The Reset Disambiguation Policy for Navigating Stochastic Obstacle Fields

Vural Aksakalli,[1] Donniell E. Fishkind,[2] Carey E. Priebe,[2] Xugang Ye[2]

[1] *Department of Industrial Engineering, Istanbul Sehir University, Istanbul*

[2] *Department of Applied Mathematics and Statistics, The Johns Hopkins University, Baltimore, Maryland*

Received 31 July 2010; revised 16 January 2011; accepted 23 January 2011
DOI 10.1002/nav.20454
Published online 1 April 2011 in Wiley Online Library (wileyonlinelibrary.com).

Abstract: The problem we consider is a stochastic shortest path problem in the presence of a dynamic learning capability. Specifically, a spatial arrangement of possible obstacles needs to be traversed as swiftly as possible, and the status of the obstacles may be disambiguated (at a cost) en route. No efficiently computable optimal policy is known, and many similar problems have been proven intractable. In this article, we adapt a policy which is optimal for a related problem and prove that this policy is indeed also optimal for a restricted class of instances of our problem. Otherwise, this policy is generally suboptimal but, nonetheless, it is both effective and efficiently computable. Examples/simulations are provided in a mine countermeasures application. Of central use is the Tangent Arc Graph, a polynomially sized topological superimposition of exponentially many visibility graphs. © 2011 Wiley Periodicals, Inc. Naval Research Logistics 58: 389–399, 2011

Keywords: mine countermeasures; probabilistic path planning; random disambiguation path; tangent arc graph; markov decision process

1. THE DISAMBIGUATION PROBLEM

A disambiguation problem instance is a tuple $(s, t, \mathcal{A}, \rho, c)$, where s and t are points in $\mathbb{R}^2$, $\mathcal{A}$ is a finite set of open discs in $\mathbb{R}^2$, ρ is a function $\mathcal{A} \to (0, 1]$, and c is a function $\mathcal{A} \to \mathbb{R}_{\geq 0}$. An agent wants to traverse from s to t through $\mathbb{R}^2$, along a continuous curve which is as short as possible in the sense of arclength. However, the discs of $\mathcal{A}$ are potential obstacles; for each $A \in \mathcal{A}$, the probability that A is an obstacle is $\rho(A)$, independently from the other discs in $\mathcal{A}$. If $\rho(A) < 1$ then we say A is ambiguous and if $\rho(A) = 1$ then A is definitely an obstacle. The traversing agent cannot enter discs which are obstacles or ambiguous but, if and when the agent is located at the boundary ∂A for any $A \in \mathcal{A}$, the agent has the option to disambiguate the disc A at a cost $c(A)$ added to the traversal arclength, and the agent will learn whether or not A is actually an obstacle. The status of a disc will never change; if A is revealed to be an obstacle then the traversing agent may never enter A, and if A is not an obstacle then A may be entered anytime thereafter. The central issue is how to direct the agent's traversal to optimally utilize this disambiguation capability; that is, to find a policy for the agent which minimizes the expected length of the agent's s, t traversal.

Correspondence to: C.E. Priebe (cep@jhu.edu)

An example of a disambiguation problem instance is shown in Fig. 1; suppose the values of $\rho(A_i)$, for $i = 1, 2, 3, 4, 5$ are 0.6, 0.4, 0.9, 0.8, 0.7, and suppose $c(A_i) = 1.1$ for all i. One particular traversal policy is illustrated in Fig. 1; from s the agent proceeds to the red bullet labeled 1, at which point A_1 is disambiguated. If A_1 is traversable then the agent is to continue till the red bullet labeled 2, at which point A_2 is disambiguated. Then the agent is to proceed to t through A_2 or clockwise around A_2, according as A_2 is traversable or not. If A_1 was not traversable, then the agent was to continue till the red bullet labeled 3, at which point A_4 is disambiguated. If A_4 is traversable then the agent continues till the red bullet labeled 4, at which point A_5 is disambiguated. Then the agent continues to t through A_5 or counterclockwise around A_5, according as A_5 is traversable or not. If A_4 was not traversable then the agent was to continue to t counterclockwise around A_4 and A_5. Under this policy, the agent's s, t traversal is an s, t-curve-valued random variable which would be γ_1, γ_2, γ_3, γ_4 or γ_5 with respective probabilities (0.4)(0.6), (0.4)(0.4), (0.6)(0.2)(0.3), (0.6)(0.2)(0.7), or (0.6)(0.8). If the respective arclengths of γ_1, γ_2, γ_3, γ_4 or γ_5 are 12, 14, 13, 16, and 18, then the expected length of the agent's s, t traversal is $(0.4)(0.6)[12+2.2]+(0.4)(0.4)[14+2.2]+(0.6)(0.2)(0.3)[13+3.3]+(0.6)(0.2)(0.7)[16+3.3]+(0.6)(0.8)[18+2.2]$. This is only one example of a policy, and it may not be the best policy.

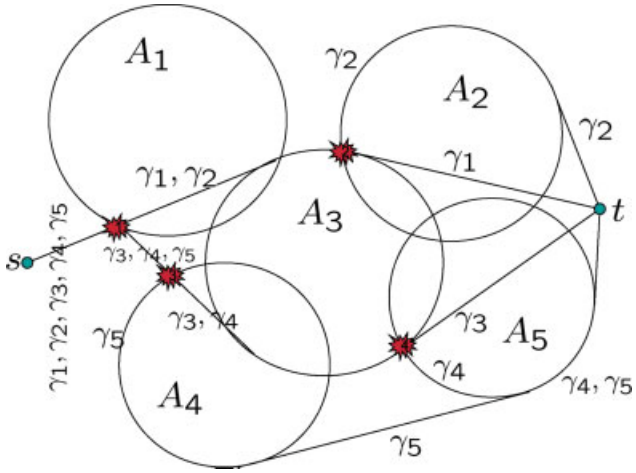

Figure 1. An example of a disambiguation problem instance and a policy for it.

More formally, a policy is a function which, to every possible disambiguation problem instance $(s, t, \mathcal{A}, \rho, c)$, assigns either an $A \in \mathcal{A}$ and an $x \in \partial A$ or else it assigns t; this is interpreted by an agent [with a disambiguation problem instance $(s, t, \mathcal{A}, \rho, c)$] as an instruction to proceed next from s to $x \in \partial A$ (or, instead, to proceed to t) going along the shortest s, x curve which avoids all discs of $\mathcal{A}$, and to then disambiguate A upon arrival at x. If A is revealed at that time to be an obstacle then $\rho(A)$ is set to 1, and if A is revealed to not be an obstacle then A is removed from $\mathcal{A}$; either way, the agent queries the policy again with x in place of s to determine the next disambiguation point, and so on iteratively until the agent reaches t.

Finding an optimal policy for a particular disambiguation problem instance is easily formulated as a Markov decision process but, unfortunately, it does not seem possible to avoid an intractably large state space in such a Markov decision process formulation, as we discuss next in Section 2.

In Section 3, we introduce a new policy called the Reset policy; it is an optimal policy for an altered problem setting described in that section. We then show in Section 5 that the Reset policy is indeed an optimal policy for a restricted family of instances of our problem, called early layer arrangements. Although the Reset policy is suboptimal for our problem in general, we show in Section 4 that it compares very favorably with the Simulated Risk policy of [10] and requires substantially less computing resources.

2. STOCHASTIC SHORTEST PATH WITH DYNAMIC LEARNING IN THE LITERATURE

The problem which we described is a minor modification of the stochastic obstacle scene problem (SOSP) of Papadimitriou and Yannakakis [14], who also describe a discrete version of SOSP which they call the Canadian traveller's problem (CTP). Papadimitriou and Yannakakis prove the intractability of several variants of SOSP and CTP. (For more background on CTP see [5].) CTP itself is a special case of the stochastic shortest paths with recourse (SPR) problem of Andreatta and Romeo [3], who present a stochastic dynamic programming formulation for SPR and note its intractability. Polychronopoulos and Tsitsiklis [17] also present a stochastic dynamic programming formulation for SPR and then prove the intractability of several variants. Provan [22] proves that SPR is intractable even if the underlying graph is directed and acyclic. Indeed, Provan [22] remarks that "all known no-reset versions of the problem are NP-hard."

