Multi-Step Multi-Sensor Hider-Seeker Games

Erik Halvorson, Vincent Conitzer and Ronald Parr
Duke University Department of Computer Science
{erikh, conitzer, parr}@cs.duke.edu

Abstract

We study a multi-step hider-seeker game where the hider is moving on a graph and, in each step, the seeker is able to search c subsets of the graph nodes. We model this game as a zero-sum Bayesian game, which can be solved in weakly polynomial time in the players' action spaces. The seeker's action space is exponential in c, and both players' action spaces are exponential in the game horizon. To manage this intractability, we use a column/constraint generation approach for both players. This approach requires an oracle to determine best responses for each player. However, we show that computing a best response for the seeker is NP-hard, even for a single-step game when c is part of the input, and that computing a best response is NP-hard for both players for the multi-step game, even if $c = 1$. An integer programming formulation of the best response for the hider is practical for moderate horizons, but computing an exact seeker best response is impractical due to the exponential dependence on both c and the horizon. We therefore develop an approximate best response oracle with bounded suboptimality for the seeker. We prove performance bounds on the strategy that results when column/constraint generation with approximate best responses converges, and we measure the performance of our algorithm in simulations. In our experimental results, column/constraint generation converges to near-minimax strategies for both players fairly quickly.

1 Introduction

Consider a sensor network (the *seeker*) that has been tasked with locating a single, mobile target (the *hider*) given a probability distribution over initial hider locations. How should the hider move to avoid being located? How should the seeker sensors be aimed to maximize the probability of locating the target? These questions are linked: depending on the aims chosen by the seeker, the hider should move differently, and vice versa. Consequently, solving this problem requires using techniques from game theory. This paper models this hider-seeker problem as a two-player, zero-sum Bayesian game, and then searches for an equilibrium.

Zero-sum games can be formulated as linear programs. However, this requires enumerating all of the actions for both players, which can be difficult for our hider-seeker game if the seeker has many sensors or if the game consists of multiple steps of searching. We address this difficulty by using the classical techniques of column and constraint generation. As presented by McMahan et al., [2003], this can be viewed as a "double oracle" algorithm. The algorithm starts with a small set of actions available to each player, solves a restricted game where the players can use only actions in the set, generates new actions for the players by computing best responses to the equilibrium strategies in the restricted game, and repeats until convergence. This algorithm is sound and complete, and is often capable of computing the minimax solution after generating only a small subset of the full action spaces.

The double oracle approach requires best response oracles for both players. While in many games best responses can be computed quickly, we show that, in our hider-seeker game, computing such responses is NP-hard for both players for the multi-step case, and NP-hard for the seeker even for the single-step case. An integer programming formulation of the best response for the hider is practical for moderate horizons, but computing an exact seeker best response is impractical due to the exponential dependence on both the number of sensors available to the seeker and the horizon. Consequently, we develop an approximation algorithm for the seeker best response problem, and use this algorithm in conjunction with the double oracle algorithm to generate an approximate minimax solution.

2 Previous Work

Pursuit-evasion games were first proposed by Isaacs [1965] as a continuous-space problem. Parsons [1976] introduced a discrete formulation which takes place on a graph, and considered the problem of computing the number of pursuers necessary to capture an arbitrarily quick evader who is aware of the locations of the pursuers. Megiddo et al. [1988] proved that computing the smallest number of pursuers (the search number) required to clear a general graph is NP-hard, but can be computed in linear time on a tree. Adler et al. [2003] study the pursuit-evasion problem on a graph where the evader can move arbitrarily, but is unaware of the location of the pursuer.

They develop pursuit strategies which give the pursuer a high probability of capture in a bounded number of steps. Isler et al. [2004] consider a pursuit-evasion problem where both the pursuer and the evader have a limited sight range. They demonstrate that a single pursuer is not sufficient to capture such an evader, but give an algorithm directing two pursuers which provably captures the evader in finite time.

The hider-seeker game discussed in this paper differs from the classical graph-based pursuit-evasion games in two ways. First, the seeker is not constrained to move on the graph, but rather chooses to search c elements from a discrete set of possible observations, each of which is a subset of the vertices. Another difference is that our hider-seeker game has a fixed length. An example of such a game would be a fugitive hider moving through a 2-dimensional grid while a seeker directs several cameras on unmanned aerial vehicles with finite fuel supplies to search for the hider. In this example, the graph would be a regular grid and the observations would correspond to aim points for the cameras on the UAVs. This is a reasonable abstraction of how a group of UAVs could monitor an area by flying in a circular pattern over an area of interest and using pan/tilt cameras to quickly focus on small regions of the larger area.

