S.Ü. Müh. Bilim ve Tekn. Derg., c.7, s.1, ss. 89-104, 2019
Selcuk Univ. J. Eng. Sci. Tech., v.7, n.1, pp. 89-104, 2019
ISSN: 2147-9364 (Electronic)
DOI: 10.15317/Scitech.2019.184

APPROXIMATE DYNAMIC PROGRAMMING FOR OPTIMAL SEARCH WITH AN OBSTACLE

Yasin GÖÇGÜN
Altınbaş University, Faculty of Engineering and Natural Sciences, Department of Industrial Engineering, 34217, İstanbul, Turkey
yasin.gocgun@altinbas.edu.tr

(Geliş/Received: 30.04.2018; Kabul/Accepted in Revised Form: 08.08.2018)

ABSTRACT: In this paper, we study a class of optimal search problems where the search region includes a target and an obstacle, each of which has some shape. The search region is divided into small grid cells and the searcher examines one of those cells at each time period with the objective of finding the target with minimum expected cost. The searcher may either take an action that is quick but risky, or another one that is slow but safe, and incurs different cost for these actions. We formulate these problems as Markov Decision Processes (MDPs), but because of the intractability of the state space, we approximately solve the MDPs using an Approximate Dynamic Programming (ADP) technique and compare its performance against heuristic decision rules. Our numerical experiments reveal that the ADP technique outperforms heuristics on majority of problem instances. Specifically, the ADP technique performs better than the best heuristic policy in 17 out of 24 problem sets. The percent improvement in those 17 problem sets is on average 7.3%.

Key Words: *Approximate dynamic programming, Markov decision processes, Optimal search*

Engelli Optimal Arama için Yaklaşımsal Dinamik Programlama

ÖZ: Bu makalede her biri bir şekle sahip olan, arama bölgesi hedef ve engel içeren optimal arama problemlerini çalışmaktayız. Arama bölgesi küçük grid hücrelerine bölünmüştür ve araştırmacı her bir zaman peryodunda minimum maliyetle hedefi bulma amacıyla bu hücrelerden birini inceler. Araştırmacı hızlı ama riskli eylemi veya yavaş fakat güvenilir olanı seçebilir ve bu eylemler için farklı ücret öder. Bu problemleri Markov Karar Süreçleri (MKS) ile formüle etmekteyiz, fakat durum uzayının çetinliğinden dolayı MKS'leri bir Yaklaşımsal Dinamik Programlama (YDP) tekniği kullanarak yaklaşık olarak çözmekteyiz ve onun performansını sezgisel karar kurallarıyla karşılaştırmaktayız. Nümerik deneylerimiz problem örneklerinin çoğunda YDP tekniğinin sezgisel yöntemlerden üstün olduğunu ortaya çıkarmıştır. Spesifik olarak YDP tekniği 24 problem kümesinden 17'sinde en iyi sezgisel yöntemden daha iyi sonuç vermektedir. Bu 17 problem kümesindeki yüzde gelişme ortalama %7,3'tür.

Anahtar Kelimeler: *Markov karar süreçleri, Optimal arama, Yaklaşımsal dinamik programlama*

INTRODUCTION

In many search problems, a region of interest is examined to find a stationary or moving target, whose location is only known probabilistically. Optimality criteria typically include maximizing the probability of detecting the target subject to a constraint on search effort, or minimizing the expected cost until locating the target. Optimal search problems arise in diverse settings such as mobile robot missions (Derr and Manic, 2009), visual search and detection (Najemnik and Geisler, 2008), and event detection in sensor network (Wettergren, 2008). For example, large areas of the ocean may be searched in order to detect an enemy submarine in military applications, or to find someone lost at sea in search and rescue operations.

Our work is motivated by a much smaller-scale search that occurs during the course of minimally invasive surgery (MIS). While pre-operative images are certainly helpful, surgeons performing MIS still do not know exactly where certain critical structures lie at the start of the actual operation. Since real-time imaging is infeasible in many cases, surgeons must effectively use their surgical instruments to search through tissues and past other structures to locate a target organ of interest (e.g., the kidney) safely and efficiently.

Towards that end, we consider a class of optimal search problems where the search region contains a stationary target and a stationary obstacle, whose locations are known through a joint probability distribution. By "target" we mean the object that when found, ends the search problem and stream of costs. Note that while we use the term "obstacle", we do not mean that it is necessarily something to be avoided. In fact, it may serve as a landmark towards finding the target, and therefore search strategies may try to find the obstacle before the target. For example, in minimally-invasive kidney operations, surgeons typically first try to find the ureter, which they can then follow along until it connects to the kidney. We also consider that the two objects may have shapes and occupy more than one cell of the search grid (as is the case with organs in MIS). We assume that upon finding one cell containing part of the object, the searcher knows the precise location of the entire object. At each step, the searcher chooses which cell to examine next and which of two actions to take on that cell: a quick-but-risky or a slow-but-safe action. In the context of MIS, the analogy is a choice between quickly cutting through the tissue that obstructs the surgeon's view (risking cutting a critical structure just behind the tissue), or slowly peeling away layers of tissue to safely explore whether the structure is there or not. In the remainder of the paper, we shall refer to these two choices as "cut" or "explore." While our motivating example is MIS and we adopt some language particular to that setting, we note that the tradeoffs involved apply to other search problems where views are obstructed, paths must be cleared to find objects, and there is a concern about damaging them along the way. For example, this applies to archeological excavations as well.

We aim to determine an optimal search policy; that is, the sequence of cells to be visited, along with the search action to choose at each of those cells, so as to minimize the total expected cost until locating the target. We formulate this as a Markov Decision Process (MDP), a mathematical framework used to model systems that evolve probabilistically over time. MDPs have been applied to diverse problems in fields such as manufacturing, healthcare, and logistics. In our problem, exact solutions for the underlying MDP model are computationally intractable. Therefore, we use Approximate Dynamic Programming (ADP) techniques to approximately solve our MDP model. We will compare the performance of ADP policies with heuristics under various instances of input parameters.