The underlying difficulty in obtaining a tractable stochastic dynamic programming formulation—even in the discrete setting—is that in order for the agent to consider any action at any location there is a need for the agent to take into account what the agent has learned about the status of all of the potential obstacles, and the exponentially many such possibilities need to be accordingly incorporated when constructing the state space.

Heuristics are suggested for CTP and SPR in [4, 6, 17], but they would not be applicable to our problem in this article without initially approximating and recasting our continuous setting to the setting of a finite graph, in which case the resolution of the discretization drives up the number of vertices and edges in the approximating graph. By contrast, the algorithm we propose here is polynomial-time solely in the number of discs $|\mathcal{A}|$.

Our problem is considered in [10, 21], and a heuristic in [10] called the Simulated Risk disambiguation policy (SR) is the only policy comparable with the Reset policy introduced in this article. In Section 4, we illustrate in a mine countermeasures path planning example and in many simulation experiments that the Reset policy introduced in this article indeed provides results as effective as the SR policy but with the benefit of using substantially less computing resources. Then, in Section 5, we prove the optimality for the Reset policy in a particular class of instances.

On a related note, there is a body of work in the literature on the Probabilistic Roadmap Method (PRM) (e.g., [7, 12]). Like our problem, they seek a path in an obstacle field. There is a "local planner" which maps local traversabilities, and these are combined into a global roadmap—nonconcurrently to the agent's actual traversal. When obstacles are found the agent has not yet departed. (Thus, in particular, the agent is not rerouted or charged the cost to get to the point where the obstacle is encountered.) Moreover, the primary goal of PRM is not speed of traversal. By contrast, in our problem, only the agent can perform disambiguations and only when the agent is at respective location of ambiguity, which fundamentally changes the problem and the nature of policies. Indeed, for us, the agent is considering where to go so as to get to the

destination quickly simultaneous to trying to position itself to learn the most useful information for achieving a quick traversal.

There is also a substantial body of literature on the Simultaneous Localization and Mapping (SLAM) methodology and paradigm. (See [8, 9] for an excellent overview.) In SLAM, the goal is having an agent map an unknown field and, at the same time, the agent is to keep track of its location within that map, all in the presence of noise in sensor readings and noise in the control. The beauty of the methodology is the symbiotic relationship between the tackling of the mapping problem and the tackling of the localization problem to overcome the influence of the noise. By contrast, we assume perfect knowledge of the agent's location and control, and perfect knowledge of the gross features of the field (just excluding the status of potential obstacles), and we are only concerned with navigating the agent from start to destination as quickly as possible.

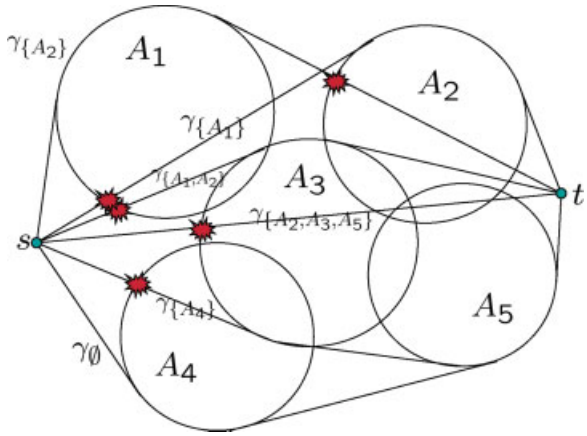

Figure 2. Some of the curves $\gamma_{\mathcal{B}} : \mathcal{B} \subseteq \mathcal{A}$. Many of the $2^5 = 32$ such curves are not shown, and many are not distinct; for example, $\gamma_{\{A_2,A_3,A_5\}} = \gamma_{\{A_1,A_2,A_3,A_5\}} = \gamma_{\{A_2,A_3,A_4,A_5\}} = \gamma_{\{A_1,A_2,A_3,A_4,A_5\}}$.

3. THE RESET POLICY

For the sole purpose of developing a policy for our problem of interest, we consider—just in this section, Section 3—a modification of our problem setting called the Reset setting. It is a simple but fundamental modification, similar to the "recoverable" variant of CTP in [5] and the "reset" variant of CTP in [22], and it admits a tractable solution. The optimal policy in this altered setting may then be extended for use in our problem setting of interest.

Suppose we are given a disambiguation problem instance $(s, t, \mathcal{A}, \rho, c)$. The Reset setting just differs in the following way: For each $A \in \mathcal{A}$, the status of A as an obstacle is not fixed over time as we assume in the rest of this article; in the Reset setting we assume that independent Bernoulli trials govern whether or not A is traversable at the times of different disambiguation queries of A, with probability $\rho(A)$ that A is indeed an obstacle at the time of each query. If at a given moment a disambiguation of A determines that A is not an obstacle, then the agent may enter A immediately, and A remains traversable until the agents exits A. Otherwise, almost immediately after each disambiguation of A, the status of A is "reset" and A becomes ambiguous again.

We can easily find an optimal policy in this Reset setting by observing one important property that, without loss of generality, an optimal policy will have: Namely, if an optimal policy dictates at any time that $A \in \mathcal{A}$ be disambiguated at $x \in \partial A$, and if the disambiguation finds that A is an obstacle then, without loss of generality (by appealing to Bellman's Principle of Optimality), the optimal policy will dictate that A be disambiguated again. This is because, with the resetting of A, the current state the agent is in is identical to the agent's state right before the first disambiguation of A, when the policy dictated that the disambiguation of A was to be performed. The implication is that disambiguations of A are to be repeated until A is traversable, and the number of disambiguations needed is thus a geometric random variable with expected value $\frac{1}{1-\rho(A)}$. This means that under an optimal policy the agent may view A as if it was deterministically traversable at a cost $\frac{c(A)}{1-\rho(A)}$ (we will say this has the value ∞ when $\rho(A) = 1$, regardless of $c(A)$). For each $\mathcal{B} \subseteq \mathcal{A}$, let $\gamma_{\mathcal{B}}$ denote the shortest s, t path in $\mathbb{R}^2 \backslash [\cup_{A \in \mathcal{B}^c} A]$ (see the illustration in Fig. 2) and let $\ell(\gamma_{\mathcal{B}})$ denote its arclength; the optimal policy is, quite simply, to just follow $\gamma_{\mathcal{B}^*}$ from s to t disambiguating discs as necessary, where $\mathcal{B}^* = \arg\min_{\mathcal{B} \subseteq \mathcal{A}} [\ell(\gamma_{\mathcal{B}}) + \sum_{A \in \mathcal{B}} \frac{c(A)}{1-\rho(A)}]$. Although this minimization is over a set of exponential size in $|\mathcal{A}|$, we will soon see (in Section 3.1) that this minimization can indeed be accomplished in polynomial time.

Returning to our setting of interest, disambiguation problem instances without reset: To each instance $(s, t, \mathcal{A}, \rho, c)$ the Reset policy assigns the first disc $A \in \mathcal{B}^*$ penetrated by $\gamma_{\mathcal{B}^*}$ and the point $x \in \partial A$ where $\gamma_{\mathcal{B}^*}$ penetrates A, with $\mathcal{B}^*$ defined as $\arg\min_{\mathcal{B} \subseteq \mathcal{A}} [\ell(\gamma_{\mathcal{B}}) + \sum_{A \in \mathcal{B}} \frac{c(A)}{1-\rho(A)}]$. (See the illustration in Fig. 2 where points of penetration for each of the curves shown are indicated by red bullets.) If there is no penetration (for example, if $\gamma_{\mathcal{B}^*}$ turned out to be $\gamma_\emptyset$ in the illustration in Fig. 2) then the Reset policy assigns t. We will see next in Section 3.1 that $\gamma_{\mathcal{B}^*}$ can be computed in $O(|\mathcal{A}|^3 \log |\mathcal{A}|)$ operations using the tangent arc graph (TAG) introduced in [10].

