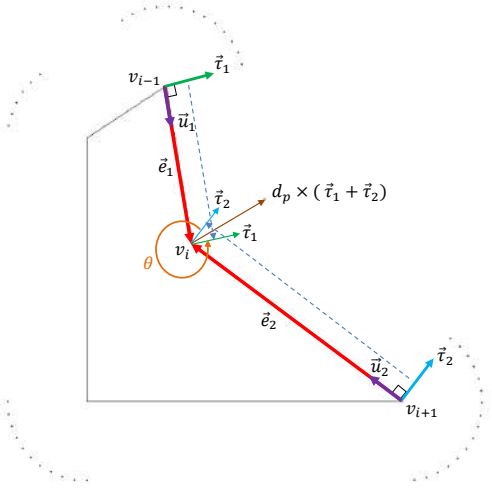
As mentioned earlier, our definition of ship-turn constraints is comprised of the following: (1) the ship's left and right turn radii are different, (2) ship's speed reduces while turning, and (3) the ship needs to navigate a certain minimum number of lattice edges along a straight line before altering course. In order to illustrate how these constraints can be implemented in practice, we shall use a typical 100,000 DWT merchant ship as a running example with the following characteristics: LBP $\ell = 232$ meters, right-turn coefficient $c_r = 4.7$, left-turn coefficient $c_l = 4.1$, cruising speed $k = 10.0$ knots, and turning speed $k' = 8.0$ knots [37].



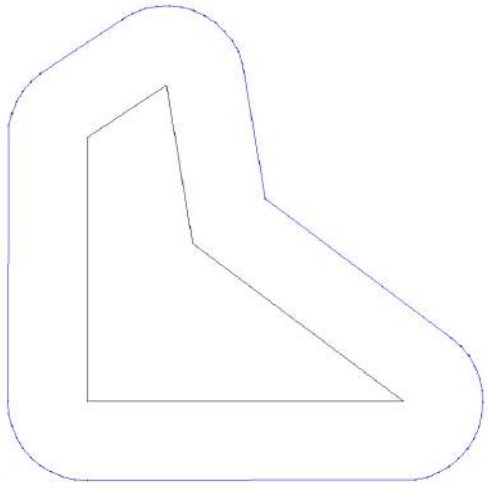
(a) Calculation with $\theta < \pi$.



(b) The resulting buffer zone polygon.



(c) Calculation with $\theta > \pi$.



(d) The resulting buffer zone polygon.

FIGURE 1.2: Two illustrations of buffer zone polygon calculation.

Right-turn diameter for the above ship is calculated as $\ell \times c_r = 232 \times 4.7 \simeq 1,090$ meters whereas the left-turn diameter is given by $\ell \times c_l = 232 \times 4.1 \simeq 951$ meters. In order to determine the number of edges in the lattice discretization corresponding to right and left turns, we first find the smallest octagons that fully contain circles with the right and left-turn diameters respectively. Edge lengths of these octagons are computed using elementary geometry as illustrated in Figure 1.3 for the right turn. Let γ_r and γ_l denote the edge length of the octagon corresponding to the right and left turns respectively. From the figure, we observe that $\gamma_r + 2\gamma_r/\sqrt{2} = 1090$, which implies $\gamma_r = 1090/(1+\sqrt{2}) \simeq 452$ meters. Likewise, $\gamma_l = 951/(1+\sqrt{2}) \simeq 394$ meters. For a right turn, 452 meters correspond to $452/93 \simeq 4.86$ or about 5 unit edges, and $452/131 \simeq 3.45$ or roughly 4 diagonal edges. Similarly, for a left turn, 394 meters

correspond to $394/93 \simeq 4.24$ or about 4 unit edges, and $394/131 \simeq 3$ diagonal edges. Thus, a right turn on the integer lattice consists of the following three steps: (1) navigate at least 5 unit or 4 diagonal edges, (2) make a 45-degree right turn, and (3) navigate again at least 5 unit 4 diagonal edges to complete the turn. Similarly, for a left turn, the ship needs to navigate at least 4 unit or 3 diagonal edges in the lattice, make a 45-degree left turn, and navigate again at least 4 unit or 3 diagonal edges. Each one these edge sequences before and after the turn shall be referred to as a "leg". It is important to note that the second leg of a turn can also be the first leg of the next turn.

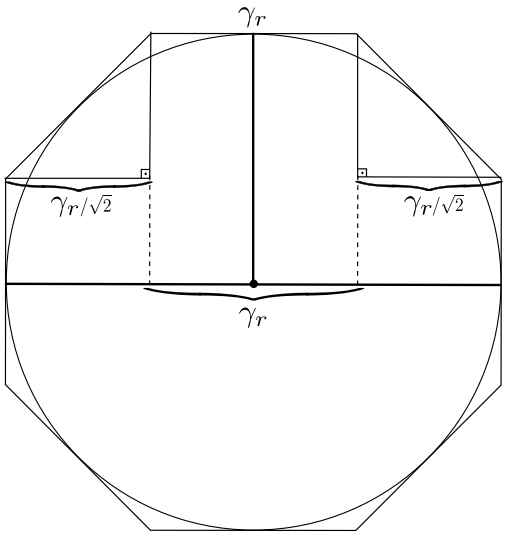


FIGURE 1.3: Calculation of γ_r. Observe that $\gamma_r + 2\gamma_r/\sqrt{2} = 1090$.

Having determined the minimum number of edges for right and left turns, the next task is to carefully define a new graph $\mathsf{G}' = (\mathsf{V}', \mathsf{E}')$ over which the turn-constrained navigation shall take place. Graph G' is constructed in such a way that it contains all legal paths and no illegal ones in order to preserve optimality. In this context, optimality is with respect to the underlying integer lattice—clearly, the optimal path on the lattice is not guaranteed to be optimal in the original continuous space. At this point, we observe that there are three types of edges in E':

1. **One-Hop Edges ("1H"):** Edges representing straight (non-turn) navigation. These correspond to usual unit or diagonal (or simply one-hop) edges in E.
2. **Right-Turn Edges ("RT"):** Edges representing one leg of a right turn.
3. **Left-Turn Edges ("LT"):** Edges representing one leg of a left turn.

The ship navigates at the normal speed $k = 10$ knots (18,520 meters per hour) along $1H$ edges, and at the reduced speed $k' = 8$ knots (14,816 meters per hour) along the RT and LT edges. Thus, time length of a unit $1H$ edge is $3600 \times 93/18520 \simeq 18$ seconds, and time length of a diagonal $1H$ edge is $3600 \times 131/18520 \simeq 25.5$ seconds. On the other hand, time length of a non-diagonal RT edge is $3600 \times 93 \times 5/14816 \simeq 113$ seconds

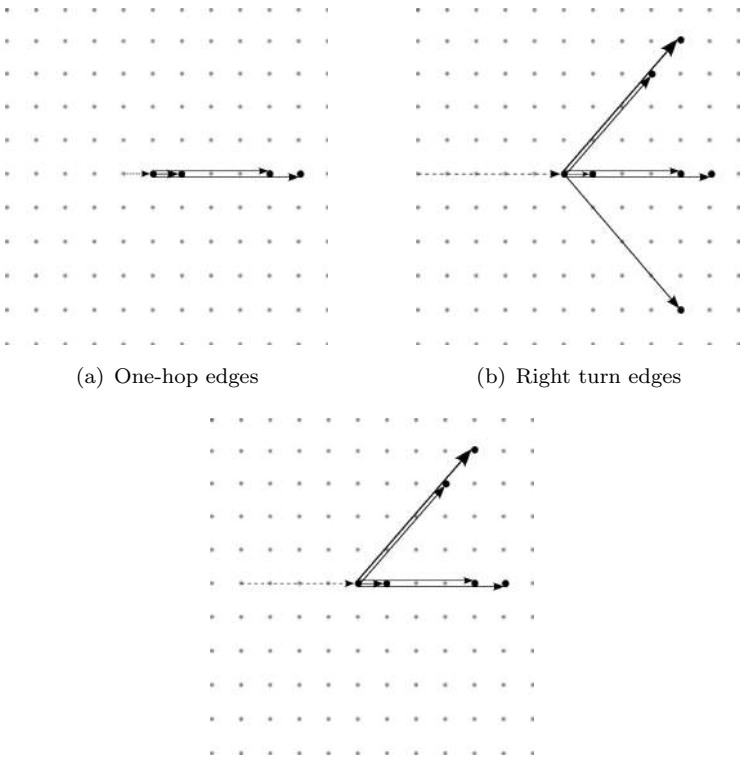
whereas time length of a diagonal RT edge is $3600 \times 131 \times 4/14816 \simeq 127.3$ seconds. Likewise, time length of a non-diagonal LT edge is $3600 \times 93 \times 4/14816 \simeq 90.4$ seconds whereas time length of a diagonal RT edge is $3600 \times 131 \times 3/14816 \simeq 95.5$ seconds. Our objective in the OSN problem is to find the time-wise shortest path. Thus, in the rest of the chapter, we will exclusively be concerned with time lengths, which we denote by $\varphi(u, v)$ for $(u, v) \in \mathsf{E}'$.

Our methodology for constructing G' is based on the idea of vertex replication where each vertex is split into copies labeled by the direction which the ship is coming from as well as the edge type. There are eight different directions (E,W,N,S,NE,SE,NW,SW), and three edge types, resulting in a total of 24 copies of each vertex in V. For convenience, coordinate information of each vertex copy $v' \in \mathsf{V}'$ will be augmented by the direction and the edge type information. As an example, we consider vertex copies corresponding to the lattice coordinate $x = 50, y = 50$ for each one of the three E' edge types with an east-bound direction. Figure 1.4 illustrates these copies and the edges emanating from them. The first vertex copy we examine is (50,50,W,1H), which represents one-hop straight navigation from west to east, i.e., from (49,50) to (50,50). Since one-hop is not sufficient for any turns, we define only three edges emanating from this vertex copy corresponding to further straight (non-turn) navigation with the following three end vertices:

1. **(51,50,W,1H):** The corresponding edge represents one-hop straight navigation. Edge length is 18 seconds.
2. **(55,50,W,RT):** The corresponding edge represents first leg of a right turn. Edge length is 113 seconds.
3. **(54,50,W,LT):** The corresponding edge represents first leg of a left turn. Edge length is 90.4 seconds.

Second vertex copy we consider is (50,50,W,RT)—representing 5 unit-edge navigation from west to east, that is, from (45,50) to (50,50). This much distance is sufficient for both right and left turns as well as further straight navigation. Therefore, we define six edges emanating from this vertex copy with the following six end vertices:

1. **(51,50,W,1H):** The corresponding edge represents one-hop straight navigation. Edge length is 18 seconds.
2. **(55,50,W,RT):** The corresponding edge represents first leg of a right turn. Edge length is 113 seconds.
3. **(54,50,W,LT):** The corresponding edge represents first leg of a left turn. Edge length is 90.4 seconds.



(a) One-hop edges (b) Right turn edges

(c) Left turn edges

FIGURE 1.4: Illustration of the edges emanating from vertex (50,50) with an east-bound direction.

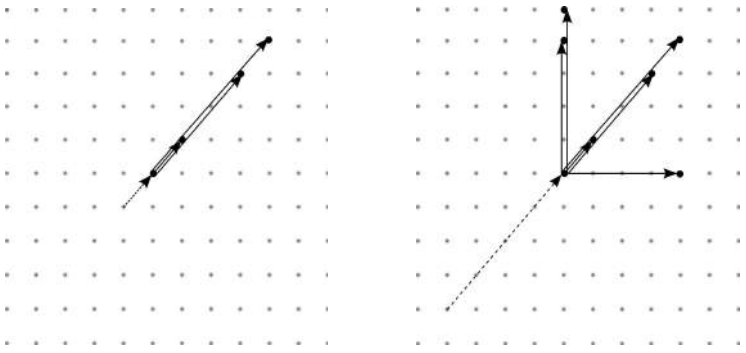
4. **(54,54,SW,RT):** The corresponding edge represents first leg of a right turn. Edge length is 127.3 seconds.

5. **(53,53,SW,LT):** The corresponding edge represents first leg of a left turn. Edge length is 95.5 seconds.

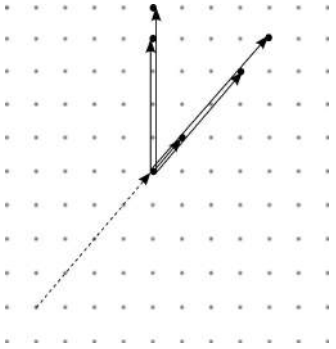
6. **(54,46,NW,RT):** The corresponding edge represents first or second leg of a right turn. Edge length is 127.3 seconds.

Third vertex copy we consider is (50,50,W,LT)—representing 4 unit-edge navigation from west to east, that is, from (46,50) to (50,50). This much distance is sufficient for a left turn as well as further straight navigation, but not long enough for a right turn. Therefore, we define five edges emanating from this vertex copy with the same end vertices as the first five vertices above for the (50,50,W,RT) copy. The sixth edge with the end point (54,46,NW,RT) is excluded as it would result in an illegal move. As a second set of example edge calculations, Figure 1.5 illustrates the process for the same lattice coordinate of $x = 50, y = 50$, this time with a northeast-bound direction.

Vertex copies for other directions and edges emanating from them are defined in a similar manner as above. In general, for any vertex copy, the following emanating edges need to be defined for each one of the 8 directions for the listed three edge types:



(a) One-hop edges (b) Right turn edges



(c) Left turn edges

FIGURE 1.5: Illustration of the edges emanating from vertex (50,50) with a northeast-bound direction.

1. **1H Edges:** Three straight navigation edges (one 1H, one RT, one LT edge).
2. **RT Edges:** Three straight navigation edges (one 1H, one RT, one LT edge), two 45-degree left turn edges (one RT, one LT), and one 45-degree right turn RT edge for a total of six edges.
3. **LT Edges:** Three straight navigation edges (one 1H, one RT, one LT edge) and two 45-degree left turn edges (one RT, one LT) for a total of five edges.

For a vertex copy, observe that for each one of the 8 directions, there are a total of 14 emanating edges. Thus, $|\mathsf{V}'| = 24|\mathsf{V}|$ and $|\mathsf{E}'| = (8 \times 14)|\mathsf{V}|$. In our example setting, we let the starting point for the navigation be (1,80) and the termination point be (240,80). A desirable feature of our methodology is that it allows for specifying an initial direction at the starting point and an approach direction at the destination point. For instance, if the ship is required to approach the termination point from the NW direction, the A^* algorithm can be terminated as soon as any one of the (240,80,NW,1H), (240,80,NW,RT), or (240,80,NW,LT) vertices is permanently labeled. In our implementation, we specify an west-east direction at the starting point and allow for an arbitrary approach direction at the termination point. Thus, we start the algorithm at the vertex (1,80,W,1H) and terminate the algorithm whenever any vertex copy at the (240,80) coordinate is permanently labeled.

We conclude this section by pointing out to the following fact regarding optimality of our methodology: the A^* algorithm that we use to find the shortest $s-t$ path on G' correctly identifies the optimal path (with our choice of the heuristic function) as discussed below. In addition, via careful definition of the edges in E', we ensure that this graph embeds all possible legal traversals and no illegal paths. Thus, A^* algorithm stands proven and no separate optimality proof is needed.

