Algorithms for Informed Cows

Ming-Yang Kao and Michael L. Littman
Department of Computer Science
Duke University, Durham, NC 27708-0129
{kao, mlittman}@cs.duke.edu

Abstract

We extend the classic on-line search problem known as the cow-path problem to the case in which goal locations are selected according to one of a set of possible known probability distributions. We present a polynomial-time linear-programming algorithm for this problem.

Introduction

In on-line search, a great deal of work has been carried out concerning how to minimize the worst-case performance of a search strategy, for example as compared to an omniscient searcher. In many applications, this notion of optimal performance is simply too conservative—the agent might know, based on its prior experience, that some locations are more promising as targets than others.

In this paper, we address the problem of finding good search strategies given a particular kind of prior knowledge. The search scenario we consider is the classic cow-path problem (Baeza-Yates, Culberson, & Rawlins 1993), in which a cow stands on a path and wants to find a pasture in which to feed (see Figure 1). There are n locations to either side of the cow: p_{-n} through p_{-1} on its left, and p_1 through p_n on its right. Exactly one location contains a pasture and the cow can identify a pasture only by standing in it. The prior knowledge the cow has is expressed in terms of discrete probability distributions over the $2n$ possible locations of the pasture. We assume the cow knows that the pasture location is distributed according to one of m possible probability distributions $\Pi^1, \ldots, \Pi^m$, but it doesn't know which. The cow would like to follow a search pattern that minimizes the worst possible expected value of the ratio of the number of steps it takes to find a pasture to the number of steps it would need if it knew the probability distribution in advance (i.e., that the pasture location is distributed according to Π^i for some $1 \leq i \leq m$). Thus, the cow is required to minimize its worst-case *competitive ratio* (Sleator & Tarjan 1985; Yao 1980).

In the next section, we look at the case in which the cow knows the probability distribution of the pasture location and must simply minimize the expected number of steps to find it; we provide a quadratic-time dynamic-programming algorithm for this problem. In the section following, we show how to extend this result to find a worst-case optimal randomized search strategy when there are multiple possible probability distributions ($m > 1$). We show how to solve the problem in time polynomial in n and m (and the number of bits in the probability values used to specify the problem), via linear programming. In the final section, we discuss some possible future directions of this work and connections to earlier work.

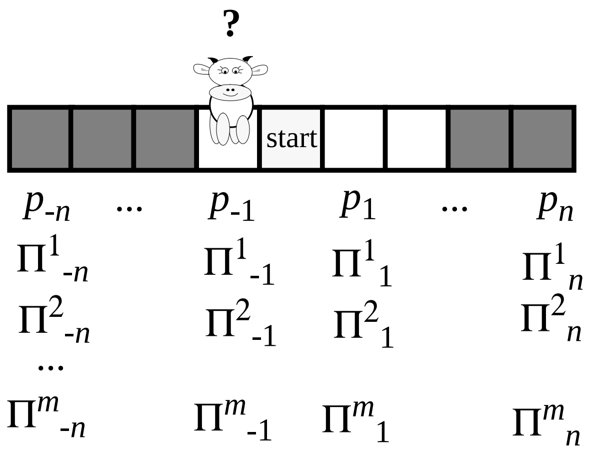
?
start
p_{-n} ... p_{-1} p_1 ... p_n
$\Pi^1{}_{-n}$ $\Pi^1{}_{-1}$ $\Pi^1{}_1$ $\Pi^1{}_n$
$\Pi^2{}_{-n}$ $\Pi^2{}_{-1}$ $\Pi^2{}_1$ $\Pi^2{}_n$
...
$\Pi^m{}_{-n}$ $\Pi^m{}_{-1}$ $\Pi^m{}_1$ $\Pi^m{}_n$

Figure 1: The informed-cow cow-path problem. This cow has already explored 3 of the 8 locations. Cow artwork used with permission (Max's Mom and Dad 1994).

Minimizing Expected Search Time

In this variation of the problem, the cow knows that the pasture location is distributed according to a probability distribution Π^i. All the relevant probabilities concerning the dynamics of the search problem are known, but the cow does not know a priori which location has the pasture. This problem can be formulated as a finite-horizon partially observable Markov decision process (POMDP) and solved using standard methods (Kaelbling, Littman, & Cassandra 1995).

