Penalty-Based Algorithms for the Stochastic Obstacle Scene Problem

Vural Aksakalli, Ibrahim Ari
Istanbul Sehir University, Uskudar, Istanbul 34662, Turkey
{aksakalli@sehir.edu.tr, ibrahimari@std.sehir.edu.tr}

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 generalization framework encompassing these algorithms that uses penalty functions to guide the navigation in real time. 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.

Data, as supplemental material, are available at http://dx.doi.org/10.1287/ijoc.2013.0571.

Keywords: probabilistic path planning; stochastic dynamic programming; Markov decision process; Canadian traveler’s problem

History: Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms; received October 2012; revised January 2013, June 2013; accepted July 2013. Published online in Articles in Advance February 11, 2014.

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 that 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 Papadimitriou and Yannakakis (1991). 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 (Blei and Kaelbling 1999, Ferguson et al. 2004, Likhachev et al. 2005), minefield countermeasures (Smith 1995, Witherspoon et al. 1995), and adaptive traffic routing (Fawcett and Robinson 2000, Gao and Chabini 2006). In fact, both problems as well as closely related ones have gained considerable attention recently—see, e.g., Nikolova and Karger (2008), Eyerich et al. (2009), Likhachev and Stentz (2009), Xu et al. (2009), Aksakalli and Ceyhan (2012).

There are no efficiently computable optimal policies known for SOSP or CTP and many similar problems have shown to be intractable (Papadimitriou and Yannakakis 1991, Provan 2003). The fundamental difficulty in obtaining a tractable model, even in the discrete setting, is that 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 Aksakalli et al. (2011) 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 Fishkind et al. (2007) and the reset disambiguation algorithm (RDA)
of Aksakalli et al. (2011). 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 SOS problem 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 paper 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 avenues with fixed policies within the avenues. 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 real time. 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 to illustrate our algorithms. Our experiments indicate that DTA provides near-optimal results with minimal computational resources.

Our presentation of the algorithms involves 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 SOSPs on arbitrary graphs in a relatively straightforward manner.

The rest of this paper is organized as follows: §2 discusses the challenges associated with the continuous version of the problem and illustrates the lattice discretization. Section 3 formally defines the continuous, discrete, and (lattice) discretized SOSP. Section 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 5 and 6 review SRA and RDA, respectively, and present their adaptations to discrete and discretized SOSP. Section 7 generalizes these two algorithms as penalty-based navigation strategies and introduces the DT algorithm. Section 8 presents computational experiments that compare the performance of DTA against SRA and RDA. Summary and conclusions are presented in §9.

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

The SOSP 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 possible-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, the online supplement (available as supplemental material at http://dx.doi.org/10.1287/ijoc.2013.0571) to this article 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, the online supplement illustrates via an example with two disks that the optimal disambiguation point of a particular disk not only depends 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.

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 a simple SOSP instance with two disks is shown in Figure 1 where 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.

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 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.
3. Definition of the Stochastic Obstacle Scene Problem

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

3.1. Continuous SOSP

Without loss of generality, we shall consider SOSPs 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 \( \rho_T: X_T \rightarrow (0, 1] \) and \( \rho_F: X_F \rightarrow (0, 1] \). When observing a realization of this process, the agent only sees \( X := X_T \cup X_F \), and \( \rho := \rho_T \cup \rho_F \). We assume that, for all \( x \in X \), \( \rho(x) \) is the probability that \( x \in X_T \). We also assume that whether 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 \( r: X \rightarrow \mathbb{R}_{\geq 0} \).

For any \( x \in X \), the probability \( \rho(x) \) shall be referred to as the “mark” of the associated disk \( D_x \). That is, the 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 \( (\bigcup_{x \in X} D_x)^C \) of shortest achievable arc length (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 \( \partial D_x \), the agent has the option to disambiguate \( x \), that is, learn if \( x \in X_T \). For a given cost function \( c: X \rightarrow \mathbb{R}_{\geq 0} \), 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.

3.2. Discrete SOSP

The discrete analogue of the previous problem, which we call the discrete SOSP, is defined as follows: Let \( G = (V, E) \) be an undirected graph with designated vertices \( s, t \in V \), and suppose there is a function \( l: E \rightarrow \mathbb{R}_{\geq 0} \) assigning a length to each edge; the goal here is to find a shortest \( s, t \) traversal (walk) in \( G \). However, not all of the edges may indeed be traversable. In particular, for a given subset \( E' \subseteq E \) of edges, called stochastic edges, there is a function \( \rho: E' \rightarrow [0, 1] \) such that, for each edge \( e \in E' \), \( \rho(e) \) is the probability that \( e \) is not traversable, independent of the other edges.