3 Preliminaries

Our hider-seeker game takes place on a directed graph $G = (V, E)$. In each step, the hider can move from a vertex v to a vertex $v' \in N(v)$, where $N(v) = \{v' \in V | (v, v') \in E)\}$. Let Φ be the set of *aims* available to the seeker, each of which sees some subset $\phi \subset V$ of vertices. In each step, the seeker chooses a subset $\phi^c \subset \Phi$, with $|\phi^c| = c$, of aims to search. Φ_c will denote the set of ϕ^c, i.e., the set of size c subsets of Φ. We assume that there is a prior π (known to both players) over the hider's starting location. The seeker has *captured* the hider in step t if the hider moves into a vertex v' during time step t such that $v' \in \phi$ for one of the seeker's chosen aims in the step t. The length of the game is T steps.

We model this game as a two-player zero-sum Bayesian game. Bayesian games give a general framework representing the players' uncertainty about the availability of actions and/or the payoffs of actions. In general, these each depend on the player's *type*, which is privately held information. Formally, a two-player zero-sum Bayesian game is described by a tuple $(\Theta_1, \Theta_2, A_1, A_2, \alpha_1, \alpha_2, \Pi, u)$, where: Θ_1 and Θ_2 are the *type* spaces of the players; A_1 and A_2 are the sets of actions available to each player; and $\alpha_1 : \Theta_1 \rightarrow 2^{A_1}$ and $\alpha_2 : \Theta_2 \rightarrow 2^{A_2}$ indicate which subsets of actions are available to each player given his type. $\Pi : \Theta_1 \times \Theta_2 \rightarrow [0, 1]$ is a prior over types, and $u : \Theta_1 \times \Theta_2 \times A_1 \times A_2 \rightarrow \Re$ is the game's payoff function (since the game is zero-sum, we only specify player 1's utility function). At the beginning of the game, each player is revealed his own type, which is drawn randomly according to Π. Each player is uncertain of the other player's type.

Our hider-seeker game is a special case of this general framework in which just one player, the hider, has multiple types. Hider types correspond to starting vertices (the only private information that the hider has is his starting vertex), so we use $v \in V$ to denote the hider's type, and the prior Π over types is the same as the prior π over hider starting locations. For each hider type $v \in V$, the available actions to the hider are the set of length T paths starting in v; the full action space for the hider (denoted A_h) is then the set of all length T paths, regardless of starting location. We denote the set of actions available to the hider given his type v as $\alpha_h(v)$. We use the notation $a_h = (v_0, v_1, \ldots, v_T)$ to refer to paths explicitly. The action space for the seeker (denoted A_s instead of α_s, since the seeker has only one type) is equal to $\Phi_c \times \Phi_c \times \ldots \times \Phi_c = (\Phi_c)^T$. We use $a_s(t) \in \Phi_c$ to denote the aims searched by the seeker in step t if he follows a_s. Similarly, we use $a_h(t)$ to denote the vertex occupied by the hider in step t if he follows action a_h.[1] The utility function u (representing the seeker's utility) is defined as:

$$u(a_s, a_h; v) = \begin{cases} 1 & \text{if } \exists t, \exists \phi \in a_s(t) \text{ such that } a_h(t) \in \phi \\ 0 & \text{otherwise} \end{cases}$$

In a Bayesian game, a pure strategy (denoted s_s or s_h) maps types to actions. Rather than computing mixed strategies, which are probability distributions over pure strategies, we will compute a special kind of mixed strategy called a *behavioral strategy*. In a Bayesian game, a behavioral strategy gives the conditional probability that the player plays an action given his type. For the hider, the behavioral strategy $\sigma_h(a_h|v)$ denotes the probability that the hider plays a_h *given* that his type is v. Since the seeker has only a single type, we will omit the conditional notation for his behavioral strategy and use $\sigma_s(a_s)$ to denote the probability that he plays action a_s. The expected utility (to the seeker) is then computed as the expectation over both types and actions. More formally:

$$E[u(\sigma_s, \sigma_h)] = \sum_{v \in V} \pi(v) \cdot \sum_{a_h \in \alpha_h(v)} \sum_{a_s \in A_s} \sigma_s(a_s) \cdot \sigma_h(a_h|v) \cdot u(a_s, a_h; v)$$