Standard exact algorithms for POMDPs work in the space of *information states* of the searcher; these information states summarize this history of the search process as a probability distribution over the underlying states (in this case, the cross product of locations of the cow and locations of the pasture). Unfortunately, the problem of solving finite-horizon partially observable Markov decision processes is PSPACE-hard (Papadimitriou & Tsitsiklis 1987) in general, so a general-purpose POMDP solver would probably not be appropriate for this problem.

The particular POMDP for the informed-cow cow-path problem has a great deal of additional structure that makes the use of a general-purpose algorithm unnecessary. In particular, the situations in which the cow needs to make a decision can be represented in an extremely efficient way: (ℓ, r, s) where ℓ is the number of explored locations to the left of the starting location, r is the number of explored locations to the right of the starting location, and s is 0 if the cow is currently on the left side of the explored region and 1 if the cow is currently on the right side of the explored region. Using this *situation* notation, the cow's initial situation is $(0,0,0)$ (or $(0,0,1)$, equivalently), and the situation depicted in Figure 1 is $(1,2,0)$. The effect of moving to the left from (ℓ, r, s) is to either find the pasture in location $p_{-(\ell+1)}$ or to move to situation $(\ell+1, r, 0)$ (if $\ell < n$), and the effect of moving to the right is to either find the pasture in location $p_{(r+1)}$ or to move to situation $(\ell, r+1, 1)$ (if $r < n$).

Given a situation (ℓ, r, s), we can compute an information state quite easily. We know the cow is at location $p_{-\ell(1-s)+r(s)}$ (i.e., the extreme left or extreme right side depending on s of the explored region), and the probability that the pasture is in location p_j is $\Pi^i_j / \nu^i(\ell, r, s)$ where

$$\nu^i(\ell, r, s) = \sum_{j'=-n}^{-(\ell+1)} \Pi^i_{j'} + \sum_{j=r+1}^{n} \Pi^i_{j'}. \qquad (1)$$

The function $\nu^i(\ell, r, s)$ represents the probability that the pasture is not found in a search that leaves the cow in situation (ℓ, r, s).

There are several properties of the informed-cow cow-path problem that justify this situation representation. First, the first time a location is visited, either the cow discovers the pasture and the search ends, or it discovers that the pasture is elsewhere, and the search continues. This implies that while the cow is searching, all locations are either known to be pasture-free or presently unexplored. Second, the linear nature of the path means that the explored locations will form a contiguous region around the starting location. Third, given that the cow is at one end of the explored region, it can choose to move to the left or to the right. If it moves into the explored region, its only rational course

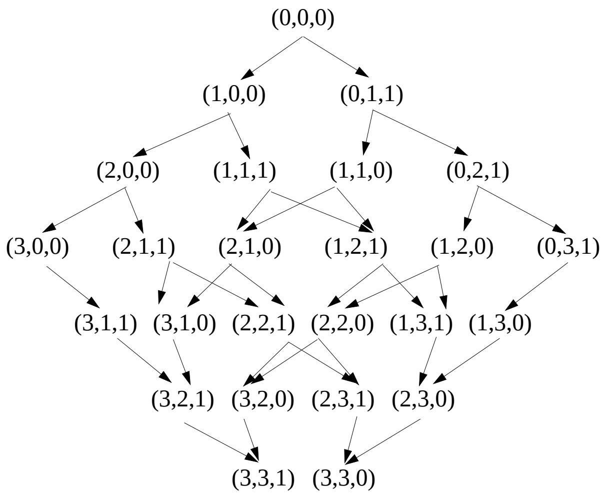

Figure 2: Relationship between the possible situations for $n = 3$ arranged as a situation DAG.

of action is to continue moving in the same direction until it reaches the opposite edge of the explored region; turning around midstream is wasteful because it provides no additional information and uses steps.

The set of reachable situations can be arranged into a directed acyclic graph, called the situation DAG, illustrated in Figure 2. The expected number of steps to the pasture from a given situation can be computed by considering the two choices, left and right. If the cow chooses to go left from situation (ℓ, r, s), it will find the pasture in the next step with probability $\Pi^i_{-(\ell+1)(1-s)+r(s)} / \nu^i(\ell, r, s)$ and with the remaining probability, it will enter situation $(\ell+1, r, 0)$. It is a simple matter to compute the expected number of steps from these quantities as a function of the expected number of steps from situation $(\ell+1, r, 0)$ (this can be computed recursively). Similarly, we can compute the expected number of steps to the pasture given that the cow moves right from (ℓ, r, s). The optimal expected number of steps to the pasture from (ℓ, r, s) is just the minimum over the expected number of steps given an initial rightward step and an initial leftward step.