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 \( \rho \). Edges in \( E \setminus E' \) are deterministic in the sense that they are known a priori to be traversable. For any edge \( e \in E' \), 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 \( c(e) \) being added to the length of the traversal, for some function \( c: E' \rightarrow \mathbb{R}_{\geq 0} \). Edges cannot be traversed until it is known that they are traversable, and the traversability status of each edge is static and will never change over the course of the traversal. Of course, if the 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 \( \rho \)). 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).

Figure 1  Lattice Discretization of a Simple SOSP Instance with Two Disks

Note. Edges intersecting the disks are shown in bold.
To avoid infinite expected length, we assume the existence of a (possibly very long) s, t path consisting of edges from \( e \in E : \rho(e) = 0 \) or \( (E \setminus E) \).

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 the pairs of integers \( i, j \) such that \( 1 \leq i \leq i_{\text{max}} \) and \( 1 \leq j \leq j_{\text{max}} \), where \( i_{\text{max}} \) and \( j_{\text{max}} \) 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, 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.

4. A Polynomial Algorithm for Discrete SOSP with Parallel Avenues

The discrete SOSP has been shown to be NP-hard (Provan 2003). In this section, however, we present a polynomial algorithm when the problem is restricted to graphs consisting of parallel avenues with fixed policies within the avenues.

4.1. Discrete SOSP on Parallel Graphs

We call a graph \( G = (V, E) \) parallel if \( V = \{s, t\} \) 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 following remark 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. (2010).

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

\[
h(e) = l(e) + \frac{c(e)}{1 - \rho(e)}
\]

for all \( e \in E \) is optimal.

4.2. Discrete SOSP with Parallel Avenues

We now extend the previous method to the case where the associated graph consists of nonoverlapping parallel avenues \( p_1, p_2, \ldots, p_n \) between \( s \) and \( t \). Suppose that for each one of these avenues, there exists a policy that specifies the actions of the agent under any circumstance and at any possible location within the avenue. Also suppose that if an avenue is found to be untraversable, it will never be taken again; the agent will return to \( s \) and try another avenue. The agent will repeat this process until the destination is reached. We shall call this problem discrete SOSP with parallel avenues, and denote it by \( [p_1, p_2, \ldots, p_n] \). The following remark presents an optimal policy for this problem.

**Remark 2.** In discrete SOSP with parallel avenues \( [p_1, p_2, \ldots, p_n] \), let \( a_i \) denote the expected traversal length of \( p_i \) conditioned on reaching \( t \), \( b_i \) denote the expected traversal length of \( p_i \) conditioned on traversing back to \( s \), and \( \rho_i \) denote the probability that \( p_i \) is untraversable. The optimal policy for \( [p_1, p_2, \ldots, p_n] \) is generated via ordering the avenues by \( h_i := a_i - b_i + b_i/(1 - \rho_i) \). That is, the optimal policy is to traverse the avenues in increasing \( h_i \), where if the current avenue is found to be untraversable at some point, the agent traverses back to \( s \) and starts traversing the next avenue—until arrival at the destination.

We now illustrate an application of this result on a simple discrete SOSP instance shown in Figure 2. The associated graph in this instance has three parallel avenues: avenue \( p_1 \) that consists of edges \( e_1 \) and \( e_2 \), avenue \( p_2 \) that consists of the edge \( e_3 \), and avenue \( p_3 \) that consists of edges \( e_4, e_5, e_6 \), and \( e_7 \). The cost of disambiguation for each edge is taken as 1 and the mark of each edge is taken as 0.3. Edge lengths are shown next to the edges in the figure.