Depending on the nature of a search problem, different search algorithms can be used. Example of search algorithms include suboptimal control algorithms (Shechter et al., 2015), particle swarm optimization algorithms (Derr and Manic, 2009), and Bayesian algorithms (Chung and Burdick, 2007).

In this paper, we make the following contributions:

• Differing from the related literature, we introduce a class of optimal search problems where the search region contains both a target and an obstacle. Additionally, unlike the objects studied in the search literature, each of target and obstacle we consider has some shape.

• We provide an MDP model for this class of problems.

• We utilize an ADP-based approximation technique for approximately solving the underlying MDP. In particular, we approximately solve the search problem using direct-search based ADP, which is a relatively new technique for optimal search problems.

• We compare the resulting ADP policy with easy-to-use heuristics under diverse scenarios.

• Our numerical experiments reveal that the ADP policy outperforms heuristics on majority of problem instances. Hence, we demonstrate that direct-search based ADP is a viable technique for optimal search problems that have target and obstacle.

The paper is organized as follows. In Section 2, we provide the relevant literature. Section 3 includes the description of the search problem and the MDP model. The details of the ADP technique used to solve our MDP model are given in Section 4. We provide numerical results in Section 5. Finally Section 6 includes discussion of the numerical experiments and provides concluding remarks.

LITERATURE REVIEW

Optimal search problems have been studied for several decades (Benkoski et al., 1991; Dobbie, 1968; Haley and Stone, 1980; Stone, 1975; Washburn, 2002). The work on these problems can be mainly categorized as searches for a moving target (Dell et al., 1996; Eagle and Yee, 1990; Singh and Krishnamurthy, 2003; Botea et al., 2013) or for a stationary target (Singh and Krishnamurthy, 2003; Botea et al., 2013; Chung, 2010; Lossner and Wegener, 1982; Ross, 1969; Snider, 2011; Kulich et al., 2014). Since we consider a stationary obstacle and target, we discuss here more the latter set of papers. Ross (1969) studies a search problem where a reward is received upon finding the target and the searcher may decide to stop searching if the reward does not justify the search costs. The author characterizes the form of the optimal strategy for this problem. Wegener (1980) considers features such as overlook probabilities that depend on the number of times a particular cell is searched. The objective is to minimize the expected time of the search. Lossner and Wegener (1982) study a search problem where a switching cost is incurred when going from searching one cell to another and provide conditions under which optimal strategies exist. Chung and Burdick (2007) study a search problem that aims to minimize the time until a decision is made regarding the target’s location. The authors consider the likelihood that the target does not lie in any of the search cells, and propose a Bayesian framework which aims to improve the decision. Chung (2010) studies probabilistic search for a stationary target in a discretized search region considering imperfect detections. Snider (2011) studies an optimal search problem where a target is hidden at a location with a known distribution and a searcher tries to find it using another distribution, with the objective of minimizing the average number of steps to locate the target. Kulich et al. (2014) study the problem of finding an optimal path in order to find a stationary object located in an environment whose map is not a a-priory known. The authors present several strategies for this problem and statistically evaluate their performance for search in comparison to exploration. Shechter et al. (2015) consider the search for a single stationary target. They consider both unconstrained and constrained versions of the search problem. In the unconstrained search, the searcher is allowed to visit cells in any order, whereas the constrained search requires that the searcher visit adjacent cells. The authors provide structural results for the unconstrained search problem and approximately solve reasonably-sized problems of the constrained search using two suboptimal control algorithms.

Furthermore, Nitinavarat and Veerevalli (2017) develop an adaptive search policy for the problem of universal search and stop. The authors show that the knowledge of the target distribution “only useful for improving the reliability for detecting that the target is missing”. El-Hadidy and El-Bagoury (2015) work on a search problem where a single searcher tries to detect a three-dimensional randomly located target in a known zone, with the objective of minimizing the expected value of the time required to detect the target. Gabal and El-Hadidy (2015) study a search problem where a single searcher searches for a randomly located target in a bounded known region. The authors aim to minimize the expected value of the time for detecting the target. El Hadidy (2016) studies a search problem where n searchers try to detect an n-dimensional randomly moving target. The author shows that there is a fuzzy optimal search plan that minimizes the expected value of the first meeting time.

The search literature also includes the application of approximation methods for dynamic programming formulations of optimal search problems. Zhao et al. (2008) study decentralized Bayesian-search problems where multiple autonomous agents search unknown number of targets. The authors formulate these problems as a decentralized optimal control problem using dynamic programming, and they propose a policy iteration algorithm for approximately solving these types of problems.

Our work differs from the search literature in the following aspects. We study a class of search problems with objects having shapes and consider their joint uncertainty. While many papers in the search literature discuss structural properties of the optimal solution, our problem is analytically intractable, and therefore we focus on the performance of suboptimal policies obtained through ADP methods. Our ADP approach is essentially a one-step lookahead method, which is a well known and standard technique for dealing with large state space MDPs (Bertsekas and Tsitsiklis, 1996). However, instead of approximating the cost-to-go through a simple greedy heuristic, we consider a more rigorous approach to approximation based on ADP methodologies. By using basis functions tuned offline to approximate cost-to-go functions, the ADP approach also offers considerable run-time advantages over traditional lookahead approaches, which is critical in settings like MIS.

PROBLEM DESCRIPTION AND MATHEMATICAL FORMULATION

We consider the following class of optimal search problems.

• The search region is discretized into N cells. Without loss of generality, we assume our search takes place over a two dimensional grid.

• There is a stationary target and a stationary obstacle, both of which have some known shape (i.e., they can occupy multiple cells in the search region). If any part of the target (obstacle) is found in a particular cell, then the cell locations of the other parts of the target (obstacle) are known with certainty.

• The initial locations of the target and the obstacle are known probabilistically. We indicate a joint probability distribution of a representative cell for the target and obstacle, since knowing where this marker cell is resolves the location uncertainty of the other parts of that object.

• The target/obstacle is detected with certainty when the searcher examines a cell containing it.