1.4.4 The A^* Algorithm

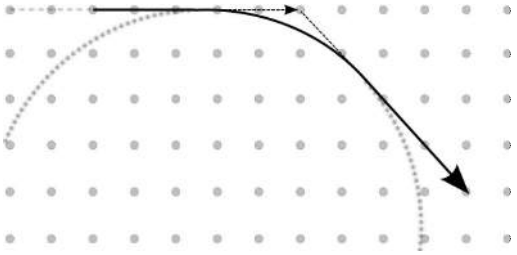
The OSN problem requires finding a deterministic shortest path on the graph G', which can be computed by the well-known (heap implementation of) Dijkstra's algorithm. When it can be used, the A^* algorithm [4, 5, 38] is an alternative to Dijkstra's algorithm, which has the same worst-case complexity, 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 h it uses is *admissible*, i.e., never overestimates the actual shortest distance, then A^* indeed terminates with an optimal solution [38].

In the setting of the graph G', which is embedded in the plane, an admissible heuristic function is the time-length of the line segment from every vertex to t with the non-turn navigation speed of k knots ("as the crow flies;" in the absence of any and all obstacles). Moreover, this heuristic function is *valid* in the sense that if $h(t) = 0$ and, for all $(u, v) \in \mathsf{E}'$, $h(u) \leq h(v) + \varphi(u, v)$. This latter inequality can be seen as a triangle inequality. Note that if h is valid, then for all $v \in \mathsf{V}'$, $h(v)$ turns out to be a lower bound on the shortest v, t path distance. In this case, the A^* algorithm can be coded much more simply as re-labeling of vertices is never necessary.

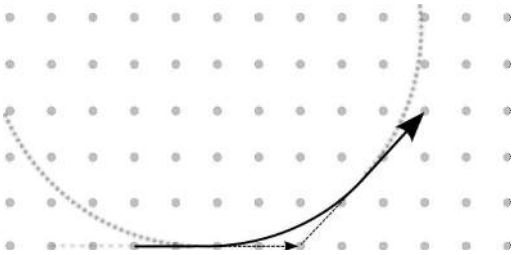
1.4.5 Smoothing the Optimal Path

Due to the nature of our lattice discretization, any (45-degree) right or left turn on the lattice looks artificial and is infeasible in reality as no ship can manoeuver this path exactly. Thus, once the optimal (discrete) path is determined, any such turn needs to be smoothed such that the smoothed paths are consistent with the ship's real-world turn dynamics. This, in turn, allows for emulating the actual (continuous-space) navigation of the ship. There has been some research on path smoothing using post-processing techniques such as curve fitting and creating the smoothed path directly [39, 40]. In this work, we employ a more natural and realistic-looking smoothing method that takes advantage of the special structure of the underlying problem. Specifically, we smoothen

the turns by replacing the inner halves of the turn legs with the arc segment of the turn circle. This process is illustrated in Figure 1.6 for right and left turns respectively.



(a) Right turn



(b) Left turn

FIGURE 1.6: Smoothing of right and left turns.

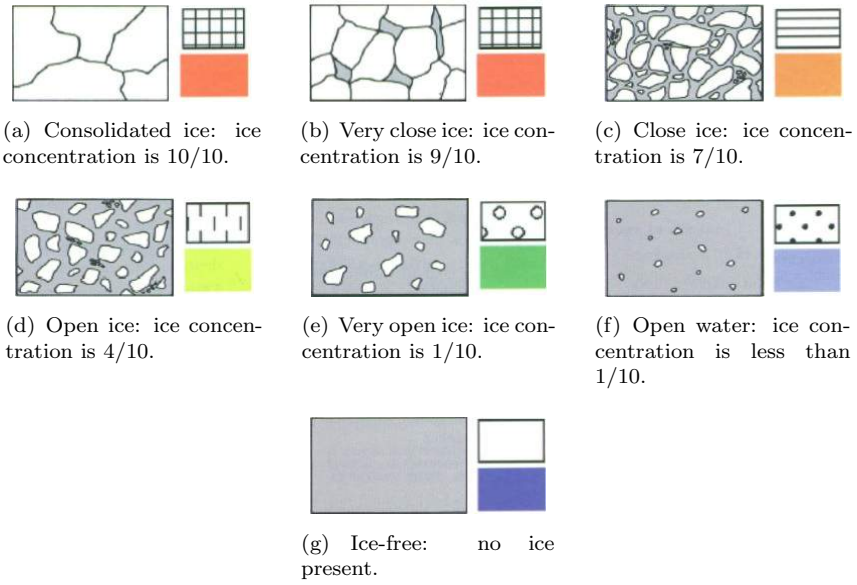
1.5 Ice Navigation Example

Sea ice covers about 7% of world's oceans [41]. Of particular interest is the sea ice in the Arctic region as recent studies reveal that the Arctic sea ice is decreasing at a much faster rate than previously forecasted [42]. This phenomena has been primarily attributed to two reasons: (1) global warming (the rate of warming at the Arctic region is twice the globally-averaged), and (2) feedback of the atmospheric circulation and oceanic circulation change [43].

Northern Sea Route (NSR) through the Arctic region links the Atlantic and Pacific oceans and allows for maritime transport between Europe, North America, and Asia. What makes NSR critical for world's international trade is that it provides a route between Europe and Asia that is 9,000 km shorter than the Panama Canal route and 17,000 km shorter than traveling around Cape Horn, South America [44]. Yet, usage of NSR has been limited so far primarily due to the fact that it has never been ice-free, even during the summer months [42]. However, with the rapid melting of the Arctic sea ice, NSR could soon open to intercontinental shipping. That being the case, ship navigation in waters with partially-melted ice poses significant safety risks. In fact, Ho [42] states that *"Before the Arctic routes can reliably be used on a large scale for transit by shipping along its passages, more investments are required on infrastructure and the provision of marine services to ensure the safe and secure transit of shipping".*

In this section, we provide an ice navigation example that illustrates how our methodology can be applied to optimal ship navigation in ice-covered waters. Our goal is to lay groundwork for further studies on this topic as it is posed to become a new research area with the opening of the NSR for international seaborne transportation. It is highly likely that a methodology such as ours for finding shorter routes in icy waters will not only result in considerable fuel savings and decreased capital costs, but also reduce ships' negative environmental impact on the delicate Arctic ecology.

Sea ice categorization is typically based on the percentage of the ocean surface it covers. International standards identify seven categories of sea ice: (1) ice-free, (2) open water, (3) very open ice, (4) open ice, (5) close ice, (6) very close ice, and (7) consolidated ice [45]. These categories are illustrated in Figure 1.7(adapted from [45]. Our example is based on an open ice navigation area, which roughly corresponds to an ice concentration of 4/10.



(a) Consolidated ice: ice concentration is 10/10.

(b) Very close ice: ice concentration is 9/10.

(c) Close ice: ice concentration is 7/10.

(d) Open ice: ice concentration is 4/10.

(e) Very open ice: ice concentration is 1/10.

(f) Open water: ice concentration is less than 1/10.

(g) Ice-free: no ice present.

FIGURE 1.7: Illustration of ice categories.

In our example, we continue to use the previous setup where the navigation area A is a 12 by 8 miles rectangular region. Resolution factor f is taken as 20, which implies that $x_{\max} = 240$ and $y_{\max} = 160$ in the integer lattice discretization. For each ice block, we set the buffer zone safety distance d_p to 150 meters. In order to simulate open ice formation, we construct random Voronoi tiles [46] as follows: We first sample 100 points from a uniform distribution in the range [1,240]x[1,160]. Then we compute the Voronoi tiles for these points. Next, we manually designate roughly 40 of these tiles as ice blocks and the remaining ones as ice-free regions so that our navigation area resembles open ice. We leave it future research to devise an automated methodology for generation of random ice fields. Figure 1.8(a) depicts the ice formation and the navigation area used in our example. Figure 1.8(b) shows the buffer zones around each ice block and the

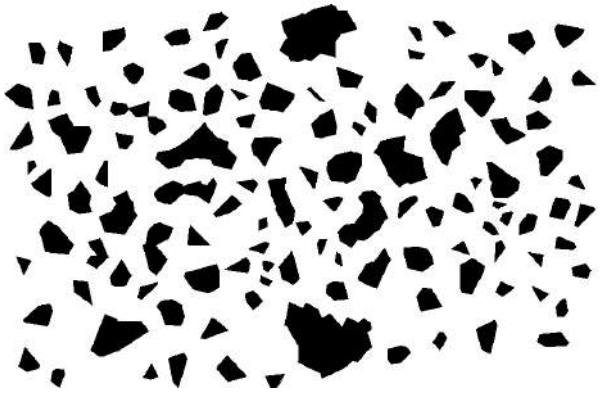
shortest path with one-edge ahead symmetric 45-degree turn constraints. Figure 1.8(c) illustrates the smoothed shortest path with ship-turn constraints. In this particular example, the shortest path with the ship-turn constraints differs dramatically from the one with simple one-edge ahead turn constraints—illustrating unsuitability of existing approaches for ship navigation.

1.6 Simulator Application

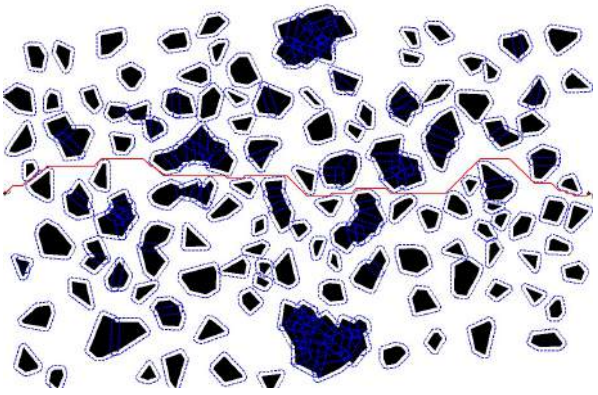
Similar to aircrafts, automated navigation systems are available for merchant ships (Tokyo Keiki Inc. (`http://www.tokyokeiki-usa.com/categories/view/6`) and Yokogawa autopilots (`http://www.yokogawa.com/ydk/mr/marine/pilot/index.htm`) are two examples of such systems). However, these autopilot systems are designed primarily for navigation in open waters. In restricted waterways or in the presence of close-by obstacles, the common practice is to switch to hand-steering mode due to safety concerns and limited maneuvering capability. Therefore, the optimal path obtained above for our ice navigation example needs to be traversed manually by a qualified helmsman in a real-world application. In this section, we present a proof-of-concept by having an experienced oceangoing captain hand-steer the graph-theoretic path and thereby simulate the ship's actual path in a full-mission ship handling simulator (FMSHS). Our purpose with this simulation is to illustrate real-world feasibility of our graph-theoretical methodology and gain insight into any limitations it might have.

FMSHSs are utilized throughout the world not only for educational and training activities, but also for fulfilling research and development objectives of maritime industry. Some examples of FMSHS research activities are as follows: analysis of environmental and maneuvering difficulties in new port constructions, port approaching, real-time navigation in the presence of obstacles, and berthing/unberthing maneuvers. The FMSHS we use is Japanese Marine Science (JMS) branded and equipped with sophisticated instruments same as a real ship bridge. As a full mission simulator, it is capable of fully simulating behavioral and physical aspects of a bridge operation as well as performing advance maneuvers in restricted waterways. The simulator system consists of two independent bridges called the main bridge (shown in Figure 1.9) and the secondary bridge. The navigation instruments on the main bridge are connected to a computer system. The secondary bridge is used primarily for radar navigation purposes. The computer-generated navigation imagery is projected to a large oval screen by seven CRT projectors with a 240-degree viewing angle from port wing to starboard wing.

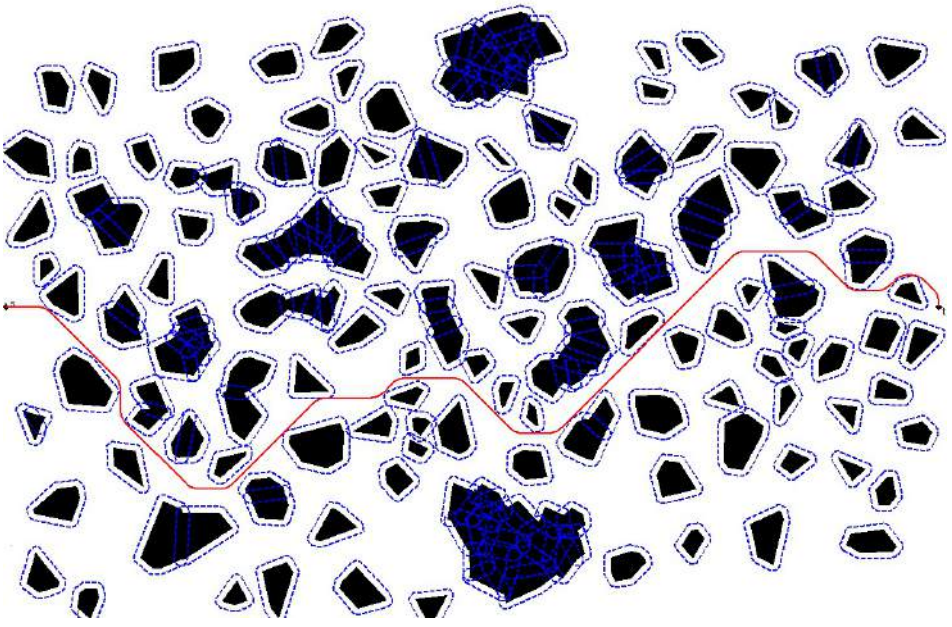
Figure 1.10 shows the manually navigated path inside the FMSHS labeled with the ship's instantaneous speed, and Figure 1.11 compares the manually navigated path to our graph-theoretical shortest path. We observe that the manual path closely (but not exactly) follows the graph-theoretical path while avoiding buffer zones at all times.



(a) Navigation area and open ice formation



(b) Buffer zones around each ice block and the shortest path with symmetric one-edge ahead 45-degree turn constraints



(c) The shortest path with ship-turn constraints and smoothing

FIGURE 1.8: Ice navigation example.

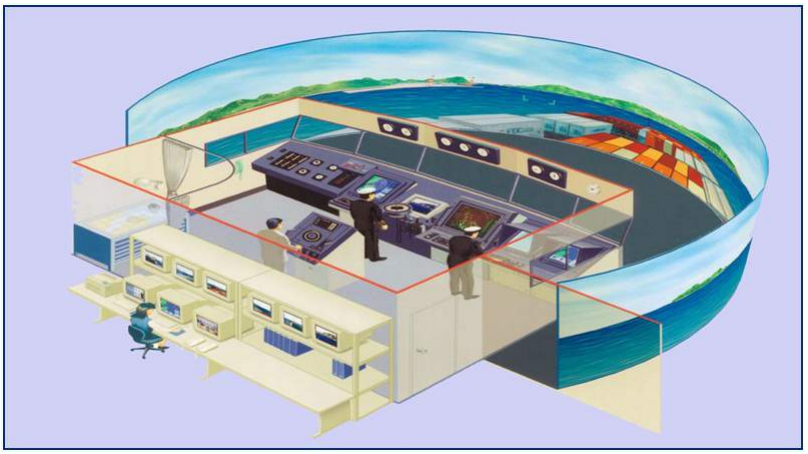


FIGURE 1.9: Illustration of the JMS full-mission ship handling simulator main bridge (source: JMS manual).

Such a slight difference is somewhat to be expected considering the human element in the process, but the crucial observation here is that, at least in our example, the graph-theoretical path seems to be consistent with the complex navigation and turn dynamics of the merchant ship under consideration. However, we do notice a limitation of our model: we observe that the ship's speed is a continuous quantity between about 8 and 10 knots as opposed to being exactly 8 knots while turning and 10 knots while straight navigation. Such a discrepancy in navigation speeds might perhaps become an issue in mission-critical applications, but its impact is likely to be less dramatic in a commercial seaborne shipping setting.