**Figure 2.** Illustration of a Discrete SOSP Instance with Three Parallel Avenues.
We assume that a depth-first traversal strategy is adopted within each avenue, which implies that the two parallel edges, \( e_s \) and \( e_t \), are an instance of the discrete SOSP on parallel graphs. Application of the theorem yields that, if \( e_s \) is disambiguated and found to be traversable, first \( e_s \) will be disambiguated, as it has a lower \( h_l \) value. For each avenue, \( \rho_i, a_i, b_i, h'_i \) and \( h'_i \) are calculated as follows:

First Avenue: \( \rho_1 = 1 - 0.7 \cdot 0.7 = 0.51; \ a_1 = 1 + 6 + 1 + 7 = 15; \ b_1 = (0.3/0.51) \cdot (1 + 0.3 \cdot 0.7/0.51) \cdot (1 + 2 \cdot 6 + 1) = 6.353; \ h'_1 = 15 - 6.353 = 6.353/0.49 = 21.61. \)

Second Avenue: \( \rho_2 = 0.3; \ a_2 = 1 + 15 = 16; \ b_2 = 1; \ h'_2 = 16 - 1 + 0.7/16.43. \)

Third Avenue: \( \rho_3 = 1 - 0.7 \cdot (1 - 0.3 \cdot 0.3) \cdot 0.7 = 0.5541; \ a_3 = 1 + 5 + 1 + 0.3 + 1 + 0.3 \cdot 0.7/0.51 \cdot 0.3 + 1 + 6 = 16.71; \ b_3 = (0.3/0.5541) \cdot (1 + 0.7 \cdot 0.3 - 0.3/0.5541) \cdot (1 + 2 + 5 + 1 + 0.3 - 1 + 0.7 \cdot 0.3 - 0.3/0.5541) \cdot (1 + 2 + 5 + 1 + 0.3 - 1 + (0.3/0.51) \cdot 2 + (0.3 \cdot 0.7/0.51) \cdot 3 + 1) = 8.72; \ h'_3 = 16.71 - 8.72 + 8.72/0.4459 = 27.55. \)

Thus, the optimal policy is to first try avenue \( p_2 \), which is edge \( e_2 \) (since it has the lowest \( h'_l \) value), then avenue \( p_1 \), and finally avenue \( p_3 \).

### 5. Discrete Adaptation of the Simulated Risk Disambiguation Algorithm

This section adapts the simulated risk disambiguation algorithm (SRA) in Fishkind et al. (2007) introduced for continuous SOSP to discrete and lattice-discretized SOSP (an earlier version of this section’s research appeared in Aksakalli et al. 2006).

#### 5.1. Adaptation to Discrete SOSP

In our framework, the traversal never uses edges while they are still ambiguous or are known to be nontraversable. The key intuition behind SRA is for the sake of 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

\[
\bar{\ell}^l(W) := -\log \prod_{e \in W \cap E^l} (1 - \rho(e)).
\]

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 \( g : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R} \) that 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 \( g(\bar{\ell}^l(W), \bar{\ell}^l(W)) \) 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

\[
\phi_g := \arg \min_{s \to t \text{ walks } W} g(\bar{\ell}^l(W), \bar{\ell}^l(W)).
\]

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 \( \phi_g \) in this particular case, we just need to find a deterministic shortest \( s \to t \) path in \( G \) via, e.g., Dijkstra’s algorithm where each edge in \( E \) is weighted as follows:

\[
\bar{w}^\text{SRA}_{\alpha}(e) := \bar{\ell}^l(e) + 1_{e \in E^d} \cdot \alpha \log (1 - \rho(e))^{-1},
\]

where \( \bar{\ell}^l(e) \) is the edge’s Euclidean length (which is either 1 or \( \sqrt{2} \), and 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 \( \bar{w}^\text{SRA}_{\alpha} \). 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 \( E^d \). If \( e \) was discovered to be nontraversable, 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 \( p_{\alpha} \) the \( s, t \) walk traversed under SRA. Observe that \( p_{\alpha} \) is an \( s, t \)-walk-valued random variable, since its realization depends on the outcomes of the dictated disambiguations. We will denote by \( \mathbb{E}p_{\alpha} \) the expected length of this walk. In our implementation, the values of \( \alpha \) minimizing \( \mathbb{E}p_{\alpha} \) are computed numerically by evaluating \( \mathbb{E}p_{\alpha} \) for a mesh of \( \alpha \) values—starting at \( \alpha_{\text{min}} = 2 \) and incrementing successively by \( \alpha_{\text{mesh}} = 5 \) units until \( \alpha \) is large enough that no disambiguations are performed.