It is important to re-emphasize that in our setting of interest (disambiguation problem instances without Reset), the policy just determines where to next disambiguate, and then the information must be updated as per the results of the disambiguation, at which point the policy is queried again. Thus, for example in Fig. 2, perhaps $\gamma_{\mathcal{B}^*}$ is γ_{A_2,A_3,A_5} here; this curve

would be followed till the red bullet, but perhaps A_3 is then discovered to be an obstacle, in which case the subsequent policy query will redirect the traversal away from continuing on γ_{A_2,A_3,A_5}.

3.1. The Tangent Arc Graph

In this section, we describe the construction of the tangent arc graph (TAG) introduced in [10], given disambiguation problem instance $(s,t,\mathcal{A},\rho,c)$. The purpose of TAG here is to provide an efficient way to compute $\gamma_{\mathcal{B}^*}$ as defined in the immediately preceding section.

For any distinct points $x,y \in \{s,t\} \cup (\bigcup_{A\in\mathcal{A}} \partial A)$, we say the closed line segment $\overline{x,y}$ is a tangent segment provided that, for all $z \in \{x,y\}\backslash\{s,t\}$, $\overline{x,y}$ is tangential at z to A for some $A \in \mathcal{A}$. The vertex set of TAG consists of s, t, all points of intersection between any tangent segment and any ∂A (over all $A \in \mathcal{A}$), and all points of intersection between two or more ∂A's. The edge set of TAG consists of all connected components of all tangent segments after the vertices of TAG are removed, and all connected components of $\cup_{A\in\mathcal{A}}\partial A$ after the vertices of TAG are removed. An example of a TAG is pictured in Fig. 3. (It is the TAG associated with the example illustrated in Fig. 1.) Each edge of TAG is weighted with its arclength plus, for each of its endpoints and for each $A \in \mathcal{A}$, an additional $\frac{1}{2}\frac{c(A)}{1-\rho(A)}$ if the edge intersects A and the endpoint is on the boundary ∂A.

Note that TAG is a topological superimposition of all the (exponentially many) visibility graphs[1] generated by $s,t,\mathcal{B}^c$ over all choices of $\mathcal{B} \subseteq \mathcal{A}$, and note that for each A that is traversed there is $\frac{c(A)}{1-\rho(A)}$ added to Euclidean traversal length; thus, a shortest s,t path in TAG is indeed $\gamma_{\mathcal{B}^*}$. Observe that there are only $O(|\mathcal{A}|^2)$ tangent segments, each intersecting $O(|\mathcal{A}|)$ discs, so we have $O(|\mathcal{A}|^3)$ vertices and $O(|\mathcal{A}|^3)$ edges in TAG. The number of operations to set up TAG is $O(|\mathcal{A}|^3 \log|\mathcal{A}|)$ (the log arises from a sorting operation) and the number of operations in a heap implementation of Dijkstra's Algorithm to locate a shortest path is also $O(|\mathcal{A}|^3 \log|\mathcal{A}|)$ (the log arises from the cost of each heap operation). Hence the total number of operations needed to find the desired shortest s,t path $\gamma_{\mathcal{B}^*}$ is $O(|\mathcal{A}|^3 \log|\mathcal{A}|)$. (See e.g., [1] for details and analysis of the heap implementation of Dijkstra's Algorithm.)

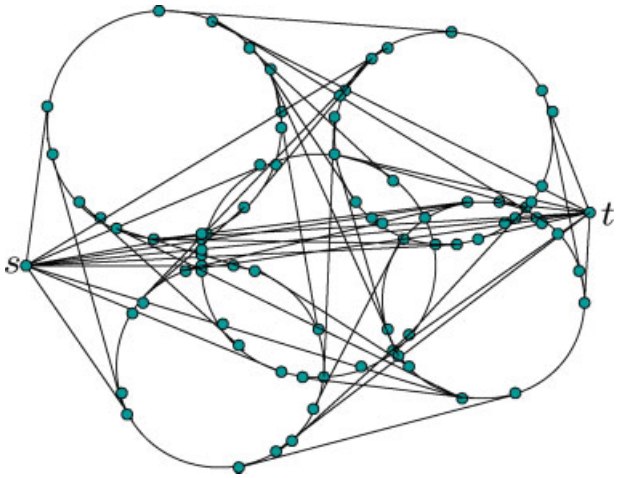

Figure 3. An example of a tangent arc graph (associated with the example in Fig. 1).

Although we use discs as the potential obstacles in the definition of our disambiguation problem, this choice is just a notational and computational convenience. Indeed, any collection of open, convex sets with smooth boundary can be used as the set of potential obstacles, and the definition and use of the tangent arc graph and the Reset policy are unchanged, but practically there would need to be a way to efficiently compute tangencies between pairs of such potential obstacles. Also, even if the boundaries are not smooth there may be efficiently computable adaptations to modify TAG and maintain its purpose; for example, in the case of polygons, their extreme points may take on the role of points of tangency in the construction of TAG, although the number of vertices and edges in TAG will go up as the number of extreme points goes up.

4. COMPUTATIONAL EXPERIMENTS

As mentioned in Section 2, the only policy comparable with the Reset policy for disambiguation problem instances (without reset) is the Simulated Risk policy proposed in [10]. We now show empirically that the Reset policy yields results comparable in quality to the Simulated Risk policy but with substantially less computational expense.

Minefield detection and localization is an important problem currently receiving much attention in the scientific and

[1] Given two points s,t and a set of obstacles in the plane $\mathbf{R}^2$ (usually the obstacles are open disks, polygonally bounded regions, or convex sets) a visibility graph is a simple graph consisting of some set of vertices on the boundary of the union-of-obstacles (s and t are also taken to be vertices) and edges between some pairs of these vertices such that the edges are either continuous curves on the boundary of the union-of-obstacles or are line segments that do not intersect obstacles. (Hence the term "visibility," indicating "unobstructed line of sight.") The visibility graph is further required to contain a path which is the shortest continuous s,t-curve avoiding all obstacles. In the case where the obstacles are open disks, it suffices to take as edges all of the unobstructed line segments whose both endpoints are (s or t or are) tangential to discs, and also to take as edges the segments of circle on the boundary of the union-of-obstacles, and the vertices are taken as the endpoints of these line segments and all of the locations on the boundary of the union-of-obstacles where disc boundaries intersect each other. This specific choice of visibility graph is often called "the" visibility graph, and indeed it does contains the shortest continuous s,t-curve avoiding all obstacles.

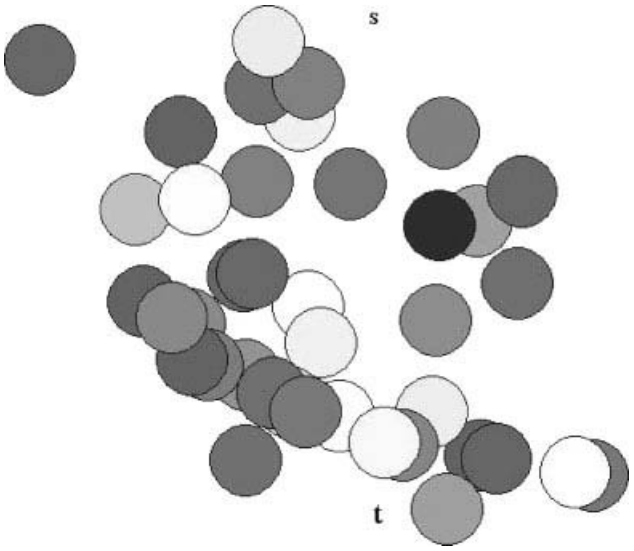

Figure 4. Mine countermeasures example, COBRA data. Gray-scale reflects $\rho(A)$ of discs.

engineering literature; see, for instance, [23] and the references cited there. Witherspoon et al. [24] depict the operational concept for minefield reconnaissance via unmanned aerial vehicles. Multispectral imagery of an area of interest is processed and a mine detection algorithm identifies locations of potential mines (see [11]), discs about these locations comprise $\mathcal{A}$. The marks $\rho(A) : A \in \mathcal{A}$, are posterior probabilities that the respective detections represent actual mines, as rendered by a postprocessing classification rule [13, 15, 16, 18, 20]. The following problem instance (called the COBRA data set) shown in Fig. 4 is referred to in [10, 13, 19, 21] and has 39 potential mines whose x- and y-coordinates are listed in Table 1.