1.7 Summary, Conclusions, and Future Research

This research is concerned with a graph-theoretical approach to the optimal ship navigation problem wherein the objective is to find the shortest path between two given coordinates in a lattice-discretized navigation area in the presence of obstacles subject to safety distance and ship-turn constraints. The latter constraint consists of the following: (1) the ship's left and right turn radii are different, (2) ship's speed reduces while turning, and (3) the ship needs to navigate a certain minimum number of lattice edges before making any turns—so that the navigation area can be discretized at any desired resolution. We present a geometry-based algorithm for computing safety buffer zone polygons corresponding to each obstacle. In order to facilitate implementation of the ship-turn constraints, we define a new graph where the lattice vertices are split into 24 copies that incorporate immediate navigation history, that is, direction and edge type information. In particular, the type of an edge specifies whether the ship was performing a straight (non-turn) navigation, or, making a right or left turn before arriving at the edge's end vertex. For each one of these vertex copies, we carefully define outgoing edges that represent all legal traversals while avoiding any illegal moves. Once the optimal

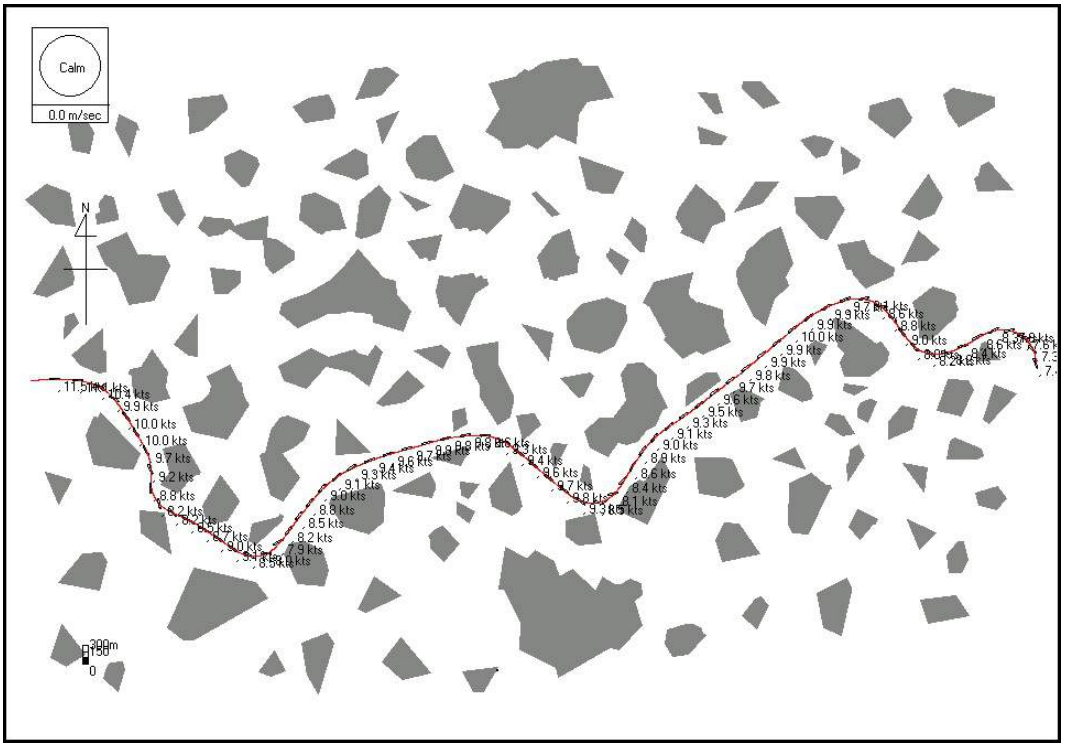


FIGURE 1.10: Manually navigated path inside the full-mission ship handling simulator. The path is labeled by the ship's instantaneous speed.

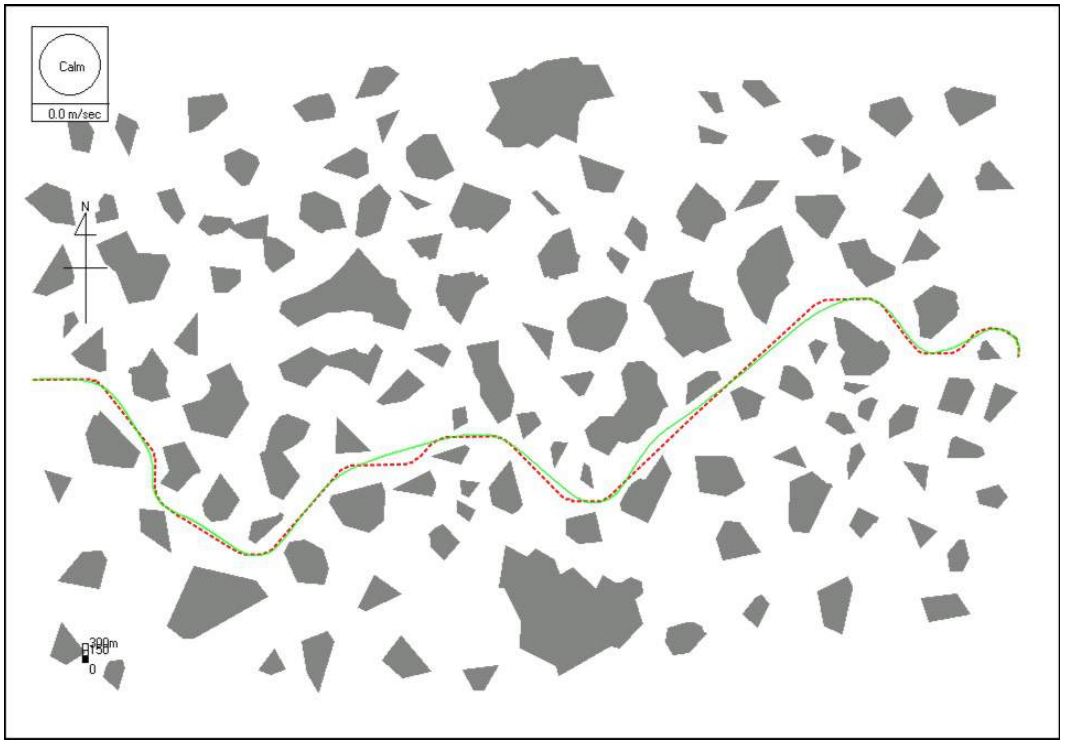


FIGURE 1.11: Comparison of the graph-theoretical shortest path (dashed line) and the manually navigated path (solid line).

(discrete) path is determined, we smoothen it using geometry to emulate the actual (continuous) navigation of the ship.

The field application we present to illustrate our methodology is a 100,000 DWT merchant ship navigation in ice-covered waters. We simulate the actual navigation of the ship in a full-mission ship handling simulator to demonstrate real-world feasibility of our approach. Finding shorter routes in icy waters using a methodology such as ours is likely to result in not only fuel and capital savings, but also reduce ships' negative impact on the environment.

In what follows, we propose several directions for future research. In this work, we ignore environmental effects such as winds, waves, or sea currents. Such conditions, on the other hand, often play a significant role in ship navigation. We plan to incorporate such environmental constraints in our future research. In addition, we assume that the obstacles inside the navigation area are static, that is, they do not move or change shape as the ship navigates. Even though certain obstacles fit this assumption (such as islands or rock formations), it is likely the case that obstacles such as ice blocks or other ships change location over the course of the ship's voyage. We plan to investigate how our methodology can be adapted to handle such a scenario. Moreover, even though the shortest path we find is guaranteed to be optimal (in our discrete setting), it is limited to 45-degree turns on the integer lattice. There exist several studies in the literature that propose algorithms for finding feasible paths with arbitrary turn angles without constraining the traversals to grid edges—though without any optimality guarantees (see, e.g., [47]). We intend to investigate how such approaches can be adapted for ship navigation in the presence of environmental and ship-turn constraints.

Chapter 2

Algorithms for Stochastic Obstacle Scenes

2.1 Introduction

We consider a probabilistic path planning problem wherein an agent needs to quickly navigate from one given point to another through an arrangement of arbitrarily-shaped regions which are possibly obstacles. At the outset, the agent is given the respective probabilities that the regions are truly obstacles. These probabilities are referred to as the *region's mark*. When situated on a region's boundary, the agent has the option to disambiguate it, i.e., learn at a cost if it is truly an obstacle. The central question is to find an algorithm 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 Obstacle Scene Problem* (SOSP), which is a minor modification of the problem as introduced in [48]. Also described in that work is a graph-theoretic analog of this problem, which the authors call the *Canadian Traveler's Problem* (CTP). In CTP, the goal is to find the minimum expected length path over a finite graph whose edges are marked with their respective probabilities of being traversable, and each edge's status can be discovered dynamically when encountered. SOSP and CTP have practical applications in important probabilistic path planning environments such as robot navigation in stochastic domains ([49–51]), minefield countermeasures ([52, 53]), and adaptive traffic routing ([54, 55]). In fact, both problems as well as closely related ones have gained considerable attention recently—see, e.g., Nikolova and Karger [56], Eyerich et al. [57], Likhachev and Stentz [58], Xu et al. [59], Aksakalli and Ceyhan [60].

There are no efficiently computable optimal policies known for SOSP or CTP and many similar problems have been shown to be intractable [48, 61]. The fundamental difficulty in obtaining a tractable model, even in the discrete setting, is that in order for the agent to consider any action at any location, it needs to take into account what it has

learned about the status of all of the potential obstacles. Thus, exponentially many such possibilities need to be incorporated when constructing the state space. The reader is referred to [62] and the references therein for a review of the literature that includes the history and development of the problems that fall under the SOSP and CTP umbrella.

Regarding suboptimal algorithms for continuous SOSP, of particular interest are the simulated risk disambiguation algorithm (SRA) of [63] and the reset disambiguation algorithm (RDA) of [62]. The idea behind SRA is to temporarily pretend, i.e., simulate, that the ambiguous obstacles are riskily traversable for the sole purpose of deciding where to disambiguate next. RDA, on the other hand, is an efficient algorithm for the SOSP that is provably optimal for a restricted class of SOSP, and it has been shown to perform relatively well for general instances of the problem.

The contributions of this chapter are as follows:

1. Even though discrete SOSP (i.e., CTP) is intractable in general, we present a polynomial-time algorithm when the associated graph is restricted to parallel graphs. This presentation has two purposes: first, it illustrates the difficulty of discrete SOSP even in extremely simple settings, and second, it shows an alternate interpretation of the reset disambiguation algorithm.

2. We show how the simulated risk and reset disambiguation algorithms for continuous SOSP can be adapted to the discrete and lattice-discretized versions.

3. We propose a generalized framework encompassing the simulated risk and reset disambiguation algorithms that uses penalty functions to guide the agent's navigation in realtime. Within this framework, we introduce a new algorithm where the navigation is guided by taking into account the distance from the current location to the termination point in addition to the disambiguation cost and true-obstacle probabilities of risk regions. We call this the DT Algorithm (DTA) where DT stands for "distance to termination". We present computational experiments that involve synthetic data as well as an actual naval minefield data set in order to illustrate our algorithms. Our experiments indicate that DTA provides near-optimal results with minimal computational resources.

Our presentation of the algorithms involve disk-shaped regions and the discretization of the continuous setting is done on an integer lattice. It should be noted that these algorithms can easily be modified for regions with different shapes as well as for different discretization techniques. In fact, the algorithms can be generalized for discrete SOS problems on arbitrary graphs in a relatively straightforward manner.

The rest of this chapter is organized as follows: Section 2.2 presents two simple SOSP examples and compares their continuous and discrete versions. Section 2.3 formally defines the continuous, discrete, and (lattice) discretized SOSP. Section 2.4 presents

a polynomial-time exact method for computing the optimal solution for discrete SOSP when the associated graph is restricted to parallel avenues and fixed policies exist within the avenues. Sections 2.5 and 2.6 review SRA and RDA respectively and present their adaptations to discrete and discretized SOSP. Section 2.7 generalizes these two algorithms as penalty-based navigation strategies and introduces the DT algorithm. Section 2.8 presents computational experiments that compare the performance of DTA against SRA and RDA. Summary and conclusions are presented in Section 2.9.

2.2 The Stochastic Obstacle Scene Problem: Continuous vs. Discrete Settings

The SOS problem is inherently a continuous-space problem. Specifically, in an appropriate terrain on land or in sea, an agent can navigate along arc segments associated with the possibly-obstacle disks. However, a major challenge in the continuous version of the problem is to decide where exactly a disk needs to be disambiguated to achieve the shortest expected length. In fact, Section 2.2.1 below illustrates that in a simple case with only one disk, the optimal disambiguation point is a function of the disk's mark and its computation requires finding the root of a rather complex nonlinear equation. Furthermore, Section 2.2.2 illustrates via an example with two disks that the optimal disambiguation point of a particular disk does not only depend on this disk's mark, but also on the location and mark of the other disks present in the obstacle field. Thus, optimal disambiguation points are not readily computable for all but the most trivial instances of continuous SOSP.

2.2.1 Deciding Where to Disambiguate: Single Disk Case

Consider an instance of continuous-space SOSP with only one disk, as shown in Figure 2.1. In this instance, the starting point is $S = (0, 0)$ and the termination point is $T = (4, 0)$. The disk is centered at $(2, 0)$ with a radius of 1. The cost of disambiguation is taken as zero. If the mark associated with this disk is $\rho = 0$, then the optimal disambiguation point is $C = (1, 0)$. Consequently, a disambiguation algorithm that dictates disambiguating at C would traverse S, C, D, T with a total length of 4 units.

On the other hand, if $\rho = 1$, then, the optimal disambiguation point is intersection point of the tangent line from S to the disk, which is denoted by A. The coordinates of A can be computed using geometry as follows: the secant line SCE is related to the tangent line SA by $|SA|^2 = |SC||CE|$. Thus, we get $|SA| = \sqrt{3}$. Since $\widehat{SAE}$ is a right angle, we have $|SA||AE| = |SE||AH|$, which yields $|AH| = \frac{\sqrt{3}}{2}$. Furthermore, since SHA is a $30 - 60 - 90^o$ triangle, we have $|SH| = \frac{1}{2}$. Thus, $A = (\frac{1}{2}, \frac{\sqrt{3}}{2}) \approx (1.5, 0.866)$.

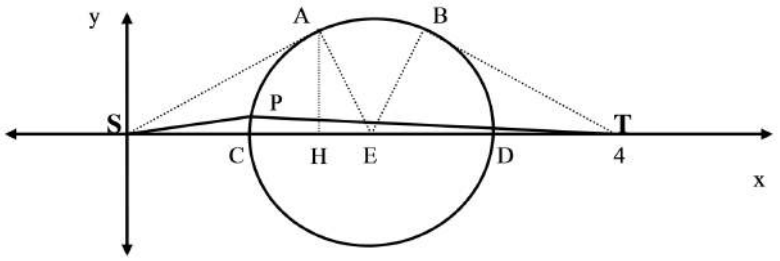


FIGURE 2.1: A continuous SOSP instance with a single disk.