#### 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

\[
\bar{\ell}^l(W) := -\log \prod_{D_i \cap D_j \cap W \neq \emptyset} (1 - \rho_i).
\]

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:

\[
w_{LD}^{SRA}(e) := l_e^S(e) + \frac{1}{2} \sum_{i=1}^{|X|} \text{#comp}(e \setminus D_i) \cdot 1_{e \in D_i} \cdot (\alpha \log(1 - \rho_e)^{-1}),
\]

where \( \text{#comp}(\cdot) \) is the number of connected components of its argument. An illustration is shown in Figure 3 with corresponding edge weights given in Table 1.

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 a 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 \( s \rightarrow t \) 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.”

6. Discrete Adaptation of the Reset Disambiguation Algorithm

The reset disambiguation algorithm (RDA) introduced in Aksakalli et al. (2011) for the continuous SOSP 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 \( s \rightarrow t \) navigation is completed. In the reset variant, however, each time an edge \( e \in E' \) 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 nontraversable, 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 \( 1/(1 - \rho(e)) \). This indicates that under an optimal policy, the agent may view \( e \) as if it was deterministically traversable at a cost \( c(e)/(1 - \rho(e)) \). This cost is defined to be \( \infty \) if \( \rho(e) = 1 \) regardless of \( c(e) \), and it is in addition to the edge’s Euclidean length \( l(e) \). Thus, the optimal policy in the reset variant of discrete SOSP boils down to finding a deterministic \( s \rightarrow t \)

<table>
<thead>
<tr>
<th>Edge</th>
<th>Edge weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_1 )</td>
<td>( 1 - \alpha \log(1 - \rho_1) )</td>
</tr>
<tr>
<td>( e_2 )</td>
<td>( 1 )</td>
</tr>
<tr>
<td>( e_3 )</td>
<td>( 1 - \alpha(1/2)\log(1 - \rho_2) + \log(1 - \rho_3) )</td>
</tr>
<tr>
<td>( e_4 )</td>
<td>( 1 - \alpha(1/2)\log(1 - \rho_4) + 2\log(1 - \rho_4) )</td>
</tr>
</tbody>
</table>
path in $G$ where the edge weights are defined as follows:

$$w_D^{RDA}(e) := l^e(c) + 1_{e \in E'} \cdot \frac{c(e)}{1 - \rho(e)}.$$  \hfill (4)

The idea in the 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 nonreset problem) using the NDR navigation strategy. It is easy to see that adaptation of RDA for discretized SOSP can be achieved by using the following weight function under the NDR navigation strategy:

$$w_D^{RDA}(e) := l^e(c) + \frac{1}{2} \sum_{i=1}^{[X]} \#comp(e\setminus D_i) \cdot 1_{e \in D_i \neq \emptyset} \cdot \frac{c(e)}{1 - \rho(e)}.$$  \hfill (5)

Per Equation (4), 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 $l^e(c) + c(e)/(1 - \rho(e))$. 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 nonreset variant in the paradigm of the NDR navigation strategy. It should be noted that either interpretation of the RD algorithm stands as an interesting idea in the 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.

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 risky traversable. On the other hand, RDA is based on the idea of using the optimal weights of a reset variant in the original nonreset 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, although 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 later.

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”:

$$w_D^{RDA}(e) := l^e(c) + 1_{e \in E} \cdot F(c),$$  \hfill (6)

and for discretized SOSP as:

$$w_D^{RDA}(e) := l^e(c) + \frac{1}{2} \sum_{i=1}^{[X]} \#comp(e\setminus D_i) \cdot 1_{e \in D_i \neq \emptyset} \cdot F(e).$$  \hfill (7)

In SRA, the penalty function $F$ is specified as $F(SR)(e) := \alpha \log(1 - \rho(e))^{-1}$, whereas it is defined as $F_RD(e) := c(e)/(1 - \rho(e))$ 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 $w_D^{RDA}(e)$ for discrete SOSP and $w_D^{RDA}(e)$ 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 $\alpha$ parameter for improved performance. The best value of this parameter is essentially found by brute force. Thus, a clear advantage of RDA over SRA is the lack of a fine-tuning parameter that results in significant computational savings. Aksakalli et al. (2011) illustrates, via computational experiments, that performance of RDA is comparable to that of SRA, whereas the 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 results 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 data set, as discussed in §8, as well as synthetic data that possess similar characteristics to this minefield data set. 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 as follows:

$$F_{DT}(e) := c(e) + \left( \frac{d_t(e)}{1 - \rho(e)} \right)^{-\log(1 - \rho(e))}.$$  \hfill (8)