• At each step of the search, the searcher makes two types of actions: 1) which cell to visit next, and 2) which type of search action to choose on the cell visited: a slow but safe exploring or a quick but risky cutting of the cell (i.e., explore or cut).

• The following cost parameters are considered:

(a) c_H^T : cost of hitting the target

(b) c_H^O : cost of hitting the obstacle

(c) c_E : cost of exploring a cell

Note that the first two costs are incurred only if the cut action is taken on a cell containing the target (obstacle), while the third cost is a fixed cost of choosing to safely explore. If the target is located at any step, the problem ends. Otherwise, the joint probability distribution of the representative target and obstacle cells gets updated appropriately (i.e., via Bayes' rule). The objective of the search is then to determine an optimal sequence of cells to be visited, along with the action to choose on each cell, which minimizes the expected cost until locating the target.

We next provide an MDP model for this class of search problems.

The MDP Model

Markov Decision Process (MDP), also known as stochastic dynamic programming, is a mathematical framework used to model systems that evolve probabilistically over time. An MDP consists of the following components:

- Stage: It consists of time periods through which the system evolves.
- State: States capture the key features of the system at various time-points.
- Action: One of the feasible actions is chosen at each state.
- Probabilistic state transitions: An action chosen at each state makes the process probabilistically evolve to a new state.
- Reward/cost: A reward/cost is associated with each state-action pair.

The objective of an MDP is to compute an action in each state so as to maximize expected net reward (or minimize expected cost). MDPs for small-sized problems are optimally solved whereas for large-sized problems, approximate dynamic programming (ADP) techniques are used to obtain approximate solutions.

In order to efficiently express the components of our MDP model when considering objects with shape, we compute the probability of an object being in a particular cell in terms of the probability distribution of a representative cell of that object. Specifically, a particular cell of the obstacle is marked O_1 and a specific cell of the target is marked T_1. We assume that the rest of cells of each object are completely known upon finding one of its components and therefore it is sufficient to represent states in terms of the joint probability distribution of the marker cells for obstacle and target, respectively. Without loss of generality, we will refer to the example of an "L" shaped target and obstacle in Figure 1 throughout rest of the section to explain various aspects of the MDP components. Random variables O_1, O_2, O_3, and T_1, T_2, and T_3 are used to represent the locations of the cell components of the obstacle and the target, respectively. For specific locations of the objects given in Figure 1, the values of O_1, O_2, O_3 are 2, 1, and 7, respectively, and the values of T_1, T_2, T_3 are 35, 34, and 40, respectively. Based on the above discussion, we assume that we can know the entire location of the obstacle and the target through representative cells O_1 and T_1, respectively.

State Space

The state space of our MDP model takes the following form:

$$s = p_{ij}, i = 1, K\ , N, j = 1, K\ , N$$

where p_{ij} is probability that $O_1 = i$ and $T_1 = j$, $i = 1, K\ , N$, and $j = 1, K\ , N$.

The relationship between the search space and the entries of the probability matrix (hereafter to be called the "P matrix") is illustrated in figures 1 and 2. Here the search region consists of $N = 48$ (6*8) cells and entry $p_{2,35}$ would indicate the a priori probability that $O_1 = 2$ and $T_1 = 35$.

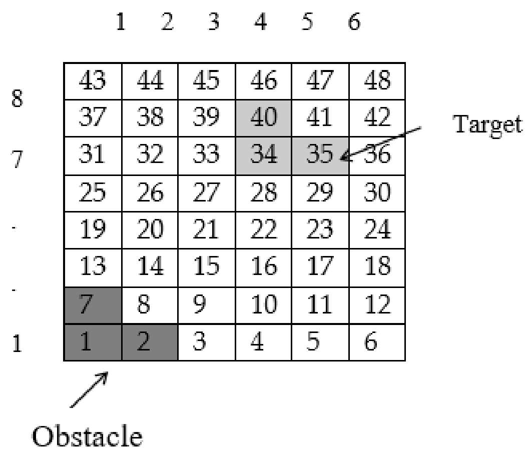

Figure 1. Search space

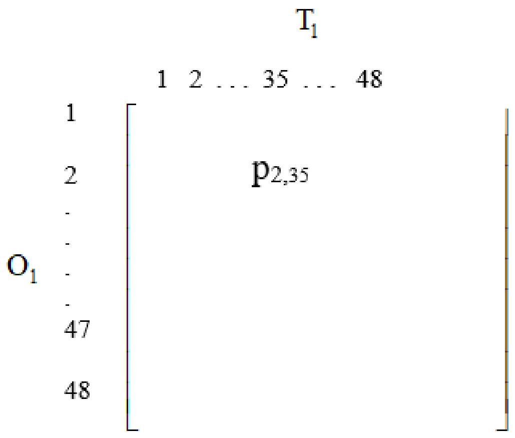

Figure 2. The Probability matrix

Action Space

The action space takes the following form:

$$a=(a_1,a_2),$$

where $a_1 \in A_k$ is the location of the cell to be searched ($A_k \subset \{1,\mathrm{K},N\}$ is the set of feasible cells that can be searched at step k (i.e., in our case, the cells adjacent to those already searched)), and $a_2 \in \{E,C\}$ is the action chosen for the cell to be searched (E and C stand for "explore" and "cut", respectively.).

Probability Updates

After searching a cell, say, i, at the current step, the process moves to a new state, which requires the update of the entries of the P matrix. Depending on whether an object is found in cell i, the probability updates will be different, which is explained next.

Case 1: Finding nothing in the cell: As an example, suppose cell 28 is searched and nothing is found there. Then we know that O_1, O_2, and O_3 are not there, which equivalently means that O_1 cannot be in cells 28, 29, or 23 by its geometry. (Note that, since we update the P matrix by considering only O_1, we zero out only 23rd, 28th, and 29th rows and columns. For instance, we do not zero out rows and columns corresponding to cells 22, 27, and 33 since O_1 can still be in one of those cells. In particular, while cell 28 is empty, the obstacle can be found in cells 32,33, and 38, or in cells 21, 22, and 27, or in cells 26, 27, and 32). The same holds for T_1. Therefore, as suggested in Figure 3, we perform the Bayesian update of the P matrix by zeroing out the appropriate rows and columns, and renormalizing the remaining probabilities. We can easily adapt the updating method to consider the geometry of any other shape.