The optimal disambiguation point for an arbitrary $\rho \in (0,1)$ can be computed using geometry and some algebra as follows: Let $E(P)$ be the expected length of the traversal when the disk is disambiguated at point $P = (u, v)$. Thus, the task is to determine the optimal disambiguation point P^* where

$$P^* = (u^*, v^*) = \arg \min_{\substack{u \in (1,1.5) \\ v = \sqrt{1-(u-2)^2}}} E(P = (u, v))$$

First, note that $\widehat{CEA} = \widehat{BED} = 60^o$, so $\widehat{AEB} = 60^o$ and therefore $|arcAB| = \frac{\pi}{3}$. Moreover, $|BT| = |SA| = \sqrt{3}$, which yields $|arcAB| + |BT| = \frac{\pi}{3} + \sqrt{3} \approx 2.7792$. Thus, for $\rho \in (0,1)$, for any point P on the arc segment $arcCA$, we have

$$E(P) = |SP| + (1-\rho)|PT| + (\rho)(|arcPA| + |arcAB| + |BT|)$$

$$= |SP| + (1-\rho)|PT| + (\rho)(|arcPA| + 2.7792)$$

$$= \sqrt{u^2+v^2} + (1-\rho)(\sqrt{(4-u)^2+v^2}) + (\rho)\Big(2.7792 + (\frac{\pi}{3} - \arctan(\frac{v}{2-u}))\Big) \quad (2.1)$$

Substituting $v = \sqrt{1-(u-2)^2}$, we get

$$= \sqrt{u^2+1-(u-2)^2} + (1-\rho)\big(\sqrt{(4-u)^2+1-(u-2)^2}\big)$$

$$+(\rho)\left(2.7792 + \left(\frac{\pi}{3} - \arctan\left(\frac{\sqrt{1-(u-2)^2}}{2-u}\right)\right)\right)$$

$$= \sqrt{4u-3} + (1-\rho)\big(\sqrt{13-4u}\big) + (\rho)\left(2.7792 + \left(\frac{\pi}{3} - \arctan\left(\frac{\sqrt{-u^2+4u-3}}{2-u}\right)\right)\right)$$

Thus, given a specific $\rho \in (0,1)$, $E(P)$ is a function of a single variable, u, which we denote by $E(u)$. Observe that $E(u)$ is a convex function, which indicates its unique minimum can be found by setting the following derivative to zero:

$$\frac{d}{du}\Big(E(u)\Big) = \frac{2}{\sqrt{4u-3}} + (1-\rho)\big(\frac{-2}{\sqrt{13-4u}}\big) - (\rho)\left(\frac{\frac{(-u+2)}{\sqrt{-u^2+4u-3}}(2-u) + \sqrt{-u^2+4u-3}}{\Big(1+\frac{-u^2+4u-3}{(2-u)^2}\Big)(2-u)^2}\right)$$

The last term can be simplified as $(\rho)\Big(\frac{1}{\sqrt{(u-1)(3-u)}}\Big)$, yielding $\frac{d}{du}\Big(E(p(u))\Big)$ as

$$\begin{aligned}
&= \frac{2}{\sqrt{4u-3}} + (1-\rho)\big(\frac{-2}{\sqrt{13-4u}}\big) - (\rho)\big(\frac{1}{\sqrt{(u-1)(3-u)}}\big) \\
&= \Big[\sqrt{(4u-3)(13-4u)(u-1)(3-u)}\Big]^{-1}\Big[2\sqrt{4u-3}(13-4u)(u-1)(3-u) \\
&\quad -2(1-\rho)\sqrt{13-4u}(4u-3)(u-1)(3-u) - \rho\sqrt{(u-1)(3-u)}(4u-3)(13-4u)\Big]
\end{aligned}$$

Thus we have the following result: For $\rho \in (0,1)$, the optimal disambiguation point is $P^* = (u^*, \sqrt{1-(u^*-2)^2})$ where u^* is the unique solution of the following equation in the interval $(1, 1.5)$:

$$\begin{aligned}
0 &= 2\sqrt{4u-3}(13-4u)(u-1)(3-u) - 2(1-\rho)\sqrt{13-4u}(4u-3)(u-1)(3-u) \\
&\quad -\rho\sqrt{(u-1)(3-u)}(4u-3)(13-4u) \qquad (2.2)
\end{aligned}$$

Using MATLAB, we tabulated ρ versus (u^*, v^*) for several different values of ρ in Table 2.1. It can be seen that the closer ρ is to 1, the closer the optimal disambiguation point is to $A = (1.5, .87)$.

TABLE 2.1: Optimal disambiguation points and corresponding expected lengths for different ρ's

ρ	(u^*, v^*)	$E(u^*, v^*)$
0	(1, 0)	4
.5	(1.06, .34)	4.33
.75	(1.15, .53)	4.44
.9	(1.27, .68)	4.49
1	(1.5, .87)	4.51

2.2.2 Deciding Where to Disambiguate: Two Disks Case

In this section, we illustrate the fact that the optimal disambiguation point of a particular disk does not just depend on this disk's mark, but also on the location and mark of the other disks present in the obstacle field.

Consider the SOSP instance with two disks in Figure 2.2. In this instance, the starting point is $S = (0, 0)$ and the termination point is $T = (8, 0)$. The first disk is centered

at $(2,0)$ and the second at $(6,0)$, both with a radius of 1. The cost of disambiguation is taken as zero. Let x_1, x_2 be the first and second disks, respectively, and ρ_1, ρ_2 be the marks of these disks. Furthermore, let $W_{1,2}$ denote the walk associated with the algorithm that calls for first disambiguating x_1 and then x_2 regardless of the outcome of the disambiguation. Now, let $P^* = (u_1^*, v_1^*)$ and $Q^* = (u_2^*, v_2^*)$ be the optimal disambiguation points associated with $W_{1,2}$. If $\rho_1 = 0, \rho_2 = 0$, then $P^* = C = (1,0)$ and $Q^* = G = (5,0)$. However, if $\rho_1 = 0, \rho_2 = 1$, then $P^* = F$ and $Q^* = H$. Thus, the optimal disambiguation point for x_1 associated with the policy $W_{1,2}$ depends on the location and mark of x_2.

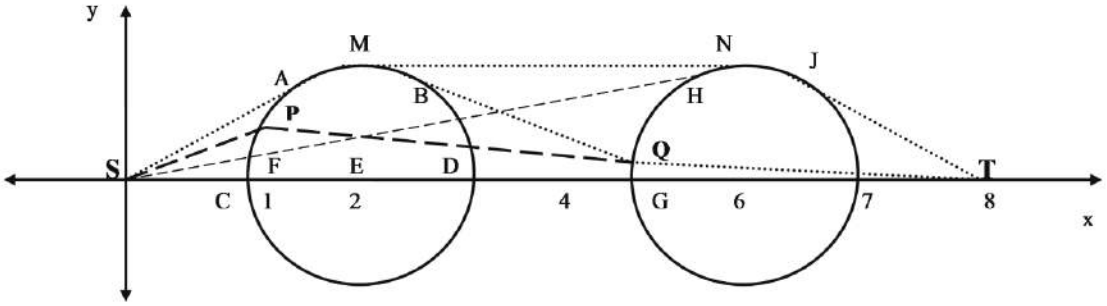


FIGURE 2.2: A continuous SOSP instance with two disks.

In fact, we can compute the optimal disambiguation point P^* associated with $W_{1,2}$ as follows:

$$\begin{aligned} E(W_{1,2}(P,Q)) &= |SP| + (1-\rho_1)\big(|PQ| + (1-\rho_2)|QT| + (\rho_2)(|arcQHJ| + |JT|)\big) \\ &\quad +(\rho_1)\big(|arcPAB| + |BQ| + (1-\rho_2)|QT| + (\rho_2)(|arcQHJ| + |JT|)\big) \end{aligned}$$

Due to the fact that P and Q are points on x_1 and x_2 respectively, it holds that $v_1 = \sqrt{1-(u_1-2)^2}$ and $v_2 = \sqrt{1-(u_2-6)^2}$. For this reason, $E(p_{1,2}(P,Q))$ can be expressed as a function of u_1 and u_2 similar to equation (2.1). One can then compute partial derivatives of $E(W_{1,2}(u_1,u_2))$ with respect to u_1 and u_2 and determine the roots in the interval (1,1.5) for u_1 and (5,6) for u_2 to obtain P^* and Q^*. That is,

$$(u_1^*, u_2^*) = \arg \min_{\substack{u_1 \in (1,1.5) \\ u_2 \in (5,6)}} E(W_{1,2}(u_1,u_2)) \tag{2.3}$$

Note that Q^* is computed in (2.3) as it is needed to determine P^* in an expected sense. Once x_1 is disambiguated, a new Q^* needs to be re-calculated as in the single disk case based upon the actual outcome of the disambiguation. It should also be noted that the solution of (2.3) involves nontrivial geometric calculations and solving a highly nonlinear system with two equations in two unknowns, namely, u_1^* and u_2^*.

2.2.3 Discretization of the Continuous Setting: An Example

Given the challenges associated with the continuous version of SOSP, we consider a lattice discretization of the problem for convenience and ease of computation. As an illustration, a lattice discretization of the SOSP instance in Figure 2.2 is shown in Figure 2.3 where one unit distance is represented by three non-diagonal lattice edges. In the figure, edges intersecting the disks are shown in bold. The endpoints of these edges that are outside of the disks are designated as the disambiguation points of the corresponding disk. A desirable feature of the lattice discretization is that its resolution can be increased or decreased as needed to achieve a desired balance between accuracy and computational burden.

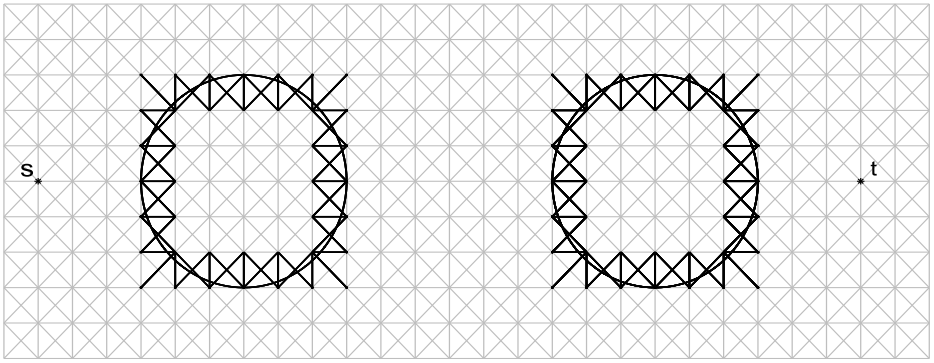


FIGURE 2.3: A lattice discretization of the SOSP instance in Figure 2.2. Edges intersecting the disks are shown in bold.

Even in the lattice-discretized version of the problem, finding an algorithm to minimize the total expected traversal length is a challenging task. This difficulty arises from the fact that in order for the agent to decide its action at any given location, it needs to take into account what it has learned about the status of all of the potential obstacles (true, false, or ambiguous respectively), and exponentially many such possibilities need to be incorporated into the agent's decision.

2.3 Definition of the Stochastic Obstacle Scene Problem

This section formally defines the continuous, discrete, and lattice-discretized SOS problems respectively.

2.3.1 Continuous SOSP

Without loss of generality, we shall consider SOS problems with disk-shaped possible-obstacles. We formally define this problem as follows: Consider a marked point process on a particular region R in $\mathbb{R}^2$—this region shall be called the *obstacle field*. This process generates random detections $X_T, X_F \subseteq R$ (respectively called *true* and *false* detections), and random marks $\rho_T : X_T \to [0, 1)$ and $\rho_F : X_F \to (0, 1]$. When observing a realization of this process, the agent only sees $X := X_T \bigcup X_F$ and $\rho := \rho_T \bigcup \rho_F$. We assume that, for all $x \in X$, $\rho(x)$ is the probability that $x \in X_T$. We also 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 D_x is an open disk centered at x with radius $r(x) > 0$, for a given function $r : X \to \mathbb{R}_{>0}$. For any $x \in X$, the probability $\rho(x)$ shall be referred to as the "mark" of the associated disk D_x. That is, mark of a disk is essentially the probability that this disk is a true obstacle and not a false one. Given a starting point $s \in R$ and a destination point $t \in R$, the agent seeks to traverse a continuous s, t curve in $(\bigcup_{x \in X_T} D_x)^C$ of shortest achievable arclength (here, C denotes the set complement operator).

We further suppose that there is a dynamic learning capability. Specifically, for all $x \in X$, when the curve is on the boundary ∂D_x, the agent has the option to *disambiguate* x, that is, learn if $x \in X_T$ or not. For a given cost function $c : X \to \mathbb{R}_{\geq 0}$, it is assumed that such a disambiguation shall result in a cost $c(x)$ being added to the overall length of the curve. We assume that there is a limit K on the number of available disambiguations. How the agent should route the continuous s, t traversal curve—and where and when the disambiguations should be performed—to minimize the expected length of this curve is called the *continuous SOSP*.

2.3.2 Discrete SOSP

The discrete analogue of the above problem, which we call the *discrete SOSP*, is defined as follows: Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be an undirected graph with designated vertices $s, t \in \mathsf{V}$, and suppose there is a function $\ell : \mathsf{E} \to \mathbb{R}_{\geq 0}$ 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 may indeed be traversable. In particular, for a given subset $\mathsf{E}' \subseteq \mathsf{E}$ of edges, called *stochastic edges*, there is a function $\rho : \mathsf{E}' \to [0, 1)$ such that, for each edge $e \in \mathsf{E}'$, $\rho(e)$ is the probability that e is not traversable, independent of the other edges. As in the continuous setting, $\rho(e)$ shall be referred to as the "mark" of the edge e. For clarity of notation, marks of disks in the continuous setting and marks of edges in the discrete setting shall both be denoted by ρ. Edges in $\mathsf{E} \setminus \mathsf{E}'$ are deterministic in the sense that they are known a priori to be traversable. For any edge $e \in \mathsf{E}'$, when the traversal is at an endpoint of e, the agent has 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 $c : \mathsf{E}' \to \mathbb{R}_{\geq 0}$. 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 agent 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 agent's goal, however, is to find an optimal algorithm in the sense of having shortest expected length. As in the continuous version, we assume that there is a limit K on the number of available disambiguations. Finding such an optimal algorithm is the discrete SOSP (also known as the Canadian Traveler's Problem (CTP) in the literature). To avoid infinite expected length, we assume the existence of a (possibly very long) s, t path consisting of edges from $\{e \in \mathsf{E}' : \rho(e) = 0\} \cup (\mathsf{E} \setminus \mathsf{E}')$.

2.3.3 Discretized SOSP

As mentioned earlier, optimal disambiguation algorithms are not readily computable for all but the most trivial instances of continuous SOSP. 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 the following four types of vertices: (1) (i, j) and $(i+1, j)$ with unit length, (2) (i, j) and $(i, j+1)$ with unit length, (3) (i, j) and $(i+1, j+1)$ with length $\sqrt{2}$, and, (4) $(i+1, j)$ and $(i, j+1)$ with length $\sqrt{2}$. One vertex in G is designated as the starting point s, another vertex in G is designated as the termination point t. The agent is to traverse from s to t in G, only through edges that do not intersect any true or ambiguous obstacles. If an edge intersects any ambiguous obstacle, then a disambiguation 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 (the terms *solution* and *policy* shall be used interchangeably). We call this lattice discretization as *Discretized SOSP*, which, in effect, is a special case of discrete SOSP with statistical dependency among the edges.

2.4 A Polynomial Algorithm for Discrete SOSP on Parallel Graphs

The discrete SOS problem has been shown to be NP-hard [61]. In this section, however, we present a polynomial algorithm when the problem is restricted to parallel graphs.

We call a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ *parallel* if $\mathsf{V} = \{s, t\}$, and all edges in E have both s and t as endpoints. Without loss of generality, the policies that need to be considered in

this case consist of an ordering on 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. The remark below gives a polynomial-time method for discrete SOSP on parallel graphs with $K = \infty$. An efficient algorithm for the problem when K is finite can be found in Blatz et al. [64].