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 following weight for discrete SOSP:

$$w_{DTA}^D(e) := 1^{\Phi}(e) + \sum_{t \in E} \left( c(e) + \left( \frac{d_t(e)}{1 - \rho(e)} \right)^{-\log(1 - \rho(e))} \right),$$ \hfill (9)

and the weight for discretized SOSP:

$$w_{DTA}^{\Phi}(e) := 1^{\Phi}(e) + \frac{1}{2} \sum_{i=1}^{X} \# \text{comp}(e \cap D_i) \cdot 1_{e \cap D_i \neq \emptyset} \left( c(e) + \left( \frac{d_t(e)}{1 - \rho(e)} \right)^{-\log(1 - \rho(e))} \right).$$ \hfill (10)

### 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 1, this time taking disk radii as 4.5 nondiagonal 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, which stands for $AO^*$ with bounds. Introduced in Aksakalli (2007), $BAO^*$ 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 fewer computational resources compared to $AO^*$ and stochastic dynamic programming. Superimposed walks as dictated by RDA are displayed in Figure 4(a). These walks are described next.

- 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 4(b) and explained as follows.

- 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 $D, E, F$ 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-unit decrease in the expected traversal length. Thus, in this particular case, SRA and DTA find the optimal policy while RDA yields a suboptimal one.
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 (Witherspoon et al. 1995, Muhandiramge 2008). A particular instance we consider is a United States Navy minefield data set (called the COBRA data) that first appeared in Witherspoon et al. (1995) and was later referred to in Priebe et al. (1997, 2005), Fishkind et al. (2007), Ye and Priebe (2010), Ye et al. (2011), and Aksakalli et al. (2011). The COBRA data is illustrated in Figure 5 and tabulated in Table 2. This data set 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 in this environment, it was conditioned that the zero-risk path length was at least 130 units. Here, the zero-risk path is defined as the shortest 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 §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 eight-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 SOSP 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 real time, 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 the excessive run times required for meshing of the \( \alpha \) 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 (2012) 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 seven 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.

### 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

<table>
<thead>
<tr>
<th>x-coord.</th>
<th>y-coord.</th>
<th>p</th>
<th>x-coord.</th>
<th>y-coord.</th>
<th>p</th>
<th>x-coord.</th>
<th>y-coord.</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>46.13</td>
<td>39.61</td>
<td>0.0731</td>
<td>50.49</td>
<td>24.26</td>
<td>0.1033</td>
<td>83.62</td>
<td>16.33</td>
<td>0.1165</td>
</tr>
<tr>
<td>30.21</td>
<td>54.62</td>
<td>0.1379</td>
<td>56.83</td>
<td>20.50</td>
<td>0.1527</td>
<td>44.87</td>
<td>66.45</td>
<td>0.1668</td>
</tr>
<tr>
<td>47.88</td>
<td>34.51</td>
<td>0.1718</td>
<td>40.55</td>
<td>76.93</td>
<td>0.1939</td>
<td>43.43</td>
<td>26.22</td>
<td>0.2575</td>
</tr>
<tr>
<td>21.93</td>
<td>53.22</td>
<td>0.3309</td>
<td>69.82</td>
<td>51.65</td>
<td>0.4353</td>
<td>65.64</td>
<td>11.08</td>
<td>0.4412</td>
</tr>
<tr>
<td>37.36</td>
<td>29.94</td>
<td>0.4917</td>
<td>29.47</td>
<td>37.21</td>
<td>0.5215</td>
<td>59.42</td>
<td>20.11</td>
<td>0.5418</td>
</tr>
<tr>
<td>38.90</td>
<td>57.22</td>
<td>0.5609</td>
<td>32.07</td>
<td>31.37</td>
<td>0.5745</td>
<td>45.71</td>
<td>24.83</td>
<td>0.5831</td>
</tr>
<tr>
<td>86.12</td>
<td>15.83</td>
<td>0.5902</td>
<td>52.01</td>
<td>56.80</td>
<td>0.5994</td>
<td>41.14</td>
<td>27.41</td>
<td>0.6200</td>
</tr>
<tr>
<td>8.43</td>
<td>74.26</td>
<td>0.6399</td>
<td>37.00</td>
<td>43.89</td>
<td>0.6416</td>
<td>72.53</td>
<td>18.22</td>
<td>0.6527</td>
</tr>
<tr>
<td>22.98</td>
<td>40.29</td>
<td>0.6543</td>
<td>70.33</td>
<td>18.61</td>
<td>0.6564</td>
<td>29.78</td>
<td>32.15</td>
<td>0.6566</td>
</tr>
<tr>
<td>63.54</td>
<td>24.81</td>
<td>0.1887</td>
<td>64.04</td>
<td>37.65</td>
<td>0.5149</td>
<td>27.00</td>
<td>37.97</td>
<td>0.5280</td>
</tr>
<tr>
<td>46.07</td>
<td>71.00</td>
<td>0.5609</td>
<td>65.16</td>
<td>64.01</td>
<td>0.5653</td>
<td>37.36</td>
<td>18.03</td>
<td>0.6108</td>
</tr>
<tr>
<td>39.43</td>
<td>70.31</td>
<td>0.6171</td>
<td>75.51</td>
<td>42.83</td>
<td>0.6189</td>
<td>76.11</td>
<td>55.73</td>
<td>0.6405</td>
</tr>
<tr>
<td>38.29</td>
<td>44.20</td>
<td>0.6444</td>
<td>28.16</td>
<td>64.10</td>
<td>0.6567</td>
<td>64.53</td>
<td>50.98</td>
<td>0.8515</td>
</tr>
</tbody>
</table>