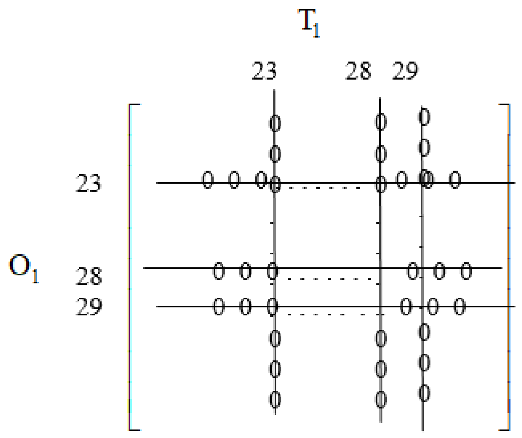

Figure 3. The view of the the probability matrix when no object is found.

Case 2: Finding the obstacle: As discussed, finding any part of the obstacle is equivalent to finding O_1 in its respective cell. Suppose the obstacle found is situated as shown. Since it becomes known that O_1 is in cell 28, we first update the P matrix by zeroing out all other rows except for row 28. Furthermore, by the geometry of T, if O takes up cells 27, 28, 33, then T_1 cannot be in those cells nor cells 22, 23, 29, nor 34, and we put zeroes in the corresponding columns for T_1. Note that T_1 cannot be in cell 34 since we know in this case that the target would have to occupy cells 34, 33, and 39, which is impossible as cell 33 is occupied by the obstacle. Finally, we renormalize the remaining probabilities in row 28 to update the P matrix.

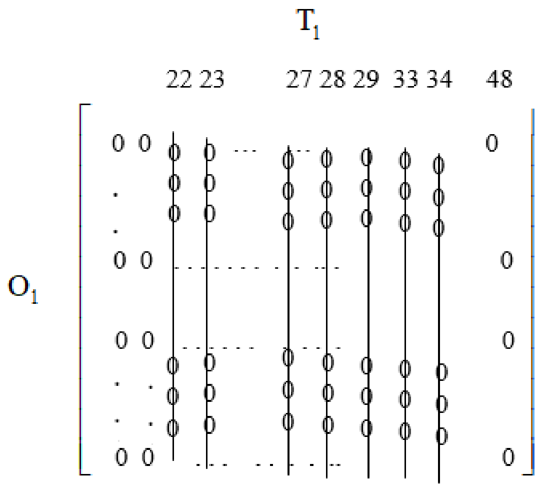

Figure 4. The view of the search space and the probability matrix when the obstacle is found

Case 3: Finding the target: As stated earlier, when any component of the target is detected in cell i, the search problem ends. In addition, when the exact location of the target is known but not detected through the "cut" or "explore" action, the problem still ends.

Immediate Costs

At any step of the search, an immediate cost is incurred upon choosing a cell to be searched along with the action to be performed on the selected cell. This cost equals the fixed exploration cost for the "explore" action, whereas it equals the expected cost of hitting an object on the selected cell for the "cut" action.

The Cost-to-Go Function

The cost-to-go function for a given state s is expressed as follows:

$$\begin{aligned} V(s) = \min_{i \in A_k} \{ \min \{ & c_E + P(O_1 = i)V(s^1) + P(O_2 = i)V(s^2) + P(O_3 = i)V(s^3) + (1 - P(T \in i) - \\ & P(O \in i))V(s^4), P(T \in i)c_H^T + P(O \in i)c_H^O + P(O_1 = i)V(s^1) + P(O_2 = i)V(s^2) + P(O_3 = i)V(s^3) + \\ & (1 - P(O \in i) - P(T \in i))V(s^4)\}\}, \end{aligned} \tag{1}$$

where s^1 is the new state upon finding O_1 in cell i (obtained using the probability updates described earlier), s^2 is the new state upon finding O_2 in cell i, s^3 is the new state upon finding O_3 in cell i, and s^4 is the new state upon finding no object in cell i. Note that by the optimality equations, conditional on visiting a particular cell, we can easily determine the optimal action of "cut" or "explore" because both actions have the same expected cost to go thereafter. However, it is the choice of which cell to visit next that greatly complicates things.

Our state space consists of all probability maps that could result after different sequences of cell visits, which makes exact solution with traditional MDP algorithms intractable for even moderately sized grids. Instead, we approximately solve the MDP using ADP techniques.

APPROXIMATE DYNAMIC PROGRAMMING

ADP has been widely used to deal with MDP's curse of dimensionality in diverse fields such as manufacturing, construction, healthcare, and revenue management (Powell, 2007). These techniques are classified as linear programming (LP)-based ADPs and simulation-based ADPs. LP-based ADPs start with a linear program whose optimal solution equals the optimal value function in Bellman's equations of the underlying MDP (in our problem, optimal value function corresponds to *V(s)* of Eq. (1)). Because of an inordinate number of variables and constraints, the resulting LP is approximately solved using techniques such as column generation and constraint sampling (Adelman, 2003; Adelman, 2004; De Farias and Roy, 2003). Simulation-based ADPs differ from LP-based ADPs in that they employ simulation models such as statistical sampling and reinforcement learning methods for estimating the value functions (Bertsekas and Tsitsiklis, 1996; Chang et al., 2007; Sutton and Barto, 1998).

In both LP and simulation-based approaches, a core concept in ADP is the approximation of the value function through a combination of basis functions. The intuition behind basis functions are that they represent some essential features of the state that may be predictive of the value of being in that state (Bertsekas and Tsitsiklis, 1996). One of the common ways of approximating the value function through basis functions is linear approximation, which takes the following form:

$$V(s) \approx \sum_{k=1}^{K} r_k \Phi_k(s), \tag{2}$$

where r_k for $k = 1, \mathrm{K}, K$ are tuning parameters and $\Phi_k(s)$ for $k = 1, \mathrm{K}, K$ are basis functions. After the value function is approximated, the approximation parameters are tuned iteratively to obtain an ADP policy. More specifically, ADP techniques aim to find the optimal parameter vector that minimizes a certain performance metric such as the sum of squared differences between the approximate cost-to-go function and the estimated cost-to-go function over sampled states. The resulting optimization problem is typically solved using regression-based techniques (Bertsekas and Tsitsiklis, 1996; Powell, 2007); however, we apply direct search, an approach that tunes the ADP parameters by solving an optimization problem that aims to maximize the performance of the policy resulting from those parameters (Maxwell et al., 2013; Shechter et al., 2015). The ADP policy is then obtained using the approximate value functions.