The associated marks $\rho(A)$ in Table 1 were generated by the postprocessing classification rule in [13]. Each detection A has radius 50, s is the point $(0, 800)$, and t is the point $(0, 100)$. We assume the disambiguation cost c is the same for all detections, and we denote the expected length of the (s, t traversal curve associated with the) Reset policy by $E^{\text{reset}}(c)$, and the expected length of the (s, t traversal curve associated with the) Simulated Risk policy by $E^{\text{simrisk}}(c)$.

Although computing the Reset policy and realizing the associated s, t traversal can be done efficiently (as previously described), evaluating $E^{\text{reset}}(c)$ and $E^{\text{simrisk}}(c)$ is much more difficult since there are exponentially many s, t traversals that are possible to be the one that is realized, and we obtain the expected length by weighting them all by their respective probabilities. Thus, in this section, to evaluate $E^{\text{reset}}(c)$ and $E^{\text{simrisk}}(c)$, we will limit the s, t traversals to a maximum of 4 disambiguations, at which point the traversal must go next to t. The comparison between $E^{\text{reset}}(c)$ and $E^{\text{simrisk}}(c)$ is shown in Table 2; for all disambiguation costs $c > 1.1$ we found that $E^{\text{reset}}(c)$ and $E^{\text{simrisk}}(c)$ were either exactly equal or differed by at most about half a percent, and when $c < 1.1$ they could differ by 2.6 percent. Thus, the two policies are quite comparable here in quality, but the Reset policy is substantially less computationally intensive since the Simulated Risk policy requires solving an additional optimization problem to choose a parameter α that is used in its execution.

4.1. Other Simulations

To further compare the Reset policy and the Simulated Risk policy, random minefields were simulated similar to the setting above (the so-called COBRA data set) as follows; s was the point $(0, 800)$, t was the point $(0, 100)$, and 39 potential mines were randomly generated from the uniform distribution on the rectangle $[-400, 400] \times [150, 750]$. The radius of the discs $\mathcal{A}$ centered at the potential mines was 50, the disambiguation cost for each disc was $c = 25$, and the marks $\rho(A)$ were each randomly generated from the uniform distribution on the interval $[0, 1]$. To make the potential mines more formidable, we further conditioned (on the distribution

Table 1. Mine countermeasure example (COBRA data): $\rho(A)$ and x, y-coordinates of A's center, for all $A \in \mathcal{A}$.

x-coordinates	y-coordinates	$\rho(A)$	x-coordinates	y-coordinates	$\rho(A)$	x-coordinates	y-coordinates	$\rho(A)$
321.17	158.27	0.59017	54.23	201.12	0.54178	158.17	516.48	0.43525
215.13	428.31	0.61890	−145.67	703.06	0.61714	−151.01	572.15	0.56076
221.12	557.31	0.64047	−166.36	299.42	0.49173	296.16	163.31	0.11649
163.31	186.14	0.65636	28.31	205.03	0.15269	−79.26	709.99	0.56085
100.40	376.47	0.51487	−105.75	262.20	0.25748	185.31	182.18	0.65266
116.39	110.84	0.44124	−128.60	274.12	0.62001	−61.19	345.12	0.17183
−91.27	664.45	0.16675	−82.87	248.29	0.58308	105.47	509.80	0.85147
−19.93	568.04	0.59937	−310.23	402.92	0.65428	−320.73	532.23	0.33092
−35.11	242.61	0.10330	−169.99	438.90	0.64163	95.39	248.12	0.18868
−78.75	396.14	0.07310	−245.28	372.05	0.52154	−166.45	180.33	0.61082
−134.53	769.27	0.19386	−258.45	641.03	0.65670	111.60	640.10	0.56529
−219.32	313.68	0.57449	−455.72	742.57	0.63987	−157.10	441.96	0.64444
−242.22	321.51	0.65655	−237.86	546.19	0.13793	−269.98	379.65	0.52802

Table 2. Comparison of the Reset and Simulated Risk policies on a mine countermeasures example (COBRA data).

Cost interval I	$E^{\text{reset}}(c)$ for all $c \in I$	$E^{\text{simrisk}}(c)$ for all $c \in I$	$E^{\text{reset}}(c) - E^{\text{simrisk}}(c)$ for all $c \in I$	$\max_{c\in I} \lvert\frac{E^{\text{reset}}(c)-E^{\text{simrisk}}(c)}{E^{\text{simrisk}}(c)}\rvert$
(0.0, 1.1)	734.90+3.1033c	717.22+2.1665c	17.68+0.9368c	0.02600
(1.2, 4.1)	717.92+2.1665c	717.92+2.1665c	0	0
(4.2, 15.9)	721.60+1.2698c	720.89+1.2698c	0.71	0.00096
(16.0, 19.6)	721.60+1.2698c	722.92+1.1423c	−1.32+0.1275c	0.00158
(19.7, 49.5)	720.89+1.2698c	722.92+1.1423c	−2.03+0.1275c	0.00549
(49.6, 50.4)	722.92+1.1423c	722.92+1.1423c	0	0
(50.5, 68.9)	722.92+1.1423c	723.07+1.1393c	−0.15+0.0030c	0.00007
(69, 76.0)	723.07+1.1393c	723.07+1.1393c	0	0
(76.1, 116.4)	723.07+1.1393c	725.81+1.1033c	−2.74+0.0360c	0.00002
(116.5, 228.1)	725.81+1.1033c	725.81+1.1033c	0	0
(228.2, ∞)	977.54	977.54	0	0

of the mines) that the shortest s, t curve avoiding all regions $A \in \mathcal{A}$ should have length at least 1000; this is similar to the COBRA data where the shortest such s, t traversal had length 977.54.

In 100 such minefield simulations, the sample median, sample mean, and sample standard deviation of $\lvert\frac{E^{\text{reset}}(25)-E^{\text{simrisk}}(25)}{E^{\text{simrisk}}(25)}\rvert$ were, respectively, 0.008, 0.029, and 0.049. So, the quality of the Reset policy was comparable with the quality of the Simulated Risk policy, yet took much less computing resources; on a 3.5 Gigahertz computer, the mean running time to find the s, t traversal curve associated with the Reset policy was on the order of seconds, whereas the mean running time to find the s, t traversal curve associated with the Simulated Risk policy was about 60 times greater!

We did another 100 simulations with the only change being that $c = 35$; here the sample median, sample mean, and sample standard deviation of $\lvert\frac{E^{\text{reset}}(35)-E^{\text{simrisk}}(35)}{E^{\text{simrisk}}(35)}\rvert$ were, respectively, 0.004, 0.019, and 0.034. Again, the mean running time to find the s, t traversal curve associated with the Reset policy was on the order of seconds, whereas the mean running time to find the s, t traversal curve associated with the Simulated Risk policy was about 60 times greater.

4.2. Reset Policy Compared with Optimal Policy When Plane is Discretized

We would like to compare the performance of the Reset policy to the performance of the optimal policy but, unfortunately, the optimal policy does not seem to be practically computable except for the most trivial of disambiguation problem instances. However, if a disambiguation problem instance—which in this article is set in the (continuous) plane—has the plane restricted to a discrete subset of the plane then, in limited circumstances, the optimal policy in this restricted discretization may be practical to compute through brute force.