**Note.** Disks in the first nine rows are false obstacles, whereas the ones in the last four rows (shown in bold) are true obstacles.
BAO* algorithm. Comparison results are presented in Table 3. On a 3.8 GHz personal computer, execution time of both the RD and DT algorithms was 0.312 seconds per run on average, whereas that of the SR algorithm was 18.5 seconds per run. Total run time required for computation of the optimal policy in Table 3, on the other hand, was 11 days and 17 hours.

In the table, expected length of the optimal policy denoted by $E^\text{OPT}(c)$ for a disambiguation cost of $c$. The expected length of the policy corresponding to the best $\alpha$ value for SRA is denoted by $E^\text{SRA}(c)$, whereas expected lengths of the policies obtained by RDA and DTA are denoted by $E^\text{RDA}(c)$ and $E^\text{DTA}(c)$, respectively. Percent deviation of the expected lengths found by the suboptimal algorithms from that of the optimal policy is denoted by $\%DO(c)$ superscripted by the algorithm name. For instance, $\%DO^\text{SRA}(c)$ means increases as $c$ increases.

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

In comparison, for $K = 1$, $\%DO^\text{DTA}(c \geq 0)$ is below 0.2, whereas for $K = 2$, median $\%DO^\text{DTA}(c \geq 0)$ is merely 0.21. Also, maximum $\%DO^\text{DTA}(c \geq 1)$ is 1.5, whereas maximum $\%DO^\text{RDA}(c \geq 1)$ is significantly higher at 32.59. In addition, the difference between $\%DO^\text{RDA}(c \geq 1)$ and $\%DO^\text{SRA}(c \geq 1)$ 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 $c \geq 6$.

### 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 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 4. Let $E^\text{RDA}_\text{mean}(c)$ and $E^\text{RDA}_\text{std}(c)$ 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 $E^\text{DTA}_\text{mean} < E^\text{RDA}_\text{mean}$ and that $E^\text{DTA}_\text{std} < E^\text{RDA}_\text{std}$. Interestingly, the difference in the means increases as $c$ decreases.

We now digress briefly and consider how $E^\text{OPT}(c)$ changes if $c$ is increased by $\delta > 0$ units. For $K = 1$, if the optimal policy requires a disambiguation, then it holds that $E^\text{OPT}(c + \delta) = E^\text{OPT}(c) + \delta$, 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 $\delta$ units, then in the best possible scenario, it would hold that $E^\text{OPT}(c + \delta) = E^\text{OPT}(c) + \delta$ (this can also be shown by contradiction). However, $E^\text{OPT}(c) + \delta$ is merely a lower bound for $E^\text{OPT}(c + \delta)$. Appendix A provides a simple parallel graph example where the optimal expected length increases by 2.2 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-unit increase. We conjecture that for any discrete
Aksakalli and Ari: Algorithms for Stochastic Obstacle Scenes