Retrieving the ADP Policy from the Approximate Value Function

After the parameter tuning phase is performed and hence the approximate value of a given state is available (i.e., the approximate value of *V(s)* for a state *s*) due to having final values of *rk*'s in Eq. (2), the ADP policy is retrieved by computing a decision vector (i.e., (a_1, a_2)) for any desired state of the system. That decision vector is myopic with respect to value function approximation of our MDP. Mathematically, the decision retrieval problem for a particular state s of our MDP model is expressed as follows:

$$\begin{aligned}&\min_{i \in A_k} \{\ \min \{c_E + P(O_1{=}i)\tilde{V}(s^1) + P(O_2{=}i)\tilde{V}(s^2) + P(O_3{=}i)\tilde{V}(s^3) + (1 - P(T \in i) - \\ &P(O \in i))\tilde{V}(s^4), P(T \in i)c_H^T + P(O \in i)c_H^O + P(O_1{=}i)\tilde{V}(s^1) + P(O_2{=}i)\tilde{V}(s^2) + \\ &P(O_3{=}i)\tilde{V}(s^3) + (1 - P(O \in i) - P(T \in i))\tilde{V}(s^4)\}\},\end{aligned} \tag{3}$$

where $\tilde{V}(s^i)$ for $i = 1, \mathrm{K}, 4$ is the approximate value of state s^i.

Note that the structure of our problem enables us to calculate the expected cost of the search exactly, owing to the fact that state transitions of our MDP model are deterministic. We therefore do not need to evaluate policies using simulation as in simulation-based ADPs. In this sense, our way of implementing ADP differs from both simulation-based ADPs and LP-based ADPs.

We next discuss the details about basis functions and the parameter tuning method we chose for solving the MDP model introduced earlier.

Basis Functions

While basis functions can take on a wide variety of forms and calculations, based on our motivating scenario (MIS), we consider easy-to-calculate basis functions, which may be better suited to real-time decision making. We experimented with several different forms of the basis function, and settled on the following one to test extensively in numerical experiments.

Component-based Basis Functions: Each basis function equals the probability that any component of the target or obstacle is located at a distinct cell. These basis functions have the following form.

$$\Phi_k(s) = p_k(s),$$

where $p_k(s)$ is $P(T \in k \cup O \in k)$ at state s. In other words, to each cell there is a corresponding basis function that indicates the probability of finding any part of the obstacle or target there. Using these basis functions, we have the following approximation for $V(s)$:

$$\tilde{V}(s) = \sum_{k=1}^{N} r_k\, p_k(s).$$

Direct Search

We tune parameters using direct search, an approach described in the ADP literature (Adelman, 2004). Whereas regression-based techniques seek good approximations to the value function, direct search focuses on the ultimate goal of finding good policies. In other words, it tackles an optimization problem where the variables consist of feasible r's and the objective function value is the expected cost of the policy induced by the corresponding parameter vector. The resulting optimization problem takes the following form:

$$\min_{r \in R^N} E[\sum_{t=0}^{T} c(s_t, \pi_r(s_t))], \tag{4}$$

where T is a random variable denoting the final step of the search, s_t is the state at step t of the search, π_r is the policy obtained by the parameter vector r, $\pi_r(s_t)$ is the action dictated by the policy π_r in the state at stage t, and $c(s_t, \pi_r(s_t))$ is immediate cost incurred at step t as a result of choosing $\pi_r(s_t)$. Here, π_r is obtained by solving the aforementioned decision retrieval problem via the parameter vector r used to approximate the value function for each possible state visited during the search. The objective function in Eq. (4) is the expected cost of the policy π_r. With the implementation of direct search, we obtain the values of r_k's in Eq. (2) and thus have the approximate value of a given state.

We employ Simulated Annealing (SA) as a method for seeking a vector r that solves the optimization problem of Eq. (1). In other words, we heuristicallly solve this optimization problem using SA and hence obtain good but not optimal solutions for the r vector. The main reason for choosing SA over other heuristic algorithms is its ability to find high-quality solutions quickly (Suman and Kumar, 2006).

Simulated Annealing for Direct Search

SA is one of the efficient heuristic algorithms used to solve global optimization problems (Suman and Kumar, 2006). Its features were inspired from an annealing process in which a material is cooled slowly to alter its properties such as strength and hardness. Starting from an initial temperature, the annealing process cools the material slowly by periodically reducing the temperature, with the objective of minimizing its energy level (Suman and Kumar, 2006).

At a given feasible solution, the SA algorithm generates a new feasible solution using a neighbourhood scheme. It accepts the new feasible solution when its objective function value is better than that of the current solution; otherwise, the new worse solution is accepted with a probability which is decreasing in the absolute difference between the values of the current and new solution, and increasing in the current temperature. To implement this, a uniform $[0,1]$ random number is generated, and if it is less than the following term, the current solution is replaced by the new solution:

$$e^{-\frac{\Delta E}{T}},$$

where ΔE is the absolute difference between the values of the new and current solution, and T is the current temperature.

The key feature of SA is that it does not get stuck in local minima because of its ability to visit worse solutions, and therefore has the potential to reach a global minima. After a certain number of iterations, the temperature of the SA algorithm is periodically decreased in order to lower the probability of accepting a worse solution. The algorithm stops after a certain number of iterations or after a stopping criteria is met.

The features of our SA algorithm for solving the optimization problem in direct search are listed below.