To be a bit more concrete, we can find an optimal search strategy for the cow by computing, for all situations (ℓ, r, s),

$$\begin{aligned} & Q((\ell, r, s), \mathrm{l}) \\ & \quad = \quad d((\ell, r, s), \mathrm{l}) + \left(1 - \frac{\Pi^i_{-(\ell+1)}}{\nu^i(\ell, r, s)}\right) V^i((\ell+1, r, 0)) \\ & Q((\ell, r, s), \mathrm{r}) \\ & \quad = \quad d((\ell, r, s), \mathrm{r}) + \left(1 - \frac{\Pi^i_{r+1}}{\nu^i(\ell, r, s)}\right) V^i((\ell, r+1, 1)) \\ & V^i((\ell, r, s)) = \min(Q((\ell, r, s), \mathrm{l}), Q((\ell, r, s), \mathrm{r})) \qquad (2) \end{aligned}$$

where $\nu^i(\ell, r, s)$ is defined by Equation 1, and d is the action-distance function defined by $d((\ell, r, s), \mathrm{l}) = (r + \ell)s+1$ and $d((\ell, r, s), \mathrm{r}) = (r+\ell)(1-s)+1$. The optimal search strategy for the cow is, in situation (ℓ, r, s), to move left to an unexplored location if $Q((\ell, r, s), \mathrm{l}) < Q((\ell, r, s), \mathrm{r})$ and right otherwise.

The resulting algorithm computes optimal expected numbers of steps for each situation in a bottom-up fashion. Because the total number of situations is $\Theta(n^2)$, the run time of the algorithm is quadratic in n.

Minimizing the Worst-Case Competitive Ratio

In this section, we consider the problem of finding a search strategy for the cow to minimize its worst-case competitive ratio given that there are multiple possible probability distributions $\Pi^1, \ldots, \Pi^m$ according to which the pasture location might be distributed. This scenario is intended to model the situation in which an online searcher possesses some information about where it should look, but is uncertain of precisely how reliable the information is.

The worst-case competitive ratio of a search strategy π (for the cow) is the maximum over all choices of Π^i (by an adversary) of the expected number of steps to find the pasture ("online" performance) divided by the expected number of steps the best searcher would take knowing Π^i in advance ("offline" performance):

$$\max_{i \in \{1, \ldots, m\}} \frac{E_{\Pi^i}[\text{search steps of strategy } \pi]}{E_{\Pi^i}[\text{search steps optimal strategy for } \Pi^i]}.$$

An *optimal* search strategy is one that minimizes the worst-case competitive ratio above.

This definition of optimality is different from simply minimizing the expected number of steps in the worst case because the cow is penalized less severely in the case in which the pasture location might be quite far from the starting location. This has the desirable effect that the optimal search strategy does not change radically as the length of the path (size of the search space) is increased. For example, the strategy of the adversary in the case in which the cow tries to minimize the expected number of steps is roughly to put the pasture as far as possible from the starting location. On the other hand, under the competitive-ratio criterion, the adversary's strategy is roughly to force the cow to change directions as often as possible; this puts the emphasis on the decision of where to turn around instead of on the act of walking.

The cow's optimal search strategy can be randomized. As an example, consider the case in which $n = 1$, $\Pi^1 = \langle 1, 0 \rangle$ (pasture is definitely on the left in location p_{-1}), $\Pi^2 = \langle 0, 1 \rangle$ (pasture is definitely on the right in location p_1). The only deterministic strategy is for the cow to take one step in one direction and then to take two steps in the other direction. The worst-case competitive ratio for this strategy is 3, because, in this case, the adversary can always place the pasture in the last place the cow looks. On the other hand, if the cow flips an unbiased coin to decide which direction to move in first, the adversary cannot tell in advance where the cow will search last. The worst-case competitive ratio for this randomized strategy is 2, because the adversary is only able to force the cow to take 3 steps half the time and the rest of the time the cow finds the pasture in a single step.

Solution Methodology

The problem of finding a search strategy to minimize the worst-case competitive ratio is equivalent to a type of two-player zero-sum game. The two players are the cow and the adversary; the cow chooses a search strategy and the adversary chooses which of the m probability distributions to use for the pasture location. There are efficient computational approaches to solving many types of games including matrix games, finite-horizon Markov games, and incomplete information game trees. We will show that the standard computational approaches for these problems do not apply directly to the informed-cow cow-path problem. The remainder of the section shows that a variation of the linear-programming approach to incomplete information game trees can be used to solve the informed-cow cow-path problem in polynomial time.