In this subsection, we return to the COBRA disambiguation problem instance from the beginning of Section 4 and perform this kind of discretization, and within it we compare the Reset policy to the optimal policy. The discretization is done as follows. For each pair of integers i, j, we declare the point $(20i, 20j)$ in the plane to be a vertex. Then, for each pair of integers i, j, the four line segments from $(20i, 20j)$ to $(20i + 20, 20j)$ of length 20, from $(20i, 20j)$ to $(20i, 20j + 20)$ of length 20, from $(20i, 20j)$ to $(20i + 20, 20j + 20)$ of length $20\sqrt{2}$, and from $(20i + 20, 20j)$ to $(20i, 20j + 20)$ of length $20\sqrt{2}$ are each declared to be edges. (Thus, every vertex is adjacent to eight edges; see the far-right picture in Fig. 5.) For this subsection only, we will restrict all traversals to be walks along these vertices and edges.

For the COBRA disambiguation problem instance from the beginning of Section 4, and for each of the costs $c = 15, 20, 25, 35, 50, 100, 200$ (and only allowing a maximum of two disambiguations), we computed the expected lengths of the s, t traversals—under the restriction that traversals may only use the vertices and edges of our discretization—for the Reset policy, for the Simulated Risk policy, and for the policy of minimum expected traversal length; their respective expected lengths are denoted $E^{\text{reset}}(c)$, $E^{\text{simrisk}}(c)$, and $E^{\text{optimal}}(c)$, and are shown in Table 3. (Also, see Fig. 5.) The computation of $E^{\text{optimal}}(c)$ was done by brute force, using the BAO* Algorithm of [2] to evaluate (or eliminate from consideration via pruning) all possible policies in this discrete setting, which resulted in the exact identification of the optimal policy. The computer running time to compute E^{optimal} for this table was approximately 40 hours, the running time to compute E^{simrisk} was approximately 80 seconds, and the running time to compute E^{reset} was about 3 seconds. Thus, in this example, the Reset policy is just about as effective as the optimal policy, but the Reset policy provides a substantial savings in computational expense.

5. OPTIMALITY OF RESET POLICY IN EARLY LAYER ARRANGEMENTS

The purpose of this section is to consider a special family of disambiguation problem instances (without reset) called

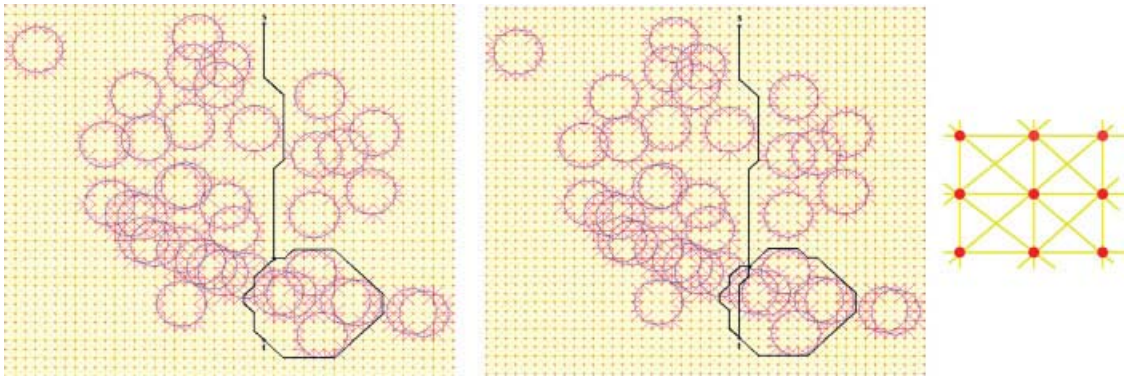

Figure 5. Superimposition of the possible s,t traversals in the COBRA data instance (when the plane is restricted to a discretization) generated by the optimal policy (left figure) and the Reset policy (right figure) when the disambiguation cost is $c = 35$. Traversal is black, vertices are red, edges are yellow, disk boundaries are blue, edges intersecting disc boundaries are red. A closeup of the discretization is in the far-right figure.

early layer arrangements and to prove that the Reset policy is optimal for these early layer arrangements. Indeed, this will form another motivation for the use of the Reset policy as a heuristic in more general instances. (In Section 3, we motivated the Reset policy by showing that it was optimal in a related, but fundamentally different setting wherein obstacles' ambiguities are continually reset. Then in Section 4, we showed empirically that the Reset policy is effective in our nonreset setting of interest. However, for general disambiguation problem instances in our setting of interest, the Reset policy is not actually the optimal policy. But in this section, we will indeed identify a family of disambiguation problem instances for which the Reset policy is actually optimal in our setting of interest.)

Before we define the early layer arrangement, we first review (next in Section 5.1) a basic stochastic decision problem which is ostensibly unrelated to the kinds of problems we have considered thus far in this article, but it will turn out in Section 5.2 to be quite relevant to early layer arrangements.

5.1. A Basic Stochastic Decision Problem

Consider the following basic stochastic decision problem, which it will be convenient to call the Single Stage Stochastic Decision Problem (SSSDP). An agent needs to accomplish a particular task and there is a set Ω of actions which the agent can perform sequentially in an effort to accomplish the task. For each action $a \in \Omega$, if a is performed then there is probability $\rho_a \in [0, 1)$ that a fails to accomplish the task and probability $1 - \rho_a$ that a successfully accomplishes the task (independently for the different actions), there is a cost $c_a \in \mathbb{R}$ charged for performing a, and if a successfully accomplishes the task then there is a cost $\ell_a \in \mathbb{R}$. At least one action $a \in \Omega$ has $\rho_a = 0$, so that the task can definitely be accomplished. A policy is a bijection $\pi : \Omega \to \{1, 2, \ldots, |\Omega|\}$ dictating an order for the actions to be attempted; action $\pi(1)$ will be attempted first, action $\pi(2)$ will be attempted next, and so on until the task is successfully accomplished. The agent's objective in SSSDP is to choose the policy π which minimizes the expected cost $f(\pi)$. Clearly,

$$f(\pi) = \sum_{i=1}^{|\Omega|} \left(\prod_{j<i} \rho_{\pi(j)} \right) (c_{\pi(i)} + (1 - \rho_{\pi(i)})\ell_{\pi(i)}). \quad (1)$$

Now, if π and π' are bijections $\Omega \to \{1, 2, \ldots, |\Omega|\}$ which are the same except that, for some index k, $\pi'(k+1) = \pi(k)$

Table 3. Comparison of the Reset, Simulated Risk, and Optimal policies when plane is restricted to a discretization, on a mine countermeasures example (COBRA data).

Cost c	$E^{\text{optimal}}(c)$	$E^{\text{simrisk}}(c)$	$E^{\text{reset}}(c)$	$\lvert\frac{E^{\text{simrisk}}(c)-E^{\text{optimal}}(c)}{E^{\text{optimal}}(c)}\rvert$	$\lvert\frac{E^{\text{reset}}(c)-E^{\text{optimal}}(c)}{E^{\text{optimal}}(c)}\rvert$
15	819.03	819.03	819.03	0	0
20	825.53	826.18	829.03	0.00079	0.0042
25	828.42	831.18	839.03	0.0033	0.013
35	841.18	841.18	847.71	0	0.0078
50	856.18	856.18	864.26	0	0.0094
100	906.18	906.18	919.43	0	0.015
200	1006.18	1006.18	1006.18	0	0