- **Initial Solution:** Each component of the vector r is set to a Uniform $[0,2\sqrt{N}]$ random number.
- **Neighbourhood Scheme:** At each iteration, a new solution is generated by perturbing each component of a randomly chosen fraction of r with a random quantity using a Uniform $[0,\sqrt{N}]$ distribution. The fraction value is fixed at 0.75.
- **Temperature Schedule:** After every m iterations, temperature T is set to $T(1-\varepsilon)$, where $\varepsilon \in (0,1)$.
- **Parameter Values:** Number of iterations K, m, ε, and initial temperature are set to 500, 10, 0.1, and 1, respectively.

A Pseudo-code for Direct-search based ADP

```
The pseudo code for our ADP algorithm is given below.
Step 1. Come up with basis functions for the value function approximation.
    Step 1.1. Determine the number of basis functions and hence the size of the parameter vector $r$ (referring to Eq. (2)).
Step 2. Apply direct search for tuning parameter vector $r$.
    Step 2.1. Solve the optimization problem in Eq. (4) heuristically using SA algorithm.
        Step 2.1.1. At each step of SA, determine a parameter vector $r$ using the neighborhood scheme.
        Step 2.1.2. Evaluate the objective function in Eq. (4) using the current solution (i.e., the parameter vector $r$ determined in Step 2.1.1.)
        Step 2.1.3. Revise the current solution by either accepting the worse solution or changing it with a better solution. Go to Step 2.1.2 until the SA algorithm ends.
Step 3. Retrieve the ADP Policy from the approximate value function
    Step 3.1. Solve the optimization problem in Eq. (3) to obtain a decision vector for any desired state of the system.
```

NUMERICAL EXPERIMENTS

We studied the search problem on a two-dimensional grid. Problem instances were generated for a variety of cases determined by the shapes and the location distributions of the objects. Specifically, we considered two different cases based on the shapes of the objects: 1) Unit-shape case, in which each of the objects occupies a single cell, and 2) L-shape case, in which each of the objects occupies three cells (as shown in Figure 1). The location distributions of both of the objects were determined using a discretized bivariate Gaussian distribution. We followed a two-step approach for constructing the joint probability distribution of O_1 and T_1. First we considered possible locations of O_1 in the search region, and then considered possible locations of T_1 conditioned on each location of O_1. The resulting values for $P(O_1)$ and $P(T_1 | O_1)$ were then used to construct the P matrix describing $P(O_1 \cap T_1)$. The location distribution of T_1 can then be obtained through the marginal distribution of the P matrix. Furthermore, we also considered the rigid connection case where upon finding the obstacle in the search region, the exact location of the target is known. In this rigid case, finding the location of O_1 or T_1 resolves the uncertainty of the other in that T_1 is located exactly at some fixed offset from O_1 (e.g., 3 to the right and 3 up).

The size of the search region was determined based on different levels of the variance of the location distributions so as to provide sufficient coverage of the possible object locations. We considered a $13*13$ search region, which can be regarded as a region for a reasonably-sized problem. The covariance of the Gaussian distribution for the obstacle was set to the identity matrix **I**, which means that σ equals 1. The covariance of the Gaussian distribution for the target conditional on the location of the obstacle was set to $0.25I$ and 0, corresponding to two different cases: the *scaled case* (i.e., the variance for the conditional target distribution $P(T_1 \mid O_1)$ is scaled) and the *rigid case*. For each level of standard deviation, we generated two different cases: 1) complete overlap, in which the location distributions of the objects overlap each other (i.e., both objects have the same expected location); 2) partial overlap, in which there is a partial overlap between the location distributions of the objects (see Figure 5). These cases were constructed through different considerations of the distance between the mean locations of the objects. As an example, the complete overlap corresponds to an offset of (0,0) for the expected location of T_1 relative to the expected location of O_1, while the partial overlap case uses an offset of (3,3).

We used different levels for the cost parameters in order to see their effects on the performance of various decision rules. Specifically, for each combination of standard deviation, problem size, and offset, we generated three problem sets for each of the Unit-shape cases and the L-shape cases by varying the hitting costs of the target and the obstacle. Finally, the exploration cost of a cell was fixed at 1.

In implementing our SA algorithm, we used multiple replications. Specifically, the number of replications was set to 5 for each problem set. This means that the SA algorithm starts with a distinct vector of r at each replication. The best value obtained after running SA with five replications was reported as the expected cost of the ADP policy.

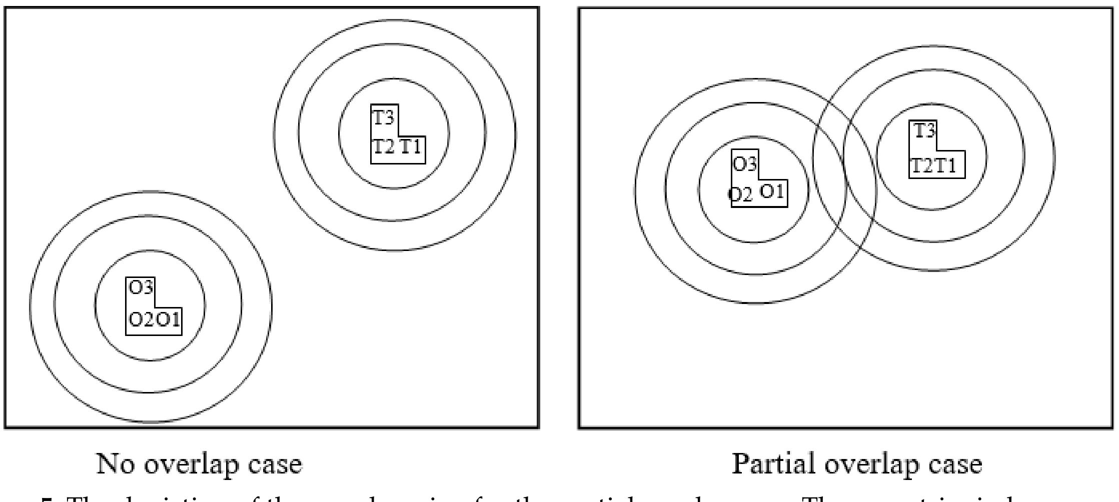

Figure 5. The depiction of the search region for the partial overlap case. The cocentric circles represent contour lines of the bivariate Gaussian distribution for the obstacle and the target, respectively.