Table 4 Environment B Simulation Results

<table>
<thead>
<tr>
<th>$K$</th>
<th>$c$</th>
<th>Zero-risk</th>
<th>$E_{RDA}^{OPT}(c)$</th>
<th>$E_{RDA}^{OPT}(c)$</th>
<th>$E_{RDA}^{OPT}(c)$</th>
<th>$E_{RDA}^{OPT}(c) - E_{RDA}^{OPT}(c)$</th>
<th>$E_{RDA}^{OPT}(c)$ - $E_{RDA}^{OPT}(c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>134.08</td>
<td>4.80</td>
<td>—</td>
<td>121.34</td>
<td>17.07</td>
<td>3.18</td>
</tr>
<tr>
<td>1</td>
<td>1.00</td>
<td>134.08</td>
<td>4.80</td>
<td>130.04</td>
<td>121.52</td>
<td>11.72</td>
<td>8.52</td>
</tr>
<tr>
<td>2.00</td>
<td>134.08</td>
<td>4.80</td>
<td>127.85</td>
<td>122.18</td>
<td>118.94</td>
<td>6.86</td>
<td>2.24</td>
</tr>
<tr>
<td>4.00</td>
<td>134.08</td>
<td>4.80</td>
<td>124.89</td>
<td>123.35</td>
<td>121.85</td>
<td>1.69</td>
<td>1.50</td>
</tr>
<tr>
<td>6.00</td>
<td>134.08</td>
<td>4.80</td>
<td>126.04</td>
<td>125.32</td>
<td>123.71</td>
<td>1.64</td>
<td>1.61</td>
</tr>
<tr>
<td>8.00</td>
<td>134.08</td>
<td>4.80</td>
<td>127.48</td>
<td>126.60</td>
<td>125.50</td>
<td>1.44</td>
<td>1.11</td>
</tr>
<tr>
<td>10.00</td>
<td>134.08</td>
<td>4.80</td>
<td>128.42</td>
<td>128.34</td>
<td>127.14</td>
<td>1.72</td>
<td>1.20</td>
</tr>
</tbody>
</table>

SOSP 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 $K = 1$, $E_{RDA}^{OPT}(1) \approx 130$, whereas $E_{RDA}^{OPT}(2) \approx 128$ and $E_{RDA}^{OPT}(4) \approx 125$. A similar behavior is exhibited for $K = 2$. The observation that $E_{RDA}^{OPT}$ decreases as the disambiguation cost increases (where, in fact, it should be the opposite) suggests the following: the penalty function $F_{RDA}(c) = c(1 - \rho(e))$ 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 $\%DORDA$ 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—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, $E_{RDA}^{OPT}$ strictly increases as $c$ increases. Thus, DTA does not seem to suffer from the limitation of RDA mentioned earlier. In addition, the median difference between $E_{RDA}^{OPT}$ and $E_{SRA}^{OPT}$ is merely 1.3 units for the entire data in Table 4.

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 (2012), centers of the true obstacles were sampled from a $V$-shaped obstacle placement window with a vertical width of 10 units. The 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 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

Table 5 Environment C Simulation Results

<table>
<thead>
<tr>
<th>$K$</th>
<th>$c$</th>
<th>Zero-risk</th>
<th>$A_{RDA}(c)$</th>
<th>$A_{DTA}(c)$</th>
<th>$A_{RDA}(c) - A_{DTA}(c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Std.</td>
<td>Mean</td>
<td>Std.</td>
</tr>
<tr>
<td>0</td>
<td>159.91</td>
<td>5.79</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>159.91</td>
<td>5.79</td>
<td>208.20</td>
<td>68.20</td>
<td>70</td>
</tr>
<tr>
<td>2</td>
<td>159.91</td>
<td>5.79</td>
<td>210.20</td>
<td>70.98</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>159.91</td>
<td>5.79</td>
<td>185.10</td>
<td>65.14</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>159.91</td>
<td>5.79</td>
<td>171.55</td>
<td>50.43</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>159.91</td>
<td>5.79</td>
<td>162.51</td>
<td>31.82</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>159.91</td>
<td>5.79</td>
<td>160.57</td>
<td>20.36</td>
<td>4</td>
</tr>
</tbody>
</table>

Notes. Actual traversal lengths of the policies obtained by the respective algorithms are denoted by $A(c)$ superscripted by the algorithm name. The #Exceed columns denote the number of instances for which the actual traversal lengths exceed the zero-risk path lengths.
actual status information of disks as the agent navigates and performs disambiguations in the obstacle field. The number of instances for which the actual traversal lengths exceed the zero-risk path lengths are shown in columns labeled “#Exceed.”