Note that with the utility function defined previously, the expected seeker utility is simply the probability that the seeker captures the hider. A *best response* to a mixed strategy is a strategy which maximizes the player's expected utility, given that the other player's strategy is held fixed. Since there are potentially many strategies which maximize a player's expected utility, best responses are generally sets. More formally, the best response sets for the seeker and the hider are:

$$BR_s(\sigma_h) = \operatorname*{argmax}_{\sigma_s} E[u(\sigma_s, \sigma_h)] \text{ and } BR_h(\sigma_s) = \operatorname*{argmin}_{\sigma_h} E[u(\sigma_s, \sigma_h)]$$

Although best responses are defined as mixed strategies, it is a known fact that there exists at least one best response which is a pure strategy, i.e., a single action for the seeker and a single mapping from starting locations to actions for the hider. We wish to find a minimax solution to this game (in two-player zero-sum games, minimax solutions and Nash equilibria are equivalent). That is, we wish to find a pair of mixed strategies (σ_s^*, σ_h^*) such that:

$$\sigma_s^* \in \operatorname*{argmax}_{\sigma_s} \min_{s_h} E[u(\sigma_s, s_h)] \text{ and } \sigma_h^* \in \operatorname*{argmin}_{\sigma_h} \max_{s_s} E[u(s_s, \sigma_h)]$$

3.1 Solving Zero-Sum Bayesian Games

Since the hider-seeker game is zero-sum, a minimax solution can be discovered directly using linear programming. An optimal solution to the LP in Figure 1 gives a minimax solution

[1] We assume that the seeker cannot capture the hider before the hider can move, so that $a_s(0) = \emptyset$ and $a_h(0)$ is the starting vertex for the path a_h. This assumption is not essential for our techniques.

Variables: u
$\forall v \in V, a_h \in \alpha_h(v)$: $\sigma_h(a_h|v)$
Minimize u
Subject to:
$\forall v \in V$ $\sum_{a_h \in \alpha_h(v)} \sigma_h(a_h|v) = 1$
$\forall a_s \in A_s$ $u \geq \sum_{v \in V, a_h \in \alpha_h(v)} \pi(v) \cdot \sigma_h(a_h|v) \cdot u(a_s, a_h; v)$

Figure 1: An LP for the hider-seeker game. The optimal assignments to the $\sigma_h(a_h|v)$ variables give the hider's minimax behavioral strategy while the dual variables for the a_s constraints give the seeker's maximin behavioral strategy.

to the hider-seeker game. This program has $|A_s| + |V|$ constraints and $\sum_{v \in V} |\alpha_h(v)| + 1$ variables. Although there are techniques (such as the Ellipsoid algorithm) which can solve this LP in time which is weakly polynomial in the number of constraints and variables, both $|A_s|$ and $|A_h|$ are exponential in T (and $|A_s|$ is also exponential in c), so solving this program straightforwardly (without column/constraint generation) does not scale well in T and c.[2]

4 Algorithms and Complexity Analyses

This section presents the double oracle algorithm for our hider-seeker game, establishes the computational complexity of determining best responses, presents exact best response oracles for both players, and presents an approximate best response oracle with bounded suboptimality for the seeker.

4.1 Double oracle algorithm

Figure 2 presents the double oracle algorithm for the general hider-seeker game. $BR_s(\sigma_h^{(j)})$ is an oracle that generates a seeker best response to the hider's behavioral strategy $\sigma_h^{(j)}$ and $BR_h(\sigma_s^{(j)}|v)$ is an oracle which generates a hider best response to the seeker's mixed strategy $\sigma_s^{(j)}$, given that the hider's type is v. $MinimaxSolution(F_s, F_h)$ is a subroutine which computes a minimax solution to the restricted game where the seeker can mix only over F_s and the hider can mix only over F_h.

For a two-player, zero-sum game, the double-oracle algorithm corresponds to column and constraint generation, a standard technique for solving large linear programs. See, for example, Bertsimas and Tsitsiklis [1997]. As with generic column and constraint generation techniques, there is no guarantee that this will avoid enumerating all rows and columns (actions for the hider and the seeker, respectively). Empirically, however, such algorithms often converge or produce near-optimal solutions with only a small fraction of the total rows and columns generated.

[2]For two-player zero-sum extensive-form games, there are linear programming techniques that operate directly on the game tree (extensive-form representation), e.g., Koller and Megiddo [1992] or von Stengel [1996]. These techniques are generally much more efficient, but they are not helpful for our hider-seeker game, because neither player ever learns anything about what the other player did previously; the players are essentially making their choices simultaneously. Our linear program has exponential size strictly because the extensive form itself has exponential size.