Heuristic Decision Rules for Comparison

We compared the performance of the ADP policy against the following two heuristics:

MinProb: This policy chooses at each step of the search the cell with the lowest probability that an object (i.e., a target or an obstacle) is located therein, and then chooses the cut or explore action according to the minimum of the exploration cost and the expected cost of hitting an object for that cell. Note that when the cost of hitting the target and the obstacle are equal, this policy is a greedy policy in the sense of choosing the action which yields the smallest immediate cost.

MaxProb: At each step of the search, MaxProb chooses the cell with the highest probability that an object (i.e., a target or an obstacle) is located therein. Therefore, this heuristic may be considered greedy in the sense of trying to locate an object as quickly as possible. After choosing the cell to examine, the MaxProb policy chooses the action (“cut" or “explore") on the chosen cell according to whichever yields the smallest immediate cost.

Results

Computational results for the different cases are reported in Tables 1-4. Each table lists the percentage improvement obtained by the ADP policy over the best of the greedy policies for each of the problem sets. As can be seen from all of the tables, the Maxprob heuristic performs much better than the Minprob heuristic in every case. Therefore, the final column of each table represents the percent improvement of the ADP policy over the Maxprob heuristic.

We begin by giving the results for the rigid case (Tables 1 and 2). These results reveal that the ADP policy outperforms MaxProb, the best greedy heuristic, in most of the problem sets in the L-shape case (Table 2), whereas it generally performs better than MaxProb in the Unit-shape case (Table 1). The average improvements over MaxProb are about 5% and 5.8% for the Unit-shape rigid case and the L-shape rigid case, respectively. On the other hand, the ADP policy outperforms MaxProb in the L-shape scaled case, while it performs roughly the same as MaxProb in the Unit-shape scaled case (Tables 3 and 4). In particular, the ADP policy performs on average 6.3% and 0.4% better than MaxProb in the L-shape scaled case and the Unit-shape scaled case, respectively. Finally, note that the maximum improvement over MaxProb is about 17% (see Tables 1 and 2).

Table 1. The results for the Unit-shape rigid case.

Offset	c_H^T	c_H^O	**MinProb**	**MaxProb**	**ADP**	**% Imprv.**
1	50	150	17.62	6.16	5.61	8.93
	100	100	18.87	6.54	6.18	5.5
	150	50	18.91	6.81	5.64	17.18
2	50	150	16.45	6.37	6.42	-0.78
	100	100	18.87	6.76	7.2	-6.51
	150	50	20.07	7.05	6.66	5.53

Table 2. The results for the L-shape rigid case.

Offset	c_H^T	c_H^O	**MinProb**	**MaxProb**	**ADP**	**% Imprv.**
1	50	150	17.62	4.33	4.35	-0.46
	100	100	18.87	4.71	4.64	1.49
	150	50	18.91	4.94	4.3	12.96
2	50	150	16.41	4.58	4.55	0.66
	100	100	18.76	4.94	4.79	3.04
	150	50	19.9	5.21	4.32	17.08

The expected costs of the search for the ADP policy and MaxProb are affected by the shapes of the objects as well as the uncertainty of the location of the target. In particular, the expected costs of the search for the ADP policy and MaxProb are higher in the Unit-shape case than in the L-shape case for a fixed type of location distributions (Table 1 versus Table 2, and Table 3 versus Table 4). While the larger objects (e.g., L-shape vs. Unit-shape) have more surface area that might be accidentally damaged, they also eliminate more cells from the probability matrix when nothing is found. This advantage in the uncertainty reduction apparently outweighs the higher risk of hitting the object. As expected, the expected costs for these policies are higher in the scaled case than in the rigid case when both objects have the same shape (Table 1 versus Table 3, and Table 2 versus Table 4). When the uncertainty of the location of the target conditional on the location of the obstacle is low (corresponding to the rigid case), finding the obstacle during the search yields more information, thereby resulting in smaller expected cost.

It is also worth noting that the performance of the ADP policy with respect to Maxprob decreases with the increase in offset for the unit-shape case. Whereas the ADP policy becomes a more viable method with the increase in offset for the L-shape rigid case.

Table 3. The results for the Unit-shape scaled case.

Offset	c_H^T	c_H^O	**MinProb**	**MaxProb**	**ADP**	**% Imprv.**
0	50	150	20.94	8.58	7.98	6.99
	100	100	23.19	8.68	8.2	5.53
	150	50	24.36	8.59	8.52	0.81
1	50	150	23.2	7.53	7.96	-5.71
	100	100	24.3	7.78	7.84	-0.77
	150	50	24.76	8.03	8.37	-4.23

Table 4. The results for the L-shape scaled case.

Offset	c_H^T	c_H^O	**MinProb**	**MaxProb**	**ADP**	**% Imprv.**
0	50	150	20.99	5.89	4.98	15.45
	100	100	23.13	5.98	5.31	11.20
	150	50	24.15	5.97	5.43	9.05
1	50	150	23.2	5.35	5.29	1.12
	100	100	24.3	5.61	5.64	-0.53
	150	50	24.76	5.87	5.79	1.36

DISCUSSION

We studied a class of optimal search problems where the objective is to find a stationary target with mimimum expected cost. Differing from other related work in the literature, we considered a search region with objects (i.e., a target and an obstacle) that have shapes and whose locations are known through a joint probability distribution. We formulated these problems as an MDP, which is intractable to exact solution methods because of the size of its state space. To approximately solve these problems, we employed an ADP technique via direct search, which tunes the resulting ADP policy based on its performance. As a direct search method, we invoked a Simulated Annealing (SA) algorithm that solves the resulting optimization problem where the variables are the ADP approximation parameters.