Similar to the simulation results in Environment B, we observe that $A_{\text{mean}}^{\text{RDA}}$ decreases as the disambiguation cost increases, this time even more drastically. For instance, $A_{\text{mean}}^{\text{RDA}}(1) \approx 208$, whereas $A_{\text{mean}}^{\text{RDA}}(10) \approx 161$. This indicates that performance of RDA deteriorates significantly for small $c$ in this particular simulation environment. In addition, $A_{\#\text{Exceed}}^{\text{RDA}}(1) = 70$ out of 100 instances. Likewise, $A_{\#\text{Exceed}}^{\text{RDA}}(2) = 64$ and $A_{\#\text{Exceed}}^{\text{RDA}}(4) = 32$, which are all relatively high values. On the other hand, $A_{\#\text{Exceed}}^{\text{DTA}}$ never exceeds 2 for any of the cost values considered. One other observation is that $A_{\text{mean}}^{\text{RDA}}$ always exceeds the corresponding zero-risk length mean, which essentially suggests that, on average, RDA does not provide any improvement over the zero-risk path in actual $s - t$ traversals. In contrast, $A_{\text{mean}}^{\text{DTA}}$ is always smaller than the corresponding zero-risk length mean, thereby providing the navigating agent a strict improvement over the zero-risk path on average. In fact, the difference between $A_{\text{mean}}^{\text{RDA}}$ and $A_{\text{mean}}^{\text{DTA}}$ can be as high as 62.76 units (for $c = 1$), although this difference reduces as the disambiguation cost increases. Regarding the standard deviations, $A_{\text{std}}^{\text{RDA}}$ is considerably smaller compared to $A_{\text{std}}^{\text{DTA}}$ for all the cost values considered. That is, in general, DTA provides substantially better policies compared to RDA on average (especially for smaller $c$) while having a much smaller standard deviation. Also in favor of DTA is the observation that $A_{\text{mean}}^{\text{DTA}}$ strictly increases as $c$ increases.

Illustrated in Figure 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 $A_{\text{mean}}^{\text{RDA}} = 204.54$ and $A_{\text{mean}}^{\text{DTA}} = 131.12$. 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 $t$.

**Figure 6** An Instance in Environment C and $s - t$ Traversals as Dictated by RDA and DTA, Respectively

### 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 study, 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 are 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 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.

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 real time. Within this framework, we introduce a new suboptimal algorithm called the DT algorithm that uses a new penalty function taking 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 data set 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 noncircular obstacle regions. Further research on instances with different characteristics is required 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 could be seen as a showcase of this framework using the DT penalty function on an important real-world variant of the problem.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/ijoc.2013.0571.

Acknowledgments

The authors thank Professor Elvan Ceyhan at Koc University and Professor Donniell E. Fishkind at Johns Hopkins University for several very helpful discussions. The authors thank the area editor Professor Karen Aardal, the associate editor, and the two anonymous reviewers for their invaluable comments and suggestions that greatly improved the content, presentation, and overall quality of this article. Problem instances used in this study can be found in the online supplement. This research was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) [Grant 111M541].

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 $l_1 = 1.55$, $l_2 = 3.97$, 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 $P_1 = \{e_1, e_1\}$ 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 where the optimal policies are marked with an asterisk for the respective costs:

- $E^{P_1}(2) = 2 + (1 - 0.55)(1.55) + 0.55(2 + (1 - 0.08)(3.97)) = 5.81$,
- $E^{P_1}(4) = 2 + (1 - 0.08)(3.97) + 0.08(2 + (1 - 0.55)(1.55)) = 5.87$,
- $E^{P_2}(4) = 4 + (1 - 0.55)(1.55) + 0.55(4 + (1 - 0.08)(3.97)) = 8.91$,
- $E^{P_2}(4) = 4 + (1 - 0.08)(3.97) + 0.08(4 + (1 - 0.55)(1.55)) = 8.03$.

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, $E^{P_1}(4) = 8.91 = E^{P_2}(2) + 3.1$. 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-unit increase.

References