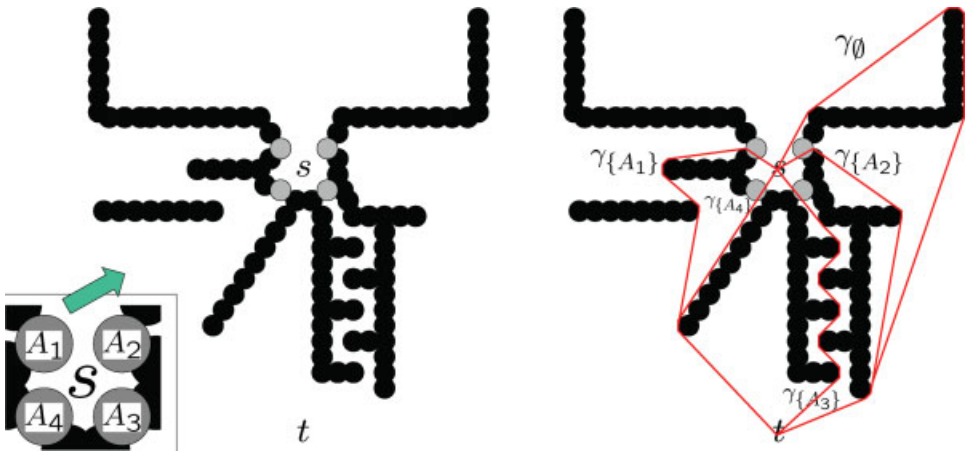

Figure 6. An early layer instance; gray discs A_1, A_2, A_3, A_4 are ambiguous and black discs are definitely obstacles. On the right (in red) are the shortest s,t paths avoiding all discs, except one (or zero) of the ambiguous discs. Distances from s to ambiguous discs are considered negligible here.

and $\pi'(k) = \pi(k+1)$, then elementary simplification yields that

$$f(\pi') - f(\pi) = \left(\prod_{j<k} \rho_{\pi(j)}\right)\left[\left(\ell_{\pi(k+1)} + \frac{c_{\pi(k+1)}}{1-\rho_{\pi(k+1)}}\right) - \left(\ell_{\pi(k)} + \frac{c_{\pi(k)}}{1-\rho_{\pi(k)}}\right)\right](1-\rho_{\pi(k)})(1-\rho_{\pi(k+1)}).$$

It then easily follows that the optimal policy for SSSDP—let us henceforth denote this policy with the letter Ψ—satisfies, for all $k = 1, 2, \ldots$, that $\ell_{\Psi(k)} + \frac{c_{\Psi(k)}}{1-\rho_{\Psi(k)}} \le \ell_{\Psi(k+1)} + \frac{c_{\Psi(k+1)}}{1-\rho_{\Psi(k+1)}}$; this can be seen by noting that f is monotonically nonincreasing during a bubble sort starting with any ordering of Ω when the sort is performed on the key $\ell_\cdot + \frac{c_\cdot}{1-\rho_\cdot}$ to order Ω in nondecreasing order of the key.

This basic stochastic decision problem and its straightforward solution will be of use next when we define and analyze the early layer arrangement.

5.2. Early Layer Arrangements

Let $(s,t,\mathcal{A},\rho,c)$ be a disambiguation problem instance. For any $\mathcal{B} \subseteq \mathcal{A}$, recall the definition of $\gamma_{\mathcal{B}}$ in Section 3; namely, $\gamma_{\mathcal{B}}$ denotes the shortest s,t path in $\mathbb{R}^2 \backslash [\cup_{A\in\mathcal{B}^c} A]$. Define Ω' to be the set of $A \in \mathcal{A}$ such that $\gamma_{\{A\}}$ intersects A and $\rho(A) < 1$. (For example, in Fig. 6, Ω' is the set of all four ambiguous discs $\{A_1, A_2, A_3, A_4\}$.) An early layer arrangement is a disambiguation problem instance in which two things happen: First of all, for each $A \in \Omega'$, the arclength of the segment-of-$\gamma_{\{A\}}$-from-s-until-$\gamma_{\{A\}}$-intersects-A is negligible. Also, for each $\mathcal{B} \subseteq \{A \in \mathcal{A} : \rho(A) < 1\}$, it holds that $\gamma_{\mathcal{B}} = \gamma_{\{A'\}}$ for some $A' \in \Omega'$, or else $\gamma_{\mathcal{B}} = \gamma_{\emptyset}$. See Fig. 6 for an example of an early layer arrangement; in that example, for all $\mathcal{B} \subseteq \{A_1, A_2, A_3, A_4\}$ it indeed holds that $\gamma_{\mathcal{B}}$ is $\gamma_{\emptyset}$ or $\gamma_{\{A_i\}}$ for an $i = 1, 2, 3$, or 4 and, for the convenience of a visually clean and uncluttered example, we consider the distance from s to A_1, A_2, A_3, A_4 to be negligible relative to the distance from s to t.

Let us consider what the Reset policy would dictate for an early layer arrangement. First, it will be notationally convenient to define Ω to be Ω' with one additional member $\star$, and define $\gamma_{\{\star\}} \equiv \gamma_{\emptyset}$, $c(\star) \equiv 0$, and $\rho(\star) \equiv 0$. (This can be thought of, in effect, as a completely benign declaration that $p_{\emptyset}$ traverses a virtual disc $\star$ with cost of disambiguation 0 and probability 0 of being nontraversable.) Then, without loss of generality, suppose the members of Ω are indexed $A_1, A_2, \ldots, A_{|\Omega|}$ such that $\ell(\gamma_{\{A_1\}}) + \frac{c(A_1)}{1-\rho(A_1)} \le \ell(\gamma_{\{A_2\}}) + \frac{c(A_2)}{1-\rho(A_2)} \le \cdots \le \ell(\gamma_{\{A_{|\Omega|}\}}) + \frac{c(A_{|\Omega|})}{1-\rho(A_{|\Omega|})}$. Now, (by its definition in Section 3) the Reset policy dictates that the agent disambiguates A_1 and, if A_1 is traversable, the agent traverses $\gamma_{\{A_1\}}$; if A_1 is not traversable then agent would disambiguate A_2 and, if A_2 is traversable, the agent traverses $\gamma_{\{A_2\}}$; if A_2 is not traversable then the agent would disambiguate A_3 and…etc., until t is reached. (Because the distance from s to these points of disambiguation are assumed negligible, it is convenient and has negligible cost to have the agent return to s after every unsuccessful disambiguation.)

Our main result of this section is the following.

THEOREM 1: The Reset policy is optimal for all early layer arrangements.

The optimality of the Reset policy for early layer arrangements will follow from viewing these disambiguation problem instances as being within the paradigm of SSSDP in Section 5.1, and by observing that the Reset policy here corresponds precisely to the optimal policy Ψ for SSSDP in

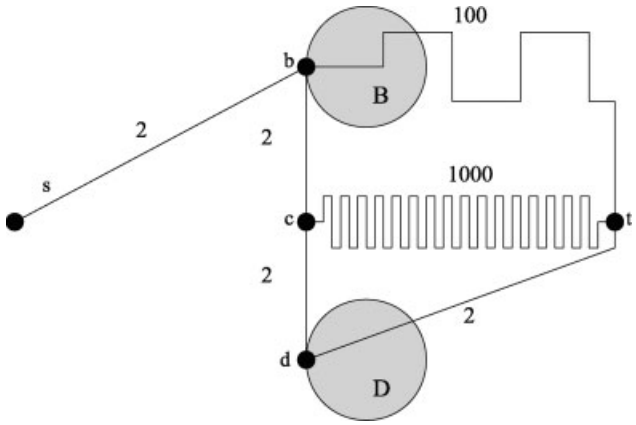

Figure 7. An example of a disambiguation problem instance where a balk is helpful.

Section 5.1. Indeed, Ω that we just defined in this section may be viewed as a set of actions which the agent may choose from; specifically, each $A \in \Omega$ may be considered as the action of disambiguating A, then traversing $\gamma_{\{A\}}$ from s to t if A is traversable (and if $A = \star$ then traversing $\gamma_{\emptyset}$ from s to t). The associated cost of performing action $A \in \Omega$ is $c(A)$, the cost if the action A is successful is $\ell(\gamma_{\{A\}})$, the probability that action A fails to accomplish the task of providing an s,t traversal is $\rho(A)$. Thus the Reset policy, in ranking actions and successively performing them in the order of $\ell_\cdot + \frac{c_\cdot}{1-\rho_\cdot}$ is precisely the optimal policy Ψ of Section 5.1.