Remark 1. Discrete SOSP on parallel graphs can be solved in $\mathrm{O}(|\mathsf{E}| \log |\mathsf{E}|)$ as opposed to the brute-force approach in $\mathrm{O}(|\mathsf{E}|!)$. Specifically, the policy that orders the edges by

$$h(e) := \ell(e) + \frac{c(e)}{1 - \rho(e)} \tag{2.4}$$

for all $e \in \mathsf{E}$ is optimal.

2.5 Discrete Adaptation of the Simulated Risk Disambiguation Algorithm

This section adapts the simulated risk disambiguation algorithm (SRA) in [63] introduced for continuous SOSP to discrete and lattice-discretized SOSP (an earlier version of this section's research appeared in [65]).

2.5.1 Adaptation to Discrete SOSP

In our framework, the traversal never uses edges while they are still ambiguous or are known to be non-traversable. The key intuition behind SRA is—for the sole purpose of deciding where to disambiguate next—to temporarily pretend (*simulate*) that the ambiguous edges are riskily traversable.

Under this simulation of risk, for any s, t walk W, its *risk length* is defined as

$$\ell^r(W) := -\log \prod_{e \in (W \cap \mathsf{E}')} (1 - \rho(e)).$$

This negative logarithm of the probability that W is permissibly traversable is a measure of the risk in traversing W—if the agent were willing to take on risk. Note that the agent might revisit a vertex over the course of the traversal, making the final trajectory a walk (and not a path).

An *undesirability function* is any function $g : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ which is monotonically nondecreasing in its arguments; that is to say, for all $r_1, r_2, z_1, z_2 \in \mathbb{R}_{\geq 0}$ such that $r_1 \leq r_2$ and $z_1 \leq z_2$, it holds that $g(r_1, z_1) \leq g(r_2, z_2)$. The number $g(\ell^e(W), \ell^r(W))$ is thought of as a measure of the undesirability of W in the sense that, if the agent were required

to traverse from s to t in G under the simulation of risk and without a disambiguation capability, the agent would select the walk

$$\phi_g := \underset{s-t \text{ walks } W}{\arg\min}\, g(\ell^e(W), \ell^r(W)).$$

The simplest undesirability functions are the linear ones where $g(r, z) := r + \alpha \cdot z$ for some given constant $\alpha > 0$, and it is these undesirability functions that we restrict our attention. To find ϕ_g in this particular case, we just need to find a deterministic shortest $s - t$ path in G via e.g, Dijkstra's algorithm where each edge in E is weighted as follows:

$$w_D^{SRA}(e) := \ell^e(e) + \mathbf{1}_{e \in \mathsf{E}'} \cdot \alpha \log(1 - \rho(e))^{-1} \tag{2.5}$$

where $\ell^e(e)$ is the edge's Euclidean length (which is either 1 or $\sqrt{2}$), and $\mathbf{1}$ is the indicator function (taking value 1 or 0 depending on whether its subscripted expression is true or false). The (adapted) SRA for discrete SOSP associated with the linear undesirability function $g(r, z) = r + \alpha \cdot z$ would have the agent do the following:

1. Find the shortest s, t path in G with respect to the edge weights w_D^{SRA}. Start from s and traverse this walk until its first ambiguous edge e is encountered at vertex v.

2. At this point (since the agent cannot traverse an ambiguous edge) disambiguate e.

3. If e was just discovered to be traversable, remove it from E'. If e was discovered to be non-traversable, set $\rho(e) := 1$.

4. Repeat this procedure using v as the new s until t is reached or there are no more disambiguations left, in which case the shortest unambiguously permissible path to t is taken.

For a fixed $\alpha > 0$, denote by p_α the s, t walk traversed under SRA. Observe that p_α is an s, t-walk-valued random variable, since its realization depends on the outcomes of the dictated disambiguations. We will denote by $\mathrm{E}p_\alpha$ the expected length of this walk. In our implementation, the values of α minimizing $\mathrm{E}\ell^e p_\alpha$ are computed numerically by evaluating $\mathrm{E}\ell^e p_\alpha$ for a mesh of α values—starting at $\alpha_{min} = 2$ and incrementing successively by $\alpha_{mesh} = 5$ units until α is large enough that no disambiguations are performed.

2.5.2 Adaptation to Discretized SOSP

We now show how SRA can be adapted to discretized SOSP. Again, under simulation of risk, for any s, t walk W, its risk length is defined as

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

Using a linear undesirability function in the form of $g(r, z) := r + \alpha \cdot z$ for some given constant $\alpha > 0$, we need to find a deterministic shortest s, t path in G where each edge in E is weighted as follows:

$$w_{LD}^{SRA}(e) := \ell^c e(e) + \frac{1}{2} \sum_{i=1}^{|X|} \#\text{comp}(e \backslash D_i) \cdot \mathbf{1}_{e \cap D_i \neq \emptyset} \cdot \left(\alpha \log(1 - \rho_i)^{-1} \right) \tag{2.6}$$

where $\#\text{comp}(\cdot)$ is the number of connected components of its argument. SRA for discretized SOSP would have the agent do the following:

1. Find the shortest s, t path in G with respect to the edge weights w_{LD}^{SRA}. Start from s and traverse this walk until its first ambiguous edge e is encountered at vertex v, with edge e intersecting disk D_i.
2. At this point (since the agent cannot enter an ambiguous disk) disambiguate D_i.
3. If D_i was just discovered to be a false obstacle, remove disk D_i's center point X_i from X. If D_i was discovered to be true obstacle, set $\rho_i := 1$.
4. Repeat this procedure using v as the new s until t is reached or there are no more disambiguations left, in which case the shortest unambiguously permissible path to t is taken.

Note that the navigation strategies for discrete and discretized SOSP as dictated by SRA share the following characteristic: The agent first finds the shortest $s - t$ path with respect to a certain edge weight function; w_D^{SRA} for discrete SOSP and w_{LD}^{SRA} for discretized SOSP. Next, the agent navigates this path until the first ambiguous edge or disk is encountered. At this point, a disambiguation is performed. Based on the outcome of the disambiguation, either the edge or disk is removed from the set of stochastic edges or possible-obstacles, or, its mark is set to 1. This procedure is repeated using the current vertex as the new s until t is reached. We call this the *NDR navigation strategy* where NDR stands for "navigate-disambiguate-repeat".

2.6 Discrete Adaptation of the Reset Disambiguation Algorithm

The reset disambiguation algorithm (RDA) introduced in Aksakalli et al. [62] for the continuous SOS problem is provably optimal for a particular variant of the problem, called the *reset variant*. It is also optimal for a restricted class of instances for the original SOSP. Otherwise, the algorithm is generally suboptimal but, it is both effective and efficiently computable. In what follows, we describe the idea behind RDA and present its adaptation to discrete and discretized SOSP respectively.

In discrete SOSP, traversability status of stochastic edges are fixed and they never change until the $s-t$ navigation is completed. In the reset variant, however, each time an edge $e \in \mathsf{E}'$ is disambiguated, its status is governed by independent Bernoulli trials with probability $\rho(e)$. If at a given time a disambiguation determines that e is traversable, then the agent may traverse e immediately, and e remains traversable until the agent reaches the other end point. Otherwise, immediately after each disambiguation of e, the status of e is "reset" and it becomes ambiguous again. Assuming that $K = \infty$, an optimal policy in this reset setting can be determined by the following observation: if an optimal policy dictates at any time that e is disambiguated, and if the disambiguation finds that e is non-traversable, then, by Bellman's Principle of Optimality, the optimal policy will dictate that e be disambiguated again. The reason is that, with the resetting of e, the agent's current state is identical to the agent's state right before the first disambiguation of e. Thus, e must be repeatedly disambiguated until it is traversable. Hence, the number of disambiguations needed is a geometric random variable with expected value $\frac{1}{1-\rho(e)}$. This indicates that under an optimal policy, the agent may view e as if it was deterministically traversable at a cost $\frac{c(e)}{1-\rho(e)}$. This cost is defined to be ∞ if $\rho(e) = 1$ regardless of $c(e)$, and it is in addition to the edge's Euclidean length $\ell^e(e)$. Thus, the optimal policy in the reset variant of discrete SOSP boils down to finding a deterministic $s-t$ path in G where the edge weights are defined as follows:

$$w_D^{RDA}(e) := \ell^e(e) + \mathbf{1}_{e\in\mathsf{E}'} \cdot \frac{c(e)}{1-\rho(e)}. \tag{2.7}$$

The idea in reset disambiguation algorithm is to use the weights w_D^{RDA} (for the reset variant) in exactly the same fashion as SRA (for the original non-reset problem) using the NDR navigation strategy. It is easy to see that adaptation of RDA for discretized SOSP can be achieved by using the weight function below under the NDR navigation strategy:

$$w_{LD}^{RDA}(e) := \ell^e(e) + \frac{1}{2}\sum_{i=1}^{|X|} \#\mathrm{comp}(e\backslash D_i) \cdot \mathbf{1}_{e\cap D_i\neq\emptyset} \cdot \Big(\frac{c(e)}{1-\rho(e)}\Big) \tag{2.8}$$

Per equation (2.7), the reset disambiguation algorithm for discrete SOSP on a parallel graph with only stochastic edges would dictate that the edges are disambiguated in increasing order of $\ell^e(e) + \frac{c(e)}{1-\rho(e)}$. On the other hand, per Theorem 1, this is precisely the optimal policy for the problem! That is, despite the fact that RDA is suboptimal for discrete SOSP in general, it is indeed optimal when the problem is restricted to parallel graphs. This observation essentially indicates that RDA can be interpreted in two different ways: It can either be seen as using the optimal edge weights of the reset variant, or it can be seen as using the optimal edge weights for parallel graphs in the original non-reset version—both within the paradigm of the NDR navigation strategy. It should be noted that either interpretation of the RD algorithm stands as an interesting idea in design of suboptimal algorithms for challenging optimization problems:

- Consider a variant of the original problem for which an efficient optimal algorithm can be computed, and then use this algorithm as a suboptimal algorithm for the original problem, or
- Consider a special case of the original problem for which an efficient optimal algorithm can be computed, and then use this algorithm as a suboptimal algorithm for the original problem.

Even more interestingly, in the case of the RD algorithm for SOSP, both ideas result in exactly the same suboptimal algorithm, and it performs rather well for the original problem.

2.7 Generalizing SRA and RDA: Penalty-Based Algorithms and DTA

The ideas behind the simulated risk and reset disambiguation algorithms for discrete SOSP are fundamentally different: SRA is based on the idea of temporarily pretending that ambiguous edges are riskily traversable. On the other hand, RDA is based on the idea of using the optimal weights of a reset variant in the original non-reset version (or the optimal weights for parallel graphs on arbitrary instances). However, a common feature they share is that both algorithms employ the NDR strategy, though with different weight functions. In this section, we show how this framework can be generalized to allow for different weight functions, hence new algorithms, to potentially improve upon both SRA and RDA as well as address their respective shortcomings as discussed below.

We first observe that the weight functions used by SRA and RDA can be generalized as follows for discrete SOSP using the notion of "penalty functions":

$$w_D^F(e) := \ell^e(e) + \mathbf{1}_{e \in \mathsf{E}'} \cdot F(e), \tag{2.9}$$

and as follows for discretized SOSP:

$$w_{LD}^{F}(e) := \ell^{e}(e) + \frac{1}{2}\sum_{i=1}^{|X|} \#\text{comp}(e \backslash D_i) \cdot \mathbf{1}_{e \cap D_i \neq \emptyset} \cdot F(e). \tag{2.10}$$

In SRA, the penalty function F is specified as $F_{SR}(e) := \alpha \log(1-\rho(e))^{-1}$ whereas it is defined as $F_{RD}(e) := \frac{c(e)}{1-\rho(e)}$ for RDA. For the purpose of generalizing this idea, we define "a penalty-based disambiguation algorithm" as deployment of the NDR navigation strategy with the weight function $w_D^F(e)$ for discrete SOSP and $w_{LD}^F(e)$ for discretized SOSP with an arbitrary (nonnegative) penalty function $F(e)$.

A major downside of SRA is that it needs to "fine-tune" the penalty term via the α parameter for improved performance. The best value of this parameter is essentially found by brute-force. Thus, a clear advange of RDA over SRA is the lack of a fine-tuning parameter that results in significant computational savings. [62] illustrates, via computational experiments, that performance of RDA is comparable to that of SRA whereas run time of SRA is about 60 times greater than that of RDA. Thus, it can be argued that F_{RD} is a "better" penalty function compared to F_{SR}. A reasonable question at this point is if there exist penalty functions even better than F_{RD} in the sense that the NDR navigation strategy with these functions result in shorter expected traversal lengths compared to those obtained by F_{RD}. Of course, F_{SR} and F_{RD} are special, as the first one is motivated by the idea of risk simulation whereas the latter is provably optimal in the case of parallel graphs. However, it is not unreasonable to expect that a different penalty function other than F_{SR} and F_{RD} may outperform them.

Before we attempt to answer this question, we point out a limitation of RDA. Despite its good performance and lack of need for a fine-tuning parameter, a significant limitation of the weight function F_{RD}, hence RDA, is that it cannot be used when the disambiguation cost is zero. In many practical applications of SOSP, however, the disambiguation cost can be zero. A simple example is an instance of the problem where a disambiguation can be performed visually with a clear line of sight. Thus, in our quest for better penalty functions, we would like to be able to address this limitation.

A reasonable approach to handle zero disambiguation cost is to have the cost as an additive term in the penalty function. Furthermore, any meaningful penalty function needs to be monotonically nondecreasing in $c(e)$ and $\rho(e)$ for stochastic edges in discrete SOSP and for edges that intersect possible-obstacles in discretized SOSP. With these two observations in mind, we experimented with a large number of penalty functions with an additive cost term that are also monotonically nondecreasing in $c(e)$ and $\rho(e)$. We also tried penalty functions that account for different metrics in the obstacle field. One particular metric we considered was the distance of an edge's midpoint to the termination point t, which we denote by $d_t(e)$. Our experiments included an actual naval

minefield dataset, as discussed in Section 2.8, as well as synthetic data that possess similar characteristics to this minefield dataset. After extensive computational experiments, we observed that one particular penalty function consistently outperformed F_{RD} and other functions in most instances. This penalty function is presented below:

$$F_{DT}(e) := c(e) + \Big(\frac{d_t(e)}{1-\rho(e)}\Big)^{-\log(1-\rho(e))} \tag{2.11}$$

This function includes a $d_t(e)$ term and therefore it is called F_{DT}. The disambiguation algorithm that uses the F_{DT} penalty function with the NDR navigation strategy is called the DT Algorithm (DTA). In particular, DTA uses the weight below for discrete SOSP:

$$w_D^{DTA}(e) := \ell^e(e) + \mathbf{1}_{e\in\mathsf{E}'} \cdot \Big(c(e) + \Big(\frac{d_t(e)}{1-\rho(e)}\Big)^{-\log(1-\rho(e))}\Big) \tag{2.12}$$

and the weight below for discretized SOSP:

$$w_{LD}^{DTA}(e) := \ell^e(e) + \frac{1}{2}\sum_{i=1}^{|X|} \#\mathrm{comp}(e\backslash D_i)\cdot\mathbf{1}_{e\cap D_i\neq\emptyset}\cdot\Big(c(e)+\Big(\frac{d_t(e)}{1-\rho(e)}\Big)^{-\log(1-\rho(e))}\Big). \tag{2.13}$$

2.7.1 Illustration of the Algorithms

We now illustrate applications of the RD, SR, DT, and the optimal algorithms on the simple discretized SOSP instance shown in Figure 2.3, this time taking disk radii as 4.5 non-diagonal lattice edges. For consistency with our definition of discretized SOSP, this instance is scaled as follows: The starting point is taken as $s = (2, 6)$, termination as $t = (26, 6)$; first disk center as (8,6), and second disk center as (20,6). Marks of the first and second disks are taken as 0.2 and 0.1 respectively, and cost of disambiguation is taken as 0.4. The optimal algorithm we utilize is the BAO^* Algorithm. Introduced in [66], BAO^* improves upon the AO^* Algorithm by efficiently exploiting the problem structure and searches only a very small fraction of the solution space. Consequently, the algorithm uses significantly less computational resources compared to AO^* and stochastic dynamic programming. Superimposed walks as dictated by RDA are displayed in Figure 2.4(a). These walks are described below.