Given a search strategy and Π^i, it is not difficult to compute the competitive ratio. As a result, the problem of choosing the optimal search strategy for the cow can naturally be cast as a matrix game (Owen 1982). In a matrix game, the set of all deterministic strategies for one player corresponds to the rows of a matrix, and the set of all deterministic strategies for the other player corresponds to the columns. The entries of the matrix give the payoff to the row player for playing the corresponding strategies against one another. Optimal randomized strategies for each player can be determined using linear programming.

To apply the matrix-game approach to the informed-cow cow-path problem, it is necessary to enumerate the set of all deterministic search strategies for the cow; a search strategy is constructed roughly by assigning a choice of "right" or "left" to each of the possible situations. Because each of the $\Theta(2^{n^2})$ deterministic strategies would be included as a row in the game matrix, this approach is computationally infeasible.

In finite-horizon Markov games, the possibly exponential-size set of deterministic strategies is expressed compactly by a set of separate strategic decisions for each state of the game. The obvious candidates for states in the informed-cow cow-path problem are the (ℓ, r, s) situations. As long as states are "Markov" (outcome probabilities are independent of the past), this type of game can be solved efficiently via dynamic programming (Shapley 1953). This can result in exponential savings over the matrix-game approach in the case in which action decisions can be made locally at the level of situations.

The informed-cow cow-path problem doesn't fit into the Markov-game framework because the adversary's choice of Π^i is never revealed to the cow; situations are not Markov. This makes the game one of incomplete information and defeats the use of dynamic programming. Optimization cannot proceed systematically from the bottom up—the values that need to be computed at the bottom of the situation DAG depend on the adversary's choice of Π^i, but the adversary's choice of Π^i is made to give the largest value at the *top* of the situation DAG. The result is a circularity that is not easily broken.

Note that if it were the case that we forced the adversary to choose first, then the cow would be able to select a search strategy knowing which Π^i it was facing. In this case, the dynamic-programming algorithm from the previous section can be applied by computing the best search strategy for each of the m possible choices of Π^i separately.

Framing the informed-cow cow-path problem as a game of incomplete information means we have to consider the complete strategies for both the cow and the adversary simultaneously to find a pair that is in equilibrium. Incomplete information games can be solved in polynomial time in the size of the game tree (Koller, Megiddo, & von Stengel 1994) via linear programming using an interesting representation of strategies based on the "sequence form" of the game. Unfortunately, even when the number of states is small, the game tree can be exponentially large (consider converting the situation DAG of Figure 2 into a game tree; the result is shown in Figure 3).

In the next few sections, we show that any randomized strategy for the cow can be expressed as a vector x with $\Theta(n^2)$ components subject to a polynomial-size set of linear constraints. A randomized strategy for the adversary can be expressed similarly as a vector y of m probabilities related by a set of linear constraints (we consider randomized strategies for the adversary because it simplifies finding a strategy that is in equilibrium with the cow's strategy). The competitive ratio of strategy x against strategy y can be written as $x^T Ry$. Because a game between two linearly constrained vector-represented strategies with a bilinear objective function can be solved in polynomial time using linear programming (Koller, Megiddo, & von Stengel 1994), we can compute an optimal randomized search strategy for the cow in polynomial time.

Evaluating Search Strategies

The set of randomized strategies for the adversary can be expressed quite simply as a vector y of probabilities: y_i is the probability that the adversary will choose distribution Π^i for $1 \le i \le m$. The vector y must satisfy

$$y_i \ge 0 \text{ for all } i, \text{ and } \sum_i y_i = 1 \qquad (3)$$

A natural representation for the cow's randomized strategy is a mapping π from situations and actions (left or right) to probabilities: the value $\pi_{((\ell,r,s),a)}$ represents the probability that the cow will move in direction a when in situation (ℓ, r, s). This type of situation-local randomized strategy is called a "behavior strategy" and is a bit different from the "sweep"-type randomized strategy used in earlier work (Kao, Reif, & Tate 1996). We chose this form for search strategies because it is succinct and simple; however, we show in the appendix that an optimal search strategy can always be expressed in this form.