To complete this proof of the optimality of the Reset policy for early layer arrangements, we need to be convinced that there is no loss of generality in restricting the agent's actions in an early layer arrangement to be the actions associated in the previous paragraph with Ω. Indeed, the definition of an early layer arrangement is designed to render this nonloss of generality, except for one single worry, which is described next in Section 5.3, and addressed after that in Section 5.4.

5.3. Balking

Sometimes in a disambiguation problem instance, it may be worthwhile to do a disambiguation and, even if the disambiguated disc is found to be traversable, it may nonetheless be worthwhile not to immediately enter the disc. We illustrate next with an example and then discuss the implications for the optimality of the Reset policy.

Consider a disambiguation problem instance for which many, many definite obstacles effectively block all movement of the agent, except for segment s,b, segment b,c, segment c,d, segment b,t, segment c,t, and segment d,t illustrated in Fig. 7, which have respective arclengths 2, 2, 2, 100, 1000, and 2. There are just two ambiguous discs B and D, and their boundaries intersect b and d, respectively, in the manner illustrated in Fig. 7, and suppose $\rho(B) = \rho(D) = \frac{1}{2}$ and $c(B) = c(D) = 0.1$.

Clearly, the agent starting at s should traverse segment s,b, then segment b,c, then segment c,d, and then should disambiguate D in the hopes of then being able to traverse segment d,t and complete the mission at a relatively low cost. However, it is easy to see that it would be advantageous for the agent to disambiguate B while at b—while on the way to d to disambiguate D, as described. Even though the agent would not immediately traverse segment b,t even if it turns out to be traversable (since a much better opportunity should yet be explored at d), nonetheless the information of whether segment b,t is traversable would be useful to obtain right away at the low cost of 0.1 on the chance that D is not traversable and B is not traversable, in which case the agent discovering at d that segment d,t is not traversable would be saved a significant backtracking cost to get to b to attempt traversing segment b,t (when it would not help anyway, and segment c,t is needed to complete the mission). A *balk* is such a disambiguation of a disc wherein even if the disc is traversable the agent does not traverse into the disc immediately.

Note that the Reset policy will never balk. Indeed, an agent executing the Reset policy is just following a shortest path, which is still a shortest path if a potential obstacle A in its way is revealed to not be an obstacle (since this knowledge uniformly reduces the length of all paths through A in the amount of $\frac{c(A)}{1-\rho(A)}$). The fact that the Reset policy will never balk is a potential weakness in the policy, since the Reset policy could never be optimal in situations like that of Fig. 7 where an optimal policy would perform a balk.

Our primary purpose in this subsection, Section 5.3, is to ask the question of whether an optimal policy for an early layer arrangement is balk free. In other words, is it ever helpful in an early layer arrangement to disambiguate a detection and then not immediately follow through to t? In the next subsection, Section 5.4, we prove that the answer is no, balks will never be helpful in an early layer arrangement. Thus there is no loss of generality in limiting the agent's actions in an early layer arrangement to be Ω, wherein $A \in \Omega$ represents the action of disambiguating disc A and, if traversable, traversing $p_{\{A\}}$ to t. Note that establishing this fact will complete the proof that the Reset policy is optimal for early layer arrangements.

5.4. Balks will Never be Helpful in an Early Layer Arrangement

To show that balks will never be helpful in early layer arrangements, we next adjust SSSDP of Section 5.1, and call the adjusted problem GSSSDP (generalized SSSDP). It differs from SSSDP only in that we subdivide each of the actions $a \in \Omega$ into two actions, $\hat{a}$ and $\tilde{a}$ (we will henceforth refer to

a as a superaction); the action $\hat{a}$ just finds out at a cost of c_a whether or not a would successfully accomplish the agent's task and, as before, the probability that a would accomplish the task is ρ_a. Only after action $\hat{a}$ is performed—and assuming that a would successfully accomplish the agent's task—may the agent perform the separate action $\tilde{a}$, which executes a at a cost of ℓ_a and accomplishes the agent's task. Even if the agent performs $\hat{a}$ and learns that a could successfully accomplish the task, nonetheless the agent is not obligated to perform $\tilde{a}$, and may perform $\tilde{a}$ immediately, later, or never, as the agent wills.

We will call a policy simple if it has the property that, for all $a \in \Omega$, if $\hat{a}$ is performed and it is discovered that a could successfully accomplish the task then the agent is to next perform $\tilde{a}$ and indeed accomplish the task. Restricting GSSSDP to simple policies is exactly SSSDP, thus we may view policy Ψ as (also being) a GSSSDP policy which is at least as good as all other simple policies. In fact, we next show in Proposition 2 that, for any policy π for a GSSSDP instance, there exists a simple policy which is at least as good as π and, transitively, the policy Ψ is at least as good as π.

PROPOSITION 2: The policy Ψ is optimal for GSSSDP.

PROOF: Induction on $|\Omega|$; if $|\Omega| = 1$ then the result is trivial, so we next suppose the result is true for all GSSSDP instances where $|\Omega|$ is an arbitrarily chosen positive integer, and consider any particular GSSSDP instance with $|\Omega|$ one greater. Let π be any policy for this instance.

Without loss of generality suppose the members of Ω are indexed $a_1, a_2, \ldots, a_{|\Omega|}$ such that $\ell_{a_1} + \frac{c_{a_1}}{1-\rho_{a_1}} \leq \ell_{a_2} + \frac{c_{a_2}}{1-\rho_{a_2}} \leq \cdots \leq \ell_{a_{|\Omega|}} + \frac{c_{a_{|\Omega|}}}{1-\rho_{a_{|\Omega|}}}$. There exists a member of Ω, say a_m, such that $\rho_{a_m} = 0$ (as mentioned before, otherwise the expected cost is infinite). Say that the first action which π dictates that the agent perform is the action $\hat{a}_r$; without loss of generality $\rho_{a_r} \neq 0$ (in particular, $a_r \neq a_m$) because otherwise no useful information would be provided by $\hat{a}_r$ (we would already know for sure a_r could accomplish the agent's task) and the only meaningful use of performing $\hat{a}_r$ would be to clear the way to perform $\tilde{a}_r$ next, in which case π is a simple policy, no better than Ψ, and we would be done. Also without loss of generality, $\ell_{a_r} < \ell_{a_m} + \frac{c_{a_m}}{1-0} = \ell_{a_m} + \frac{c_{a_m}}{1-\rho_{a_m}}$, or else a_r could have been eliminated from Ω at the outset.

After $\hat{a}_r$ is performed, either it is revealed that a_r cannot accomplish the agent's task, or it is revealed that a_r could accomplish the agent's task. In the former case, a_r may be discarded from Ω, and this GSSSDP incidence is reduced to an instance where the induction hypothesis applies, and (without loss of generality, by appealing to Bellman's Principle of Optimality) π is henceforth the policy Ψ. Let us denote by X the expected cost henceforth under Ψ. In the latter case where it is revealed that a_r can accomplish the agent's task, a_r would stay in Ω but its disambiguation cost and its probability of nontraversability would both henceforth be 0. Thus we can discard a_m from Ω (both would definitely accomplish the task, but a_r at a lower cost) and this GSSSDP incidence is reduced to an instance where the induction hypothesis applies, and π is henceforth the policy Ψ. Let us denote by Y the expected cost henceforth under Ψ. Thus the expected cost of π overall is $c_{a_r} + \rho_{a_r} \cdot X + (1 - \rho_{a_r}) \cdot Y$.