- Start at vertex s and disambiguate the first disk x_1 at vertex A. If x_1 is found to be a false obstacle, traverse to B and disambiguate x_2 at that vertex. If x_2 is found to be a false obstacle as well, directly traverse to t. If x_2 is found to be true, traverse to t while avoiding x_2; namely, via vertices C, D, E, and F.
- If x_1 is found to be a true obstacle, traverse to vertex D while avoiding x_1 and disambiguate x_2 at D. If it is found to be a false obstacle, traverse to t via vertex F. If x_2 is found true, traverse to t via vertices E and F while avoiding x_2. Total expected traversal length is 26.83 units.

Superimposed walks as dictated by the SR, DT, and BAO^* Algorithms are displayed in Figure 2.4(b) and explained below.

- Start at vertex s and disambiguate the first disk x_1 at vertex A. If x_1 is found to be a false obstacle, traverse to B and disambiguate x_2 at that vertex. If x_2 is found to be a false obstacle as well, directly traverse to t. If x_2 is found to be true, traverse to t while avoiding x_2; namely, via vertices C, D, E, and F. Note that these walks are exactly the same as in RDA.
- If x_1 is found to be a true obstacle, traverse to vertex C while avoiding x_1 and disambiguate x_2 at C. If it is found to be a false obstacle, traverse to t via vertex F. If x_2 is found true, traverse to t via vertices D, E and F while avoiding x_2. Total expected traversal length is 26.34 units.

The main difference between RDA and the other algorithms is that if x_1 is disambiguated and found to be a true obstacle, RDA dictates disambiguation of x_2 at vertex D whereas the other algorithms dictate its disambiguation at vertex C, resulting in a 0.49 units decrease in the expected traversal length. Thus, in this particular case, RDA fails to find the optimal policy while SRA and DTA do not.

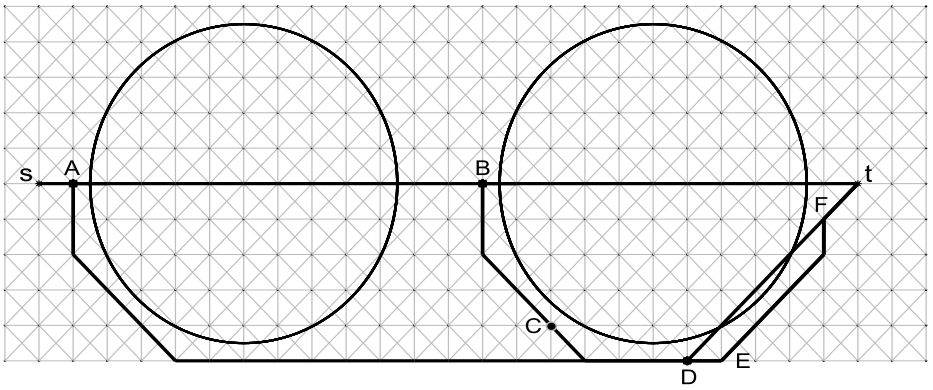
2.8 Computational Experiments

This section empirically compares the performances of SR, RD, and DT algorithms. The specific application domain we consider is maritime minefield navigation, which has received considerable attention from scientific and engineering communities recently [53, 67]. A particular instance we consider is a U.S. Navy minefield dataset (called the COBRA data) that first appeared in Witherspoon et al. [53] and was later referred to in Fishkind et al. [63], Priebe et al. [68, 69], Ye and Priebe [70], Ye et al. [71], and Aksakalli et al. [62]. The COBRA data is illustrated in Figure 2.5 and tabulated in Table 2.2. This dataset has a total of 39 disk-shaped possible-obstacles: 12 of these disks are mines (i.e., true obstacles) and the remaining ones are clutter (that is, false obstacles). For convenience, original data coordinates were scaled and shifted so that disk centers are inside the region $[10, 90] \times [10, 90]$. The starting point is $s = (54, 80)$ and the termination point is $t = (54, 10)$ with disk radius taken as $r = 5$.

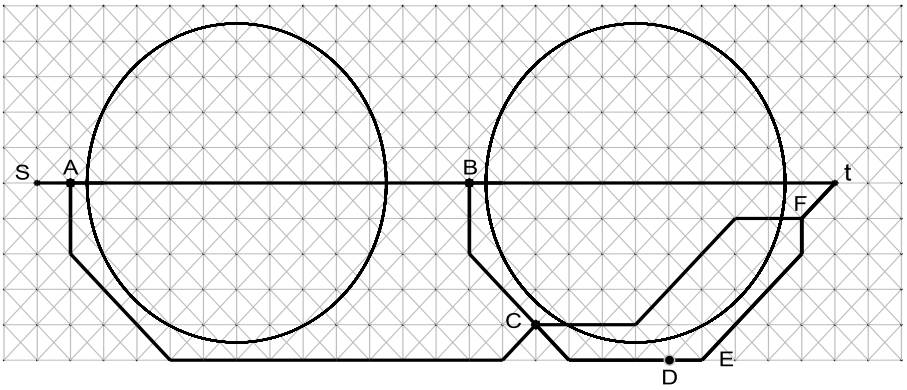
Our experiments were conducted in the following three simulation environments:

Environment A: The actual COBRA data.

Environment B: COBRA-like instances with 12 true and 27 false disk-shaped obstacles. Centers of these 39 disks were randomly sampled from the uniform distribution over the region $[10, 90] \times [10, 90]$. To make the disk layout more formidable



(a) Superimposed walks as dictated by RDA. In this particular case, RDA fails to find the optimal policy.



(b) Superimposed walks as dictated by SRA, DTA, and the optimal algorithm BAO^*.

FIGURE 2.4: Illustration of the RD, SR, DT, and optimal algorithms on the problem instance shown in Figure 2.3, this time with disk radii taken as 4.5 non-diagonal lattice edges.

TABLE 2.2: Scaled and shifted center coordinates and marks of COBRA disks. Disks in the first nine rows are false obstacles whereas the ones in the last four rows (shown in bold) are true obstacles.

x-coord.	**y-coord.**	ρ	**x-coord.**	**y-coord.**	ρ	**x-coord.**	**y-coord.**	ρ
46.13	39.61	0.0731	50.49	24.26	0.1033	83.62	16.33	0.1165
30.21	54.62	0.1379	56.83	20.50	0.1527	44.87	66.45	0.1668
47.88	34.51	0.1718	40.55	76.93	0.1939	43.43	26.22	0.2575
21.93	53.22	0.3309	69.82	51.65	0.4353	65.64	11.08	0.4412
37.36	29.94	0.4917	29.47	37.21	0.5215	59.42	20.11	0.5418
38.90	57.22	0.5609	32.07	31.37	0.5745	45.71	24.83	0.5831
86.12	15.83	0.5902	52.01	56.80	0.5994	41.14	27.41	0.6200
8.43	74.26	0.6399	37.00	43.89	0.6416	72.53	18.22	0.6527
22.98	40.29	0.6543	70.33	18.61	0.6564	29.78	32.15	0.6566
63.54	**24.81**	**0.1887**	**64.04**	**37.65**	**0.5149**	**27.00**	**37.97**	**0.5280**
46.07	**71.00**	**0.5609**	**65.16**	**64.01**	**0.5653**	**37.36**	**18.03**	**0.6108**
39.43	**70.31**	**0.6171**	**75.51**	**42.83**	**0.6189**	**76.11**	**55.73**	**0.6405**
38.29	**44.20**	**0.6444**	**28.16**	**64.10**	**0.6567**	**64.55**	**50.98**	**0.8515**

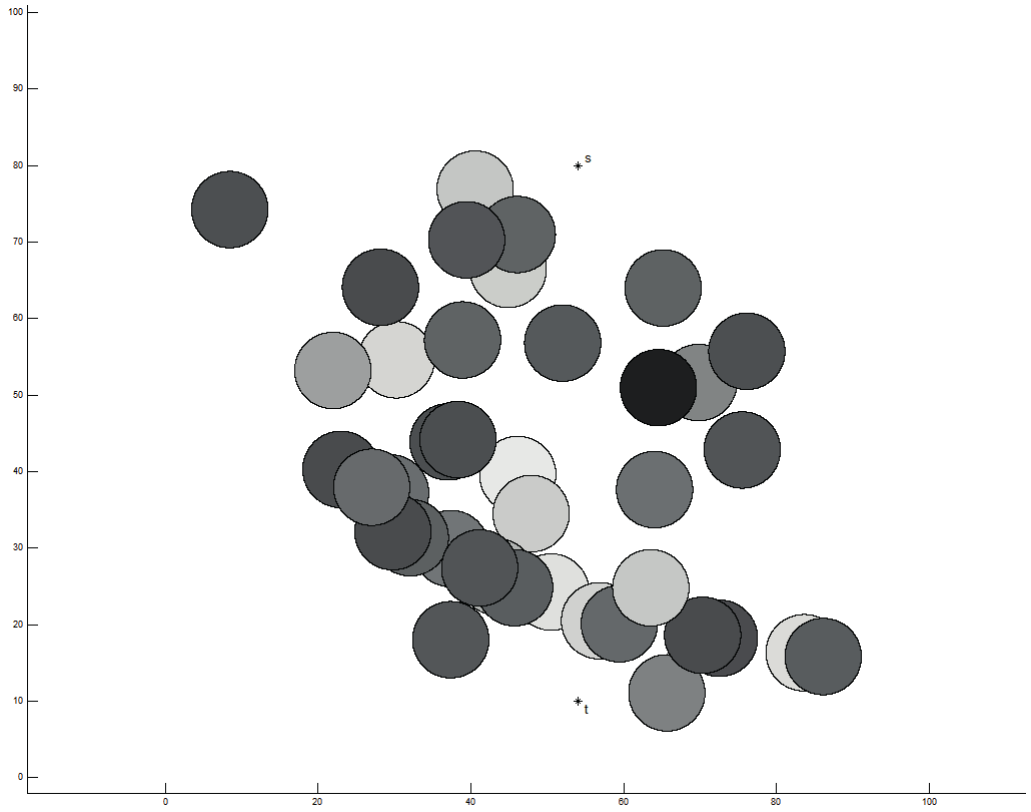


FIGURE 2.5: Illustration of the COBRA data. In the figure, gray intensity scale of disks reflects marks of each disk with darker colors indicating a higher mark.

in this environment, it was conditioned that the zero-risk $s-t$ path length was at least 130 units. Here, the zero-risk $s-t$ path is defined as the shortest $s-t$ path over the integer lattice that avoids all stochastic edges, i.e., the edges intersecting any disks.

Environment C: Instances with 40 true and 100 false disk-shaped obstacles. As in Environment B, centers of the false obstacles were randomly sampled from the uniform distribution over the region $[10, 90] \times [10, 90]$. Centers of the true obstacles, however, were sampled from a V-shaped obstacle-placement window, as described in Section 2.8.3.

In Environments B and C, marks of the true obstacles were sampled from Beta(2,6) (with a mean of 0.75) and marks of the false ones were sampled from Beta(6,2) (with a mean of 0.25). Also, the starting and termination points were taken as $s = (50, 100)$ and $t = (50, 1)$ respectively for both of the environments. In addition, in all three environments, the navigation area was considered to be the 8-adjacent integer lattice over $[1, 100] \times [1, 100]$ with disk radius being $r = 5$. This setup ensures that there is always an admissible path from s to t.

Computing the expected length of a walk in all three variants of the SOS problem requires computation of walk lengths for each possible outcome of any disambiguations performed. Thus, complexity of computing expected walk length of any policy is $O(2^K)$. In other words, even though a penalty-based algorithm can be executed efficiently in realtime, computation of the expected length of the associated walk is exponential in K.

In Environments A and B, we compare performances of SRA, RDA, and DTA and we only consider cases where $K = 1$ or $K = 2$. In Environment C, we let $K = \infty$ and we compare performances of only RDA and DTA (SRA is not included in the comparison due to its excessive run times required for meshing of the α parameter). Our goal in Environment C is two-fold: (1) compare RDA and DTA in the presence of an unlimited disambiguation capability, and (2) compare performances of these algorithms when true obstacles are placed strategically inside the navigation area. Regarding the first goal, computation of the expected walk length for unlimited K is computationally infeasible due to the exponential nature of the process. For this reason, instead of the expected walk length, we compare RDA and DTA based on the lengths of the actual $s-t$ walks as dictated by the respective algorithms. Within the context of the second goal, Aksakalli and Ceyhan [60] consider the problem of identifying optimal obstacle placement patterns in SOSP that maximize traversal length of the navigating agent in a game-theoretic sense. Our second goal therefore is a rather interesting analysis from a game theory point of view as what we investigate is whether performance of our disambiguation algorithms is affected by specific location of the true obstacles as determined by an obstacle placing agent.

Another particular characteristic we would like to investigate is the sensitivity of the performances of the navigation algorithms to the cost of disambiguation. For this purpose, we consider 7 different disambiguation costs ($c = 0, 1, 2, 4, 6, 8, 10$) in each one of the above environments where it is assumed that disambiguation cost is the same across all the disks.

2.8.1 Environment A (The COBRA Data) Experiments

This section compares the performances of the SR, RD, and DT algorithms for the COBRA data. In this section only, we also include the optimal policy in the comparison where this policy is obtained via the BAO^* Algorithm. Comparison results are presented in Table 2.3. On a 3.8 gigaHertz personal computer, execution time of both the RD and DT algorithms was 0.312 seconds per run on the average for whereas that of the SR algorithm was 18.5 seconds per run. Total run time required for computation of the optimal policy in Table 2.3, on the other hand, was 11 days and 17 hours. In the table, expected length of the optimal policy is denoted by $E^{OPT}(c)$ for a disambiguation cost of c. The expected length of the policy corresponding to the best α value for SRA is denoted by $\mathrm{E}^{SRA}(c)$ whereas expected lengths of the policies obtained by RDA and DTA

are denoted by $E^{RDA}(c)$ and $E^{DTA}(c)$ respectively. Percent deviation of the expected walk lengths found by the suboptimal algorithms from that of the optimal policy is denoted by $\%DO(c)$ superscripted by the algorithm name. For instance, $\%DO^{RDA}(c) = \frac{E^{RDA}(c)-E^{OPT}(c)}{E^{OPT}(c)} * 100$.

TABLE 2.3: Cobra data simulation results.