The vector of probabilities π representing a behavior strategy has an advantageous property that it is easily constrained to represent a valid strategy via linear constraints: $\pi_{((\ell,r,s),a)} \ge 0$ for all (ℓ, r, s) and a, and $\sum_a \pi_{((\ell,r,s),a)} = 1$ for all (ℓ, r, s).

To evaluate the competitive ratio achieved by playing the cow's behavior strategy π against adversary strategy y, we have

$$\sum_{i=1}^{m} y_i \frac{E_{\Pi^i}[\text{search steps of strategy } \pi]}{E_{\Pi^i}[\text{search steps optimal strategy for } \Pi^i]} \qquad (4)$$
$$= \sum_{i=1}^{m} \frac{y_i}{V^i((0,0,0))} E_{\Pi^i}[\text{search steps of strategy } \pi],$$

where V^i is defined in Equation 2. Evaluating the expected number of search steps gives us

$$\begin{aligned}
&E_{\Pi^i}[\text{search steps of strategy } \pi] \\
&= \sum_{(\ell,r,s),a} \Pr((\ell,r,s) \text{ reached, } a \text{ chosen} | \pi, \Pi^i) \\
&\quad \times (\text{steps from } (\ell,r,s) \text{ to next situation via } a) \\
&= \sum_{(\ell,r,s),a} \Pr(\text{pasture not in } (\ell,r,s) | \pi, \Pi^i) \\
&\quad \times \Pr((\ell,r,s) \text{ reached, } a \text{ chosen} | \text{no pasture}, \pi) \\
&\quad \times (\text{steps from } (\ell,r,s) \text{ to next situation via } a) \\
&= \sum_{(\ell,r,s),a} \nu^i(\ell,r,s) x_{((\ell,r,s),a)} d((\ell,r,s),a) \qquad (5)
\end{aligned}$$

where ν^i is defined in Equation 1, d is the distance function used in Equation 2, and $x_{((\ell,r,s),a)} =$

$$\Pr((\ell,r,s) \text{ is reached, } a \text{ chosen} | \text{no pasture}, \pi). \qquad (6)$$

Equation 5 is justified by breaking up the expected number of steps according to the contribution of each edge in the situation DAG.

We discuss x in detail in the next subsection.

Realization Weights

As defined in Equation 6, x is the probability of reaching situation (ℓ, r, s) and taking action a while following behavior strategy π from $(0, 0, 0)$ given that the

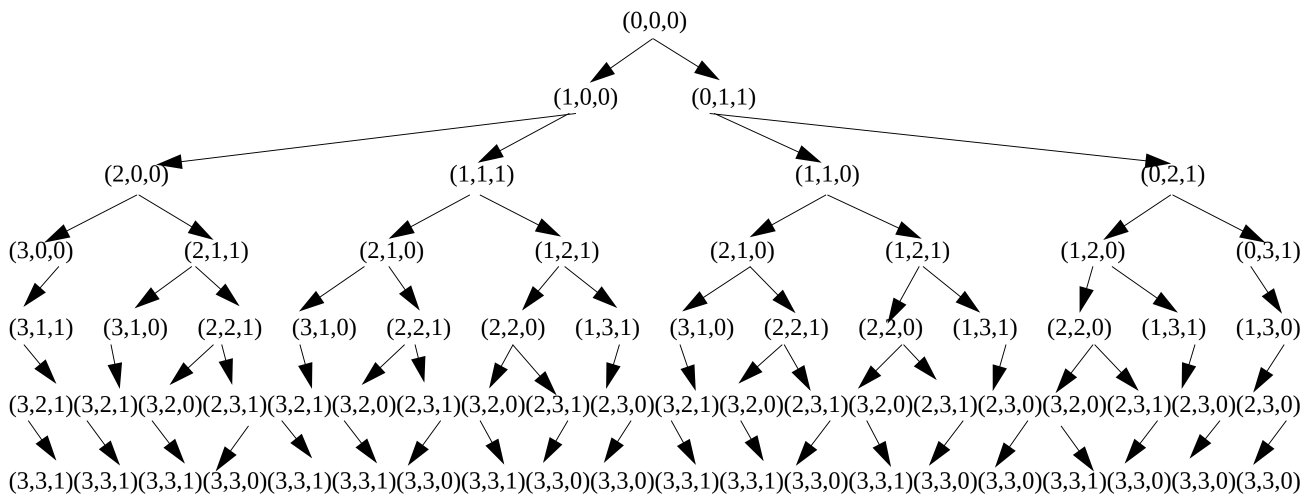

Figure 3: Complete game tree for $n = 3$.