As noted earlier, ADP techniques have been successfully used for solving a variety of large-scale MDPs. Our work demonstrated that employing these techniques to solve optimal search problems can yield promising results as long as the approximation architecture used to approximate the value function of the underlying MDP is properly determined. In particular, our numerical experiments revealed that the ADP policy generally performs better than easy-to-use heuristic decision rules in the majority of the cases. Furthermore, we observed that, the success of the ADP technique for these problems is highly contingent on the set of basis functions chosen for the approximation architecture. For example, we tried one set of basis functions which were the entries of the P matrix itself, but this resulted in poor performance of the ADP policy. We also tried basis functions calculated by solving some sub-problems via greedy heuristics, but these were computationally expensive, and would be impractical for real-time decision making such as MIS. It is therefore crucial to try a variety of basis functions before finalizing the components of the ADP technique.

A challenge in applying direct search for tuning the ADP parameters is that in the end, we do not know for sure if we have chosen basis functions that could not possibly yield good policies, or if our global optimization approach simply did not find a good set of tunable parameters. As noted, our motivating problem is MIS, which would require fast real-time decision making. This puts practical constraints on the types of basis functions that may be considered. Moreover, it may not be reasonable to invest the huge computational effort to search for a variation of ADP that outperforms a heuristic policy. In practice, whether to apply an easy-to-use decision rule or ADP would depend on the problem parameters.

REFERENCES

Adelman D. "Price-directed replenishment of subsets: methodology and its application to inventory routing". Manufacturing and Service Operations Management. 5, 4, 348-371, 2003.

Adelman D. "A price-directed approach to stochastic inventory routing". Operations Research. 52, 4, 499-514, 2004.

Benkoski S J. Monticino, M. G., Weisinger, J. R.. "A survey of the search theory literature". Naval Research Logistics. 38, 469-494, 1991.

Bertsekas D, Tsitsiklis J. "Neuro-Dynamic Programming", Athena Scientific, 1996.

Botea A, Baier J, Harabor D, Hernandez C. "Moving target search with compressed path databases". In Proceedings of ICAPS-13, 2013.

Chang HS, Fu MC, Hu J, Marcus SI. "Simulation-based algorithms for Markov Decision Processes", Springer, 2007.

Chung TH, Burdick JW. "A decision-making framework for control strategies in probabilistic search". Proceedings of IEEE International Conference on Robotics and Automation, 2007.

Chung TH. "On probabilistic search decisions under searcher motion constraints". Algorithmic Foundations of Robotics, 2010.

De Farias DP, Roy BV. "The linear programming approach to Approximate Dynamic Programming". Operations Research. 51, 850-865, 2003.

Dell RF, Eagle JN. Martins, G. H. A., Santos, A. G. "Using multiple searchers in constrained-path, moving-target search problems". Naval Research Logistics. 43, 463-480, 1996.

Derr K, Manic, M. "Multi-robot, multi-target particle swarm optimization search in noisy wireless environments". Proceedings of the 2nd conference on human system interactions, Catania, Italy, 2009.

Dobbie JM. "A survey of search theory". Operations Research. 16(3), 527–537, 1968.

Eagle JN, Yee JR. "An optimal branch-and-bound procedure for the constrained path, moving target search problem", Operations Research. 38, 110-114, 1990.

El-Hadidy, M. A. A., El-Bagoury, A. A. H., "Optimal search strategy for a three-dimensional randomly located target", International Journal of Operational Research. 29, 2015.

El-Hadidy, M. A. A. "Fuzzy Optimal Search Plan for N-Dimensional Randomly Moving Target", International Journal of Computational Methods. 13, 2016.

Gabal, H. M. A., El-Hadidy, M. A. A., "Optimal searching for a randomly located target in a bounded known region", International Journal of Computing Science and Mathematics. 6, 2015.

Haley WB, Stone LD. "Search Theory and Applications", Plenum Press, New York, 1980.

Kulich M, Preucil L, Jose J, Bront M. "Single robot search for a stationary object in an unknown environment", IEEE International Conference on Robotics   Automation, 2014.

Lossner U, Wegener I. "Discrete sequential search with positive switch cost". Mathematics of Operations Research. 7(3), 426–440, 1982.

Maxwell MS, Henderson SG, and Topaloglu H. "Tuning Approximate Dynamic Programming Policies for Ambulance Redeployment via Direct Search". Stochastic Systems. 3, 1-40, 2013.

Najemnik J, Geisler WS. "Eye movement statistics in humans are consistent with an optimal search strategy", J. Vis. 8, 1-14, 2008.

Nitinawarat, S., Veeravalli, V. V., "Universal scheme for optimal search and stop", Bernoulli. 23, 1759-1783, 2017.

Powell WB. "Approximate Dynamic Programming: Solving the Curses of Dimensionality", Wiley, 2007.

Ross SM. "A problem in optimal search and stop". Operations Research. 17, 984–992, 1969.

Shechter SM, Ghassemi F, Gocgun Y, Puterman ML. "Trading off quick versus slow actions in optimal search". Operations Research. 63, 353-362, 2015.

Singh S., Krishnamurthy, V. "The optimal search for a markovian target when the search path is constrained: the infinite-horizon case". IEEE Transactions on Automatic Control, 2003.

Snider J. "Optimal random search for a single hidden target". Physical Review. 83, 011105, 2011.

Stone LD. "Theory of Optimal Search", Academic Press, 1975.

Suman B, Kumar B. "A survey of simulated annealing as a tool for single and multiobjective optimization". Journal of the Operational Research Society. 57, 1143–1160, 2006.

Sutton RS, Barto AG. "Reinforcement Learning : an introduction", MIT Press, 1998.

Washburn A." Search and Detection", Fourth Edition, Institute for Operations Research and the Management Sciences, 2002.

Wegener I. "The discrete sequential search problem with nonrandom cost and overlook probabilities". Mathematics of Operations Research. 5, 373–380, 1980.

Wettergren TA. "Performance of search via track-before-detect for distributed sensor networks", IEEE Transactions on Aerospace and Electronic Systems, 44, 2008.

Zhao Y, Patek SD, Beling PA. "Decentralized Bayesian Search Using Approximate Dynamic Programming Methods". IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 38, 2008.