K	c	$E^{OPT}(c)$	$E^{RDA}(c)$	$E^{DTA}(c)$	$E^{SRA}(c)$	$\%DO^{RDA}(c)$	$\%DO^{DTA}(c)$	$\%DO^{SRA}(c)$
1	0	80.02	-	80.17	80.02	-	0.18	0.00
	1	81.02	105.33	81.17	81.02	30.00	0.18	0.00
	2	82.02	82.17	82.17	82.02	0.18	0.18	0.00
	4	84.02	84.02	84.17	84.02	0.00	0.17	0.00
	6	86.02	86.02	86.17	86.02	0.00	0.17	0.00
	8	88.02	88.17	88.17	88.02	0.17	0.17	0.00
	10	90.02	90.17	90.17	90.02	0.16	0.16	0.00
2	0	75.47	-	80.25	77.37	-	6.34	2.51
	1	77.47	102.72	78.63	78.47	32.59	1.50	1.29
	2	79.47	82.36	79.74	79.57	3.64	0.33	0.13
	4	81.77	84.58	81.94	81.78	3.44	0.21	0.01
	6	83.97	83.99	84.15	83.99	0.02	0.21	0.02
	8	86.18	86.43	86.36	86.19	0.28	0.20	0.02
	10	88.39	88.63	88.56	88.40	0.28	0.20	0.01

As expected, SRA shows somewhat better performance compared to DTA (and especially RDA) as it finetunes the penalty term via the α parameter, though it runs about 60 times slower compared to either algorithms. RDA is not even applicable for $c = 0$, and it shows the worst performance at $\%DO^{RDA}(1) = 30$ for $K = 1$ and, $\%DO^{RDA}(1) = 32.59$ for $K = 2$ respectively. However, $\%DO^{RDA}(c \geq 6)$ is below 0.3 for both K's.

In comparison, for $K = 1$, $\%DO^{DTA}(c \geq 0)$ is below 0.2 whereas for $K = 2$, median $\%DO^{DTA}(c \geq 0)$ is merely 0.21. Also, maximum $\%DO^{DTA}(c \geq 1)$ is 1.5 whereas maximum $\%DO^{RDA}(c \geq 1)$ is significantly higher at 32.59. In addition, the difference between $\%DO^{RDA}(c \geq 1)$ and $\%DO^{SRA}(c \geq 1)$ is never more than 0.21. Thus, in general, solutions obtained by DTA compare favorably to both the optimal solutions as well as those obtained by SRA for the COBRA data. The same observation holds for RDA, but only when $c \geq 6$.

2.8.2 Environment B Experiments

This section compares performances of RDA, DTA, and SRA on COBRA-like instances with 12 true and 27 false disk-shaped obstacles where disk centers were randomly sampled from the uniform distribution over the region $[10, 90] \times [10, 90]$. We generated 100 of such instances where the zero-risk $s - t$ path length was conditioned to be at least 130 units.

Comparison results including means and standard deviations of the expected lengths along with the zero-risk lengths are presented in Table 2.4. Let $E^{RDA}_{mean}(c)$ and $E^{RDA}_{std}(c)$ denote the mean and standard deviation of the expected lengths of the solutions obtained

by RDA for disambiguation cost c. For all the c, K combinations considered, we observe that $E^{DTA}_{mean} < E^{RDA}_{mean}$ and that $E^{DTA}_{std} < E^{RDA}_{std}$. Interestingly, the difference in the means increases as c decreases.

TABLE 2.4: Environment B simulation results.

K	c	Zero-Risk		$E^{RDA}(c)$		$E^{DTA}(c)$		$E^{SRA}(c)$		RDA-DTA	DTA-SRA
		mean	std	mean	std	mean	std	mean	std	mean	mean
1	0.00	134.08	4.80	-	-	121.34	12.70	118.16	7.16	-	3.18
	1.00	134.08	4.80	130.04	13.99	121.52	12.11	118.97	6.92	8.52	2.55
	2.00	134.08	4.80	127.85	14.04	122.18	11.86	119.94	6.86	5.67	2.24
	4.00	134.08	4.80	124.89	10.46	123.35	9.74	121.85	6.69	1.54	1.50
	6.00	134.08	4.80	126.04	10.47	125.32	9.71	123.71	6.44	0.72	1.61
	8.00	134.08	4.80	127.48	10.09	126.60	7.50	125.50	6.14	0.87	1.11
	10.00	134.08	4.80	128.42	7.48	128.34	7.24	127.14	5.72	0.08	1.20
2	0.00	134.08	4.80	-	-	114.12	7.13	112.32	5.26	-	1.80
	1.00	134.08	4.80	122.14	11.07	115.23	6.67	113.84	5.23	6.91	1.39
	2.00	134.08	4.80	121.10	11.06	116.48	6.64	115.25	5.25	4.62	1.22
	4.00	134.08	4.80	120.47	8.46	118.93	6.95	117.89	5.42	1.54	1.04
	6.00	134.08	4.80	121.90	7.53	121.36	7.11	120.39	5.59	0.53	0.97
	8.00	134.08	4.80	123.78	7.36	123.41	6.79	122.77	5.67	0.37	0.63
	10.00	134.08	4.80	125.72	6.83	125.59	6.72	124.98	5.60	0.13	0.61

We now digress briefly and consider how $E^{OPT}(c)$ changes if c is increased by $\delta > 0$ units. For $K = 1$, if the optimal policy requires a disambiguation, then it holds that $E^{OPT}(c + \delta) = E^{OPT}(c) + \delta$, which can easily be shown by contradiction. For $K \geq 2$, let us consider a special case where the optimal policy requires exactly K disambiguations regardless of the outcomes of previous disambiguations (such a scenario is likely to be the case when K is small and number of possible-obstacles is large). In that case, if c is increased by δ units, then in the best possible scenario, it would hold that $E^{OPT}(c + \delta) = E^{OPT}(c) + \delta$ (this can also be shown by contradiction). However, $E^{OPT}(c) + \delta$ is merely a lower bound for $E^{OPT}(c + \delta)$. Appendix A provides a simple parallel graph example where the optimal expected length increases by 2.22 units when the cost is increased by 2 units. Another example is the COBRA data: for $K = 2$, when the disambiguation cost is increased from 4 to 6, the optimal expected length increases from 81.77 to 83.97, which is a 2.2 units increase. We conjecture that for any discrete SOS problem instance for which the optimal policy dictates at least one disambiguation, it holds that $E^{OPT}(c + \delta) \geq E^{OPT}(c) + \delta$.

Back to the simulation results, a close inspection reveals a rather peculiar behavior regarding RDA. For $K = 1$, $E^{RDA}_{mean}(1) \approx 130$ whereas $E^{RDA}_{mean}(2) \approx 128$ and $E^{RDA}_{mean}(4) \approx 125$. A similar behavior is exhibited for $K = 2$. The observation that E^{RDA}_{mean} decreases as the disambiguation cost increases (where in fact it should be the opposite) suggests the following: the penalty function $F_{RD}(e) = \frac{c(e)}{1-\rho(e)}$ is perhaps not providing "the right amount of penalty" to guide the navigation when c is relatively small. An alternative interpretation is that performance of RDA seems to improve as the disambiguation cost increases. The fact that $\%DO^{RDA}$ is below 0.3 only when $c \geq 6$ for the COBRA data is another indication that RDA requires relatively high disambiguation costs for adequate performance. This behavior, on the other hand, can be seen as an important limitation

of RDA—which is in addition to the limitation that this algorithm cannot be used in the case of zero disambiguation cost.

In contrast, for all the c, K combinations considered, E^{DTA}_{mean} strictly increases as c increases. Thus, DTA does not seem to suffer from the limitation of RDA mentioned above. In addition, median difference between E^{DTA}_{mean} and E^{SRA}_{mean} is merely 1.3 units for the entire Table 2.4.

2.8.3 Environment C Experiments

This section compares performances of RDA and DTA on instances with 40 true and 100 false disk-shaped obstacles in the presence of an unlimited disambiguation capability. Centers of the false obstacles were randomly sampled from the uniform distribution over the region $[10, 90] \times [10, 90]$. Similar to what was done in Aksakalli and Ceyhan [60], centers of the true obstacles were sampled from a V-shaped obstacle placement window with a vertical width of 10 units. Top left corner of this window was taken as $(x, y) = (10, 70)$, with the remaining corner points being (50,40), (90,70), (90,60), (50,30), and (10,60).

Comparison results for 100 randomly generated such instances are presented in Table 2.5. In the table, actual traversal lengths of the policies obtained by RDA and DTA are denoted by $A^{RDA}(c)$ and $A^{DTA}(c)$ respectively. These lengths are calculated by using the actual status information of disks as the agent navigates and performs disambiguations in the obstacle field. Number of instances for which the actual traversal lengths exceed the zero-risk path lengths are shown in columns labeled "#Exceed".

Table 2.5: Environment C simulation results.

K	c	Zero-Risk		$A^{RDA}(c)$			$A^{DTA}(c)$			$A^{RDA}(c) - A^{DTA}(c)$
		mean	std	mean	std	#Exceed	mean	std	#Exceed	mean
∞	0	159.91	5.79	-	-	-	141.23	19.47	2	-
	1	159.91	5.79	208.20	68.20	70	145.45	18.15	2	62.76
	2	159.91	5.79	210.20	70.98	64	148.14	17.26	2	62.06
	4	159.91	5.79	185.10	65.14	32	152.11	15.52	2	32.99
	6	159.91	5.79	171.55	50.43	18	154.63	9.90	1	16.92
	8	159.91	5.79	162.51	31.82	7	157.00	8.64	1	5.51
	10	159.91	5.79	160.57	20.36	4	157.99	7.18	0	2.58

Similar to the simulation results in Environment B, we observe that A^{RDA}_{mean} decreases as the disambiguation cost increases, this time even more drastically. For instance, $A^{RDA}_{mean}(1) \approx 208$ whereas $A^{RDA}_{mean}(10) \approx 161$. This indicates that performance of RDA detoriates significantly for small c in this particular simulation environment. In addition, $A^{RDA}_{\#Exceed}(1) = 70$ out of 100 instances. Likewise, $A^{RDA}_{\#Exceed}(2) = 64$ and $A^{RDA}_{\#Exceed}(4) = 32$, which are all relatively high values. On the other hand, $A^{DTA}_{\#Exceed}$ never exceeds 2 for any of the cost values considered. One other observation is that A^{RDA}_{mean} always exceed the corresponding zero-risk length mean, which essentially suggests that, on the

average, RDA does not provide any improvement over the zero-risk path in actual $s-t$ traversals. In contrast, A^{DTA}_{mean} is always smaller than the corresponding zero-risk length mean, thereby providing the navigating agent a strict improvement over the zero-risk path on the average. In fact, the difference between A^{RDA}_{mean} and A^{DTA}_{mean} can be as high as 62.76 units (for $c = 1$), though this difference reduces as the disambiguation cost increases. Regarding the standard deviations, A^{DTA}_{std} is considerably smaller compared to A^{RDA}_{std} for all the cost values considered. That is, in general, DTA provides substantially better policies compared to RDA on the average (especially for smaller c) while having a much smaller standard deviation. Also in favor of DTA is the observation that A^{DTA}_{mean} strictly increases as c increases.

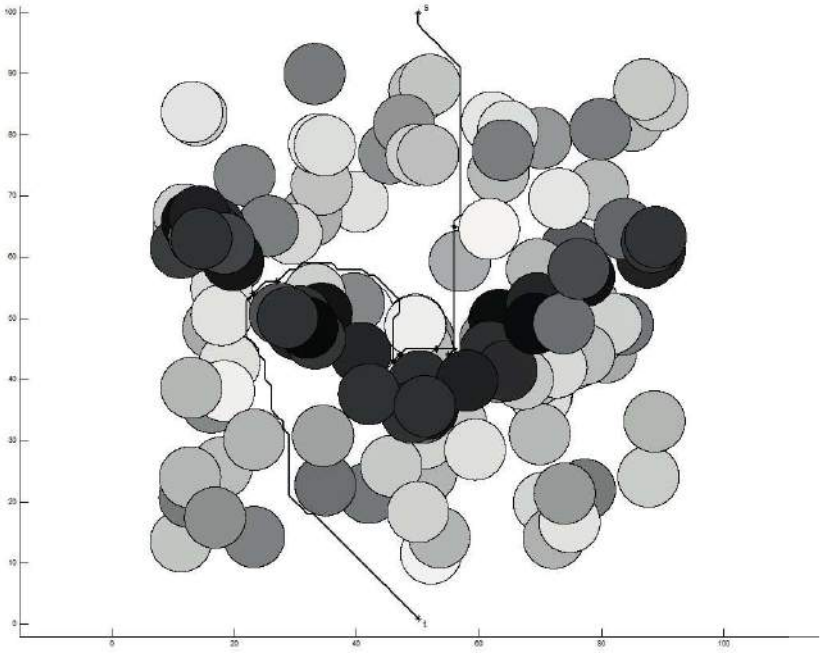
Illustrated in Figure 2.6 is a problem instance in Environment C and the $s-t$ traversals as dictated by RDA and DTA respectively for $c = 2$. In this particular case, zero-risk length is 156.78 whereas $A^{RDA} = 204.54$ and $A^{DTA} = 131.12$. It appears from the figure that RDA gets trapped inside the elbow-like region of the V-shaped area whereas DTA quickly finds the passage on the left side of the V-shape and then directly traverses to t.

2.9 Summary and Conclusions

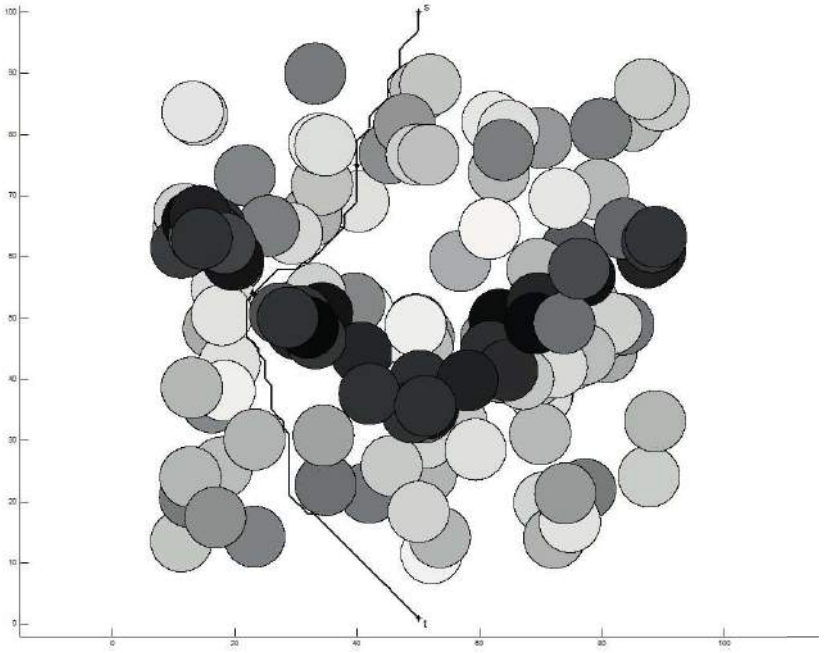
The stochastic obstacle scene (SOS) problem is a challenging stochastic optimization problem that has practical applications in important domains such as robot navigation in stochastic environments, minefield navigation, and adaptive traffic routing.

Two previously introduced suboptimal algorithms for the SOS problem are the Simulated Risk (SR) and Reset Disambiguation (RD) algorithms. SRA is based on the idea of temporarily pretending that ambiguous regions are riskily traversable. On the other hand, the idea behind RDA is to use the optimal navigation strategy in a reset variant as a suboptimal strategy in the original problem. In this chapter, we adapt SRA and RDA originally proposed for continuous SOSP to discrete and lattice-discretized SOSP. We then present a polynomial-time method when the associated graph is restricted to parallel graphs. Having identified this method, we make a rather interesting observation that the optimal edge weights in this parallel graph special case is the same as the weights in the reset variant of the original problem, and hence RDA. This connection stands as an alternative interpretation of RDA.