pasture was not found. We can express x in terms of itself and π by

$$x_{((\ell,r,s),a)} = (x_{((\ell_1,r_1,s_1),a_1)} + x_{((\ell_2,r_2,s_2),a_2)})\pi_{((\ell,r,s),a)} \tag{7}$$

where a_1 is the action that causes a transition from situation (ℓ_1, r_1, s_1) to situation (ℓ, r, s) and a_2 is the action that causes a transition from situation (ℓ_2, r_2, s_2) to (ℓ, r, s). Note that any situation (ℓ, r, s) has at most two immediate predecessors: $(\ell-(1-s), r-s, 0)$ and $(\ell-(1-s), r-s, 1)$. Equation 7 states that the probability that action a is taken from situation (ℓ, r, s) is the probability that situation (ℓ, r, s) is reached times the probability that a is taken from (ℓ, r, s).

The components of the x vector are called *realization weights* (Koller, Megiddo, & von Stengel 1994), and they provide an alternative to π for representing the cow's randomized strategy. In particular, note that we can compute π from x by

$$\pi_{((\ell,r,s),a)} = \frac{x_{((\ell,r,s),a)}}{x_{((\ell,r,s),\mathrm{l})} + x_{((\ell,r,s),\mathrm{r})}}. \tag{8}$$

Intuitively, the probability that we choose action a from situation (ℓ, r, s) is the proportion of the probability assigned to action a compared to the total probability of taking an action from situation (ℓ, r, s).

For x to specify a valid set of realization weights, we only need to require that for all (ℓ, r, s) and a,

$$\begin{aligned} & x_{((\ell,r,s),a)} \geq 0, \\ & x_{((0,0,0),\mathrm{l})} + x_{((0,0,0),\mathrm{r})} = 1, \text{ and} \\ & x_{((\ell_1,r_1,s_1),a_1)} + x_{((\ell_2,r_2,s_2),a_2)} \\ & \quad = \; x_{((\ell,r,s),\mathrm{l})} + x_{((\ell,r,s),\mathrm{r})}. \end{aligned} \tag{9}$$

The third constraint in Equation 9 is justified by substituting Equation 8 into Equation 7 for π and dividing through by $x_{((\ell,r,s),a)}$.

This shows that the vector x of realization weights is equivalently expressive to the behavior strategy π and that, like π, x can be constrained to represent a valid strategy by a small set of linear constraints (Equation 9). The previous section expressed the competitive ratio of a randomized strategy in terms of x. In the next section, we put together a complete algorithm for finding the optimal competitive ratio.

The Complete Algorithm

Define the matrix

$$R_{((\ell,r,s),a),i} = \frac{\nu^i(\ell,r,s)d((\ell,r,s),a)}{V^i((0,0,0))}; \tag{10}$$

note that the entries of R do not depend at all on the strategies adopted by the cow or the adversary. Substituting Equation 5 into Equation 4 and using the definition of R, we have that the expected competitive ratio of policy x against policy y is

$$\sum_{i=1}^{m} \sum_{(\ell,r,s),a} x_{((\ell,r,s),a)} R_{((\ell,r,s),a),i} \; y_i = x^T R \; y \tag{11}$$

in matrix notation.

Our goal is to compute the optimal competitive ratio, which we can express as $\min_y \max_x x^T R y$ where x is subject to the linear constraints in Equation 9 (which we write as $Ex = e$ and $x \geq 0$) and y is subject to the linear constraints in Equation 3. The value of this expression is equal to the value of the following linear program (which is a special case of linear program for solving incomplete information game trees (Koller, Megiddo, & von Stengel 1994)):

$$\begin{aligned} & \underset{x,\, q}{\text{maximize}} \; -q \\ & \text{s.t. } x^T R - q \leq 0 \end{aligned}$$

$$Ex = e$$
$$x \geq 0.$$

Here, q is a scalar that is related to the dual of the adversary's strategy y.

This linear program has $\Theta(n^2)$ variables and constraints and the optimal value of x is a representation of the cow's optimal search strategy. The time required to create this linear program depends on the time to create E ($\Theta(n^2)$, because it is sparse) and R. Each of the $\Theta(mn^2)$ entries of R is the combination of an entry of ν^i, d, and V^i; by precomputing all the values of ν^i and V^i, the total time to fill in R is $\Theta(mn^2)$. Therefore, the total time required to compute the optimal randomized search strategy for the cow is polynomial.