Because X and Y are expected costs under policy Ψ, they are computed in the same manner as in Eq. (1) of Section 5.1, which yields that $X = \sum_{i=1, i\neq r}^{n} (\prod_{j:j<i, j\neq r} \rho_{a_j})(c_{a_i} + (1 - \rho_{a_i})\ell_{a_i})$ and $Y = \sum_{i=1}^{v} (\prod_{j:j<i} \rho_{a_j})(c_{a_i} + (1 - \rho_{a_i})\ell_{a_i}) + (\prod_{j=1}^{v} \rho_{a_j})(0 + (1 - 0)\ell_{a_r})$, where v is the maximum index such that $\ell_{a_v} + \frac{c_{a_v}}{1-\rho_{a_v}} < \ell_{a_r} + \frac{0}{1-0} = \ell_{a_r}$. Next, it is useful to break X into two separate summations $X = \sum_{i=1}^{v} (\prod_{j:j<i} \rho_{a_j})(c_{a_i} + (1 - \rho_{a_i})\ell_{a_i}) + \sum_{i=v+1, i\neq r}^{n} (\prod_{j:j<i, j\neq r} \rho_{a_j})(c_{a_i} + (1 - \rho_{a_i})\ell_{a_i})$, so that the expected cost of policy π can be simplified $c_{a_r} + \rho_{a_r} \cdot X + (1 - \rho_{a_r}) \cdot Y =$

$$
\begin{aligned}
&\left[\sum_{i=1}^{v}\left(\prod_{j:j<i}\rho_{a_j}\right)(c_{a_i}+(1-\rho_{a_i})\ell_{a_i})\right] \\
&\quad+\left[c_{a_r}+\left(\prod_{j=1}^{v}\rho_{a_j}\right)(1-\rho_{a_r})\ell_{a_r}\right] \\
&\quad+\left[\sum_{i=v+1,i\neq r}^{n}\left(\prod_{j:j<i,j\neq r}\rho_{a_j}\right)\rho_{a_r}(c_{a_i}+(1-\rho_{a_i})\ell_{a_i})\right] \\
&\geq\left[\sum_{i=1}^{v}\left(\prod_{j:j<i}\rho_{a_j}\right)(c_{a_i}+(1-\rho_{a_i})\ell_{a_i})\right] \\
&\quad+\left[\left(\prod_{j=1}^{v}\rho_{a_j}\right)(c_{a_r}+(1-\rho_{a_r})\ell_{a_r})\right] \\
&\quad+\left[\sum_{i=v+1,i\neq r}^{n}\left(\prod_{j:j<i,j\neq r}\rho_{a_j}\right)\rho_{a_r}(c_{a_i}+(1-\rho_{a_i})\ell_{a_i})\right]
\end{aligned}
$$

which is the expected cost (at the outset) of the simple policy Ψ-except-for-moving-a_r-to-between-a_v-and-a_{v+1}, which is at least the expected cost (at the outset) of the policy Ψ. Since policy π was arbitrary, Proposition 2 then follows by induction. □

With Proposition 2 proven, we have that balks will never be of use in an early layer arrangement, and we have established Theorem 1.

6. CONCLUSION AND FURTHER WORK

The Reset policy introduced in this article for our disambiguation problem is motivated by the fact that it is an optimal policy in a related setting, and it is an optimal policy for early layer arrangements, a particular class of instances of our disambiguation problem. The fact that the Reset policy is balk-free exposes a weakness in this policy; it can not be optimal for an instance where a balk is required. Nonetheless, we believe that articulating this weakness will pave the way for increasingly sophisticated second generation policies to follow the Reset policy. In general, the Reset policy is efficiently computable and seems to be an effective heuristic for a difficult problem.

ACKNOWLEDGMENTS

Research (of all of the authors) was supported by US Office of Naval Research, grant N000140610013. A large portion of this article is included in the PhD dissertation of Vural Aksakalli [2] while at Johns Hopkins University. The authors thank the anonymous referees for very valuable suggestions that greatly improved this article, and Kevin Byrnes for very helpful discussions.

REFERENCES

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network flows, Prentice Hall, USA, 1993.
[2] V. Aksakalli, Protocols for stochastic shortest path problems with dynamic learning, Ph.D. Dissertation, Johns Hopkins University, Baltimore, MD, 2007.
[3] G. Andreatta and L. Romeo, Stochastic shortest paths with recourse, Networks 18 (1988), 193–204.
[4] M. Baglietto, G. Battistelli, F. Vitali, and R. Zoppoli, Shortest path problems on stochastic graphs: A neuro dynamic programming approach, Proceedings of the 42nd IEEE Conference on Decision and Control, 2003, pp. 6187–6193.
[5] A. Bar-Noy and B. Schieber, The canadian traveller problem, SODA '91: Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms, 1991, pp. 261–270.
[6] D.M. Blei and L.P. Kaelbling, Shortest paths in a dynamic uncertain domain, IJCAI Workshop on Adaptive Spatial Representations of Dynamic Environments, 1991.
[7] R. Bohlin and L.E. Kavraki, Path planning using lazy PRM, Proc. IEEE International Conference on Robotics and Automation, 2000.
[8] H. Durrant-Whyte and T. Bailey, Simultaneous localization and mapping: Part I, IEEE Robotics and Automation Magazine, June 2006.
[9] H. Durrant-Whyte and T. Bailey, Simultaneous localization and mapping: Part II, IEEE Robotics and Automation Magazine, September 2006.
[10] D.E. Fishkind, C.E. Priebe, K. Giles, L.N. Smith, and V. Aksakalli, Disambiguation protocols based on risk simulation, IEEE Trans Syst Man Cybernet Part A 37 (2007), 814–823.
[11] Q.A. Holmes, C.R. Schwartz, J.H. Seldin, J.A. Wright, and L.J. Wieter, "Adaptive multispectral CFAR detection of land mines," Proceedings of the SPIE, Vol. 2496: Detection Technologies for Mines and Minelike Targets, Orlando, Florida, April 1995, pp. 421–432.
[12] L. Jaillet and T. Simeon, A PRM-based motion planner for dynamically changing environments, Proc. International Conference on Intelligent Robots and Systems, 2004.
[13] T. Olson, J.S. Pang, and C.E. Priebe, A likelihood-MPEC approach to target classification, Math Program Ser A 96 (2002), 1–31.
[14] C.H. Papadimitriou and M. Yannakakis, Shortest paths without a map, Theoret Comput Sci 84 (1991), 127–150.
[15] C.D. Piatko, C.P. Diehl, P. McNamee, C. Resch, and I.-J. Wang, Stochastic search and graph techniques for MCM path planning, Proceedings of the SPIE, Vol. 4742, 2002, pp. 583–592.
[16] C.D. Piatko, C.E. Priebe, L.J. Cowen, Wang I.-J., and P. McNamee, Path planning for mine countermeasures command and control, Proceedings of the SPIE, Vol. 4394, 2001, pp. 836–843.
[17] G.H. Polychronopoulos and J.N. Tsitsiklis, Stochastic shortest path problems with recourse, Networks 27 (1996), 133–143.
[18] C.E. Priebe, D.Q. Naiman, and L. Cope, Importance sampling for spatial scan analysis: Computing scan statistic p-values for marked point processes, Comput Statist Data Anal 35 (2001), 475–485.
[19] C.E. Priebe, T.E. Olson, and D.M. Healy Jr., Exploiting stochastic partitions for minefield detection, Proceedings of the SPIE, Vol. 3079, 1997, pp. 508–518.
[20] C.E. Priebe, J.S. Pang, and T. Olson, Optimizing sensor fusion for classification performance, Proceedings of the CISST '99 International Conference, Las Vegas, Nevada, June 1999, pp. 397–403.
[21] C.E. Priebe, D.E. Fishkind, L. Abrams, and C.D. Piatko, Random disambiguation paths, Nav Res Logist 52 (2005), 285–292.
[22] J.S. Provan, A polynomial-time algorithm to find shortest paths with recourse, Networks 42 (2003), 115–125.
[23] D.L. Smith, Detection technologies for mines and minelike targets, Proceedings of the SPIE, Vol. 2496: Detection Technologies for Mines and Minelike Targets, Orlando, Florida, April 1995, pp. 404–408.
[24] N.H. Witherspoon, J.H. Holloway, K.S. Davis, R.W. Miller, and A.C. Dubey, The coastal battlefield reconnaissance and analysis (COBRA) program for minefield detection, Proceedings of the SPIE Vol. 2496: Detection Technologies for Mines and Minelike Targets, Orlando, Florida, April 1995, pp. 500–508.