Both SRA and RDA employ a navigate-disambiguate-repeat (NDR) strategy guided by particular penalty functions. A major downside of SRA is that it needs to fine-tune the penalty term via brute-force to achieve reasonable performance levels. RDA does not require such a fine-tuning parameter, yet, it has a significant limitation in the sense that it cannot be used when the disambiguation cost is zero.



(a) Navigation dictated by RDA



(b) Navigation dictated by DTA

FIGURE 2.6: An instance in Environment C and $s - t$ traversals as dictated by RDA and DTA respectively.

In an attempt to address respective shortcomings of SRA and RDA, we first propose a generalized framework encompassing these algorithms that uses penalty functions to guide the navigation in realtime. Within this framework, we introduce a new suboptimal algorithm called the DT algorithm that uses a new penalty function that takes into account edge distances to the termination point. DTA addresses limitations of both SRA and RDA in that it does not require a fine-tuning parameter and it can be used even with a zero disambiguation cost. Computational experiments involving an actual minefield dataset called the COBRA data suggest that DTA provides near-optimal results with minimal computational resources. In the meantime, simulations involving COBRA-like synthetic data indicate a rather subtle weakness of RDA: performance of this algorithm depends heavily on the disambiguation cost. In particular, RDA requires relatively large costs for acceptable performance. In contrast, DTA did not suffer from this weakness in our experiments and consistently gave superior results regardless of the cost.

At this point, a critical observation needs to be made: Despite the fact that DTA performed remarkably well for COBRA and COBRA-like problem instances in our simulations, it may or may not perform at the same level on obstacle fields with different topologies or with non-circular obstacle regions. Further research on instances with different characteristics is required in order to confirm that high performance of DTA is consistent across various problem settings. To that end, it might as well be the case that perhaps a different penalty function outperforms that of DTA in certain problem environments. Nonetheless, the NDR strategy guided by appropriate penalty functions seems to be an efficient and effective algorithmic framework for SOSP, and our study should be seen as a show case of this framework using the DT penalty function on an important real-world variant of the problem.

Appendix A

Impact of Cost Change in Parallel Graphs

This section provides an example of a parallel graph for which optimal policy changes when the disambiguation cost changes. The parallel graph in this simple instance has two edges e_1 and e_2 with respective lengths $\ell_1 = 1.55$, $\ell_2 = 3.97$ and marks $\rho_1 = 0.55$, $\rho_2 = 0.08$. Two different costs are considered: $c = 2$ and $c = 4$ where $c_1 = c_2 = c$. Note that there are only two feasible policies in this case, which are denoted by $P_1 = \{e_1, e_2\}$ and $P_2 = \{e_2, e_1\}$. In particular, P_1 dictates disambiguation of e_1 and then e_2, whereas the ordering in P_2 is the opposite. For $c = 2$ and $c = 4$, expected length calculations corresponding to policies P_1 and P_2 are shown below where the optimal policies are marked with an asterisk for the respective costs:

- $E^{P_1^*}(2) = 2 + (1 - .55)(1.55) + .55(2 + (1 - .08)(3.97)) = 5.81$,
- $E^{P_2}(2) = 2 + (1 - .08)(3.97) + .08(2 + (1 - .55)(1.55)) = 5.87$,
- $E^{P_1}(4) = 4 + (1 - .55)(1.55) + .55(4 + (1 - .08)(3.97)) = 8.91$,
- $E^{P_2^*}(4) = 4 + (1 - .08)(3.97) + .08(4 + (1 - .55)(1.55)) = 8.03$.

Interestingly, when cost is increased from 2 to 4, policy P_1 is no longer optimal. Thus, the optimal disambiguation sequence changes when the cost changes. In this particular case, $E^{P_1}(4) = 8.91 = E^{P_1}(2) + 3.1$. In addition, again when cost is increased from 2 to 4, the optimal expected length increases from 5.81 to 8.03, which is a 2.22 units increase.

Bibliography

[1] UN. *Review of maritime transport*. United Nations UNCTAD Secretatiat, 2011.

[2] T.C. Gillmer. *Modern Ship Design*. Naval Institute Press, Annapolis, MD, 1975.

[3] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[4] P.E. Hart, N.J Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost graphs. *IEEE Transactions on Systems Science and Cybernetics*, 4:100–107, 1968.

[5] 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.

[6] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[7] A. Bicchi, G. Casalino, and C. Santilli. Planning shortest bounded-curvature paths for a class of nonholonomic vehicles among obstacles. *Journal of Intelligent and Robotic Systems*, 16:387–405, 1996.

[8] R. Takei, R. Tsai, H. Shen, and Y. Landa. A practical path-planning algorithm for a simple car: a hamilton-jacobi approach. *American Control Conference (ACC)*, 2010.

[9] S. M. LaValle. Rapidly-exploring random trees: a new tool for path planning. Technical report, Computer Science Dept., Iowa State University, Oct. 1998.

[10] K. Yang and S. Sukkarieh. Real-time continous curvature path planning of uavs in cluttered environments. *In Proceeding of the 5th International Symposium on Mechatronics and its Applications(ISMA08), Amman, Jordan*, 2008.

[11] K. Fagerholt, S. Heimdal, and A. Loktu. Shortest path in the presence of obstacles: An application to ocean shipping. *Journal of the Operational Research Society*, 51: 683–688, 2000.

[12] H. Lee, G. Kong, S. Kim, C. Kim, and J. Lee. Optimum ship routing and it's implementation on the web. *Lecture Notes in Computer Science*, pages 125–136, 2002.

[13] M-C. Fang, J-H. Luo, and M-L. Lee. A nonlinear mathematical model for ship turning circle simulation in waves. *Journal of Ship Research*, 49(2):69–79, 2005.

[14] S.K. Bhattacharyya and M.R. Haddara. Parametric identification for nonlinear ship maneuvering. *Journal of Ship Research*, 50(3):197–207, 2006.

[15] Efstathios Bakolas and Panagiotis Tsiotras. Optimal synthesis of the asymmetric sinistral/dextral Markov–Dubins problem. *Journal of optimization theory and applications*, 150(2):233–250, 2011.

[16] M.J. van Hilten and P.H.M. Wolkenfelt. The rate of turn required for geographically fixed turns: A formula and fast-time simulations. *Journal of Navigation*, 53:146–155, 2000.

[17] T. Caldwell. On finding minimum routes in a network with turn penalties. *Communications of the ACM*, 4(2):107–108, 1961.

[18] A. Boroujerdi and J. Uhlmann. An efficient algorithm for computing least cost paths with turn constraints. *Information Processing Letters*, 67:317–321, 1998.

[19] J. Clossey, G. Laporte, and P. Soriano. Solving arc routing problems with turn penalties. *The Journal of the Operational Research Society*, 52(4):433–439, 2001.

[20] D. Soler, E. Martinez, and J. C. Mico. A transformation for the mixed general routing problem with turn penalties. *The Journal of the Operational Research Society*, 59(4):540–547, 2007.

[21] J. Albiach, J. M. Sanchis, and D. Soler. An asymmetric TSP with time windows and with time-dependent travel times and costs: an exact solution through a graph transformation. *European Journal of Operational Research*, 189:789–802, 2008.

[22] J.L. Solka, J.C. Perry, B.R. Poellinger, and G.W. Rogers. Fast computation of optimal paths using a parallel Dijkstra algorithm with embedded constraints. *Neurocomputing*, 8:195–212, 1995.

[23] E. Gutierrez and A.L. Medaglia. Labeling algorithm for the shortest path problem with turn prohibitions with application to large-scale road networks. *Annals of Operations Research*, 157:169–182, 2008.

[24] D. Villeneuve and G. Desaulniers. The shortest path problem with forbidden paths. *European Journal of Operational Research*, 165(1):97–107, 2005.

[25] J.O. Royset, W.M. Carlyle, and R.K. Wood. Routing military aircraft with a constrained shortest-path algorithm. *Military Operations Research*, 14(3):31–52, 2009.

[26] A. Perugiaa, L. Mocciab, J-F. Cordeaud, and G. Laporte. Designing a home-to-work bus service in a metropolitan area. *Transportation Research Part B: Methodological*, 45:1710–1726, 2011.

[27] R. Geisberger and C. Vetter. Efficient routing in road networks with turn costs. *Experimental Algorithms 10th International Symposium, SEA 2011, Proceedings*, 2011.

[28] J. C. Mico and D. Soler. The capacitated general windy routing problem with turn penalties. *Operations Research Letters*, 39(4):265–271, 2011.

[29] L. D. P. Pugliese and F. Guerriero. Shortest path problem with forbidden paths: the elementary version. *European Journal of Operational Research, DOI: 10.1016/j.ejor.2012.11.010*, 2012.

[30] S. Vanhove and V. Fack. Route planning with turn restrictions: a computational experiment. *Operations Research Letters*, 40(5):342–348, 2012.

[31] M. Zabarankin, S. Uryasev, and P. Pardalos. Optimal risk path algorithms. In R. Murphey and P.M. Pardalos, editors, *Cooperative Control and Optimization*, pages 273–298. Kluwer Academic Publishers, Netherlands, 2002.

[32] A. Bartlett, A. Berger, C. Lipkin, N. Mavinga, E. Perez, E. Tweedy, and E. Wheeler. Problem 1: Optimal mission planning. *In Proceedings of the Eleventh Industrial Mathematical and Statistical Modeling Workshop for Graduate Students*, pages 2–30, 2005.

[33] R.J. Szczerba, P. Galkowski, I.S. Glickstein, and N. Ternullo. Robust algorithm for real-time route planning. *IEEE Transactions on Aerospace and Electronic Systems*, 36(3):869–878, 2000.

[34] M. Zabarankin, S. Uryasev, and R. Murphey. Aircraft routing under the risk of detection. *Naval Research Logistics*, 53(8):728–747, 2006.

[35] E. Edison and T. Shima. Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms. *Computers & Operations Research*, 38(1):340–356, 2011.

[36] Jiann-Shing Wu and Jin-Jang Leou. New polygonal approximation schemes for object shape representation. *Pattern Recognition*, 26(4):471–484, 1993.

[37] American Bureau of Shipping. *Guide for vessel maneuverability.* Houston, TX, 2006.

[38] N.J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann, Palo Alto, CA, 1980.

[39] T. Berglund. *Path-planning with obstacle-avoiding minimum curvature variation b-splines*. Lulea University of Technology, Licentiate Thesis, 2003.

[40] N. Wang, D. Zhang, L. Zhou, and Q. Liu. Near optimal path planning for vehicle with heading and curvature constraints. *In Proceedings of the 8th World Congress on Intelligent Control and Automation, Jinan, China*, 2010.

[41] E. Hanna. How and why does sea ice vary? *Journal of Meteorology*, 23(229): 153–158, 1998.

[42] J. Ho. The implications of Arctic sea ice decline on shipping. *Marine Policy*, 34: 713–715, 2010.

[43] C. Shen and W. Shi. Review of climate change in the Arctic. *Energy Procedia*, pages 2466–2473, 2011.

[44] K.J. Wilson, J. Falkingham, H. Melling, and R. De Abreu. Shipping in the Canadian Arctic: other possible climate change scenarios. *In Proceedings of the International Geoscience and Remote Sensing Symposium*, 2004.

[45] J. Buysse. *Handling ships in ice*. The Nautical Institute, London, 2007.

[46] A. Okabe, B. Boots, K. Sugihara, and S.N. Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley & Sons, Hoboken, NJ, 2000.

[47] A. Nash, K. Daniel, S. Koenig, and A. Felner. Theta*: any-angle path planning on grids. *In Proceedings of the AAAI Conference on Artificial Intelligence*, 2007.

[48] C.H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Comp. Sci.*, 84:127–150, 1991.

[49] D.M. Blei and L.P. Kaelbling. Shortest paths in a dynamic uncertain domain. *In Proc. IJCAI Workshop on Adaptive Spatial Representations of Dynamic Environments*, 1999.

[50] D. Ferguson, A. Stenz, and S. Thrun. PAO* for planning with hidden state. *In Proc. the 2004 IEEE Internat. Conf. on Robotics and Automation*, 2004.

[51] 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.

[52] D.L. Smith. Detection technologies for mines and minelike targets. *In Proc. SPIE*, 2496:404–408, 1995.

[53] 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. *In Proc. SPIE*, 2496:500–508, 1995.

[54] J. Fawcett and P. Robinson. Adaptive routing for road traffic. *IEEE Comp. Graphics Appl.*, 20(3):46–53, 2000.

[55] S. Gao and I. Chabini. Optimal routing policy problems in stochastic time-dependent networks. *Transp. Res. Part B-Methodological*, 40(2):93–122, 2006.

[56] E. Nikolova and D. R. Karger. Route planning under uncertainty: the Canadian traveller problem. *In Proc. the 23rd AAAI Conf. on Artificial Intelligence, Chicago, Illinois*, 2008.

[57] Patrick Eyerich, Thomas Keller, and Malte Helmert. High-quality policies for the canadian traveler's problem. *In Proc. the Third Annual Symposium on Combinatorial Search*, 2010.

[58] M. Likhachev and A. Stentz. Probabilistic planning with clear preferences on missing information. *Artificial Intelligence*, 173:696–721, 2009.

[59] Y. Xu, M. Hu, B. Su, B. Zhu, and Z. Zhu. The Canadian traveller problem and its competitive analysis. *J. Combinatorial Opt.*, 18:195–205, 2009.

[60] V. Aksakalli and E. Ceyhan. Optimal obstacle placement with disambiguations. *Ann. Appl. Stat., DOI: 10.1214/12-AOAS556*, 2012.

[61] J.S. Provan. A polynomial-time algorithm to find shortest paths with recourse. *Networks*, 41(2):115–125, 2003.

[62] V. Aksakalli, D.E. Fishkind, C.E. Priebe, and X. Ye. The reset disambiguation policy for navigating stochastic obstacle fields. *NRL*, 58:389–399, 2011.

[63] D.E. Fishkind, C.E. Priebe, K. Giles, L.N. Smith, and V. Aksakalli. Disambiguation protocols based on risk simulation. *IEEE Trans. on Systems, Man, and Cybernetics, Part A*, 37(5):814–823, 2007.

[64] J. Blatz, D.E. Fishkind, and C.E. Priebe. Efficient, optimal stochastic-action selection when limited by an action budget. *Math. Meth. Oper. Res.*, 72:63–74, 2010.

[65] V. Aksakalli, D.E. Fishkind, and C.E. Priebe. Adaptation of the simulated risk disambiguation protocol to a discrete setting. *In Proc. ICAPS Workshop on POMDP, Classification and Regression: Relationships and Joint Utilization*, 2006.

[66] V. Aksakalli. The BAO* algorithm for stochastic shortest path problems with dynamic learning. *In Proc. the 46th IEEE Conf. on Decision and Control, New Orleans, LA*, 2007.

[67] R. Muhandiramge. *Maritime manoeuvring optimization: path planning in minefield threat environments.* University of Western Australia, Ph.D. Dissertation, 2008.

[68] C.E. Priebe, T.E. Olson, and D.M. Healy. Exploiting stochastic partitions for minefield detection. *In Proc. the SPIE*, 3079:508–518, 1997.

[69] C.E. Priebe, D.E. Fishkind, L. Abrams, and C.D. Piatko. Random disambiguation paths for traversing a mapped hazard field. *NRL*, 52:285–292, 2005.

[70] X. Ye and C.E. Priebe. A graph-search based navigation algorithm for traversing a potentially hazardous area with disambiguation. *Internat. J. Oper. Res. and Information Sys.*, 1(3):14–27, 2010.

[71] X. Ye, D.E. Fishkind, and C.E. Priebe. Sensor information monotonicity in disambiguation protocols. *J. Oper. Res. Society*, 62(1):142–151, 2011.