Discussion and Conclusions

This paper constitutes an early exploration of a scenario that is very important to on-line search: the searcher wants to minimize its search effort in the worst case, but it has gathered some information that constrains where the target of its search can be. In our work, we have chosen to model this combination of a worst-case criterion with an informed expected-case criterion by postulating an adversary that is constrained to place the target according to any one of a finite set of probability distributions. The particular search problem we examined is the classic cow-path problem, although we expect that this adversary model could be used in other search problems.

We have not studied whether there are practical situations in which this "informed searcher" model applies; however, it is interesting that it includes as special cases both the completely informed expected-case model used for decision problems in operations research (Puterman 1994) ($m = 1$), as well as the worst-case model popular in theoretical computer science (Kao, Reif, & Tate 1996) ($m = 2n$, the number of locations, and the adversary has a separate probability distribution for each situation). In addition, the model can express a simple kind of probability interval notation, in which the searcher has bounds on the probability that the search target resides in a given location, with the width of the bound corresponding to the searcher's degree of uncertainty.

The problem we address appears to have some deep connections to the "minimum latency problem" (Blum *et al.* 1994). In this problem, a searcher must visit all nodes in a graph while minimizing the average visit time. The informed-cow cow-path problem with $m = 1$ is like the minimum latency problem on a one-dimensional graph. In fact, the dynamic-programming algorithm we describe above is nearly identical to one described by Blum et al.

There are a number of directions in which the informed-cow cow-path problem can be extended. Instead of having a probability distribution for the location of the one pasture, independent distributions could be defined for each location concerning whether the location contains a pasture. A similar extension can be made to the case in which there are 2 or more pastures with probability distributions on their joint locations. Both of these extensions can be solved easily within the framework developed in this paper.

Another natural extension is to the case in which there are k paths branching off from the starting location (Kao *et al.* 1994); this paper looked at the $k = 2$ case. Larger values of k can be handled using similar techniques; unfortunately, the run times of the algorithms we proposed scale exponentially with k, because the number of reachable situations scales exponentially with k. We are not aware of any approaches that scale more gracefully.

In our work, we choose to define the competitive ratio in one particular way (Equation 4). Generally, competitive ratios have the form

$$\frac{\text{Performance of online algorithm}}{\text{Performance of offline algorithm}},$$

and there are many, many other ways to define it for this problem, depending on the amount of information available to the "offline" algorithm and whether expected values are taken inside or outside the ratio. In fact, even the $m = 1$ case can be redefined using a competitive ratio, in which the offline algorithm knows precisely where the pasture is. We do not know of any well-motivated way to select between these many options, although we feel the definition we chose is the most natural.

An interesting open issue is to understand what happens to the optimal randomized strategy as $n \to \infty$ in the case in which $\Pi^i = \langle 0, \ldots, 0, 1, 0, \ldots, 0 \rangle$ $-n \leq i \leq n$? Conceptually, this is analogous to the classic uninformed cow-path problem, in which there is no limit in the length of the path and the adversary can choose to place the pasture at any location. An optimal randomized search strategy for this problem has been described (Kao, Reif, & Tate 1996), but it is possible that viewing it as a limiting case of searching a finite set of locations could shed additional light on the structure of the optimal strategy.

This paper also describes a modification of an algorithm for efficiently solving incomplete information game trees to a particular game played on a DAG. For this game, our modified algorithm is exponentially faster than the original algorithm. It would be worth understanding whether this modification can be generalized to solve an even wider class of DAG-structured incomplete information games efficiently.

Appendix: Situation-based Behavior Strategies are Optimal

We show in this section that an optimal search strategy for the cow can always be expressed as a situation-based behavior strategy—a mapping from situations to probability distributions over actions. This justifies our use of this compact representation in our worst-case competitive-ratio algorithm.

To show this, we make use of the concept of realization weights, introduced in Equation 7. Note that we can take the situation DAG and unfold it into a tree of all possible histories of the cow; this is the game tree as illustrated in Figure 3. Let x^* be the optimal realization weights for the game-tree representation of the situation space; x^* must be optimal over all possible search strategies because the cow is using its complete history to make action choices—no amount of other information could help it make better decisions.

Define $\mathcal{S}(t)$ to be the situation corresponding to game-tree node t. Define a set of realization weights x over the situation DAG by

$$x_{((\ell,r,s),a)} = \sum_{t \text{ s.t. } \mathcal{S}(t)=(\ell,r,s)} x^*_{(t,a)};$$

$x_{((\ell,r,s),a)}$ is the sum of the realization weights in x^* for every edge in the game tree that corresponds to situation-action pair $((\ell,r,s),a)$. The x vector satisfies the constraints on realization weights in Equation 9 (non-negativity, initial state has probability 1, "flow" conservation), and therefore represents a valid randomized strategy over the situation DAG.

Define R^* to be a matrix of coefficients defined analogously to R in Equation 10; $R^*_{(t,a),i}$ is the contribution to the competitive ratio of the action a edge at game-tree node t when Π^i is the distribution of the pasture location. An important fact is that the components of R^* for a given game-tree node t depend only on the situation corresponding to t, $\mathcal{S}(t)$.

The optimal worst-case competitive ratio for a situation-based behavior strategy cannot be better than the worst-case competitive ratio of x^*, because x^* is optimal over all strategies. We show that, in fact, the worst-case competitive ratio for x on the situation DAG is equal to that of x^* on the full game tree; therefore, restricting attention to situation-based behavior strategies does not sacrifice our ability to represent optimal search strategies.

By a similar argument to the one leading up to Equation 11, the worst-case competitive ratio for x^* can be written

$$\begin{aligned}
&\min_i \sum_{t,a} x^*_{(t,a)} R^*_{(t,a),i} \\
&= \min_i \sum_{t,a} x^*_{(t,a)} R_{(\mathcal{S}(t),a),i} \\
&= \min_i \sum_{(\ell,r,s),a} \; \sum_{t \text{ s.t. } \mathcal{S}(t)=(\ell,r,s)} x^*_{(t,a)} R_{(\mathcal{S}(t),a),i} \\
&= \min_i \sum_{(\ell,r,s),a} R_{((\ell,r,s),a),i} \sum_{t \text{ s.t. } \mathcal{S}(t)=(\ell,r,s)} x^*_{(t,a)} \\
&= \min_i \sum_{(\ell,r,s),a} x_{((\ell,r,s),a)} R_{((\ell,r,s),a),i}.
\end{aligned}$$

The last expression is the worst-case competitive ratio for x over the situation DAG; because this is exactly equal to the optimal worst-case competitive ratio, this implies that there is always a situation-based behavior strategy that is as good as any other possible strategy.

Acknowledgments

We gratefully acknowledge Sven Koenig for helpful conversations on this topic.

References

Baeza-Yates, R. A.; Culberson, J. C.; and Rawlins, G. J. E. 1993. Searching in the plane. *Information and Computation* 106:234–252.

Blum, A.; Chalasani, P.; Coppersmith, D.; Pulleyblank, B.; Raghavan, P.; and Sudan, M. 1994. The minimum latency problem. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing.*

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1995. Planning and acting in partially observable stochastic domains. Technical Report CS-96-08, Brown University, Providence, RI.

Kao, M. Y.; Ma, Y.; Sipser, M.; and Yin, Y. 1994. Optimal constructions of hybrid algorithms. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, 372–381.

Kao, M.-Y.; Reif, J. H.; and Tate, S. R. 1996. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Information and Computation* 131(1):63–79.

Koller, D.; Megiddo, N.; and von Stengel, B. 1994. Fast algorithms for finding randomized strategies in game trees. In *Proceedings of the 26th ACM Symposium on the Theory of Computing*, 750–759.

Max's Mom and Dad. 1994. *Max's Special Beginning.* `http://www.cs.duke.edu/~mlittman/max/max-book/story.html`.

Owen, G. 1982. *Game Theory: Second edition.* Orlando, Florida: Academic Press.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of Markov decision processes. *Mathematics of Operations Research* 12(3):441–450.

Puterman, M. L. 1994. *Markov Decision Processes—Discrete Stochastic Dynamic Programming.* New York, NY: John Wiley & Sons, Inc.

Shapley, L. 1953. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America* 39:1095–1100.

Sleator, D. D., and Tarjan, R. E. 1985. Amortized efficiency of list update and paging rules. *Communications of the ACM* 28(2):202–208.

Yao, A. C. C. 1980. New algorithms for bin packing. *Journal of the ACM* 27:207–227.