Double Oracle Algorithm $(G,\ \Phi,\ T)$

// Computes a minimax solution to the length T hider-seeker game on the graph G with aims Φ.
// G : A graph.
// Φ : A set of possible aims.
// T : The length of the game.

$j \leftarrow 0$
$F_s \leftarrow$ { **an arbitrary seeker action** }
$\forall v \in V, F_h(v) \leftarrow$ { **an arbitrary hider action in** $\alpha_h(v)$ }
repeat
  $(\sigma_s^{(j)}, \sigma_h^{(j)}) \leftarrow MinimaxSolution(F_s, F_h)$
  $a_s^{(j)} \leftarrow BR_s(\sigma_h^{(j)})$
  $\forall v \in V, a_h^{(j)}(v) \leftarrow BR_h(\sigma_s^{(j)}|v)$
  if $u(a_s^{(j)}, \sigma_h^{(j)}) = u(\sigma_s^{(j)}, \sigma_h^{(j)})$ **and** $u(\sigma_s^{(j)}, \sigma_h^{(j)}) = u(\sigma_s^{(j)}, a_h^{(j)})$
    return $(\sigma_s^{(j)}, \sigma_h^{(j)})$
  else
    $F_s \leftarrow F_s \cup \{a_s^{(j)}\}$
    $\forall v \in V, F_h(v) \leftarrow F_h(v) \cup \{a_h^{(j)}(v)\}$.
    $j \leftarrow j + 1$

Figure 2: The double oracle algorithm for solving the general hider-seeker game.

4.2 Complexity of the seeker BR problem

In this subsection we show that the general seeker best-response problem is intractable for two different cases, when $T = 1$ and c is part of the problem input, and when $T > 1$ and $c = 1$.

Definition Given G, T, rational z, Φ, c, π (represented as a list of rational numbers), and a hider behavioral strategy $\sigma_h(a_h|v)$ over the actions in A_h, *SeekerBR* asks whether there exists a seeker strategy giving an expected utility of at least z against σ_h. σ_h is represented as a list of only the paths which receive positive probability (representing each path as a list of vertices) and the probabilities they receive as rational numbers. *OptSeekerBR* is the optimization version of SeekerBR which, given the same input (without z), computes an optimal seeker best-response to $\sigma_h(a_h|v)$.

Theorem 1 *SeekerBR is NP-hard with $T = 1$, even when G is a grid and Φ is the set of all $k \times k$ squares.*

Proof The problem is in NP because checking whether a seeker action a_s gives the seeker expected utility at least z can be done in time polynomial in $|V|$ and $|\sigma_h|$ by computing:

$$\sum_{v \in V} \sum_{v' \in N(v): v' \in \phi \in a_s} \pi(v) \cdot \sigma_h((v, v')|v),$$

Thus, SeekerBR is in NP.

To prove NP-hardness on a grid, we reduce from a special case of the rectilinear p-center problem. The p-center problem is defined as: *Given a set of n points P in $\Re^d$, is it possible to place p (closed) "balls" of radius r (in some metric space) such that all the points in P are contained in at least one ball?* This problem is NP-complete (so long as p is part of the input) even when $d = 2$ and the metric is L_∞ [Fowler *et al.*, 1980].[3] The problem remains intractable even if the points are required to have integer coordinates. We will reduce this version of p-center to SeekerBR.

[3]Note that a $\Re^2$ "ball" in L_∞ is a square in the plane.

Given a set P of n points which are on an integer lattice, translate the points so that they have positive x and y coordinates (note that this cannot change the answer to the original instance of p-center) and that the minimum x and y coordinates are zero. Now compute the maximum x and y coordinates and create a $x_{\max} \times y_{\max}$ grid. In this grid, every node (x, y) is connected to itself, $(x \pm 1, y)$ and $(x, y \pm 1)$ (except on the boundary). This grid is guaranteed to be polynomial in size for instances of the rectilinear p-center problem generated by the original reduction from 3-SAT [Fowler *et al.*, 1980]. Define the prior to be:

$$\pi(x, y) = \begin{cases} \frac{1}{n} & \text{if } (x, y) \in P \\ 0 & \text{otherwise} \end{cases}$$

The set of aims Φ consists of all $(2r + 1) \times (2r + 1)$ squares which fit completely in the grid and contain a grid square (x, y) with $\pi(x, y) > 0$. Therefore, $k = 2r + 1$. Let $c = p$ and $z = 1$. Define $\alpha_h(v)$ to be the set of pairs (v, v') in the grid such that v' is adjacent to v and $\sigma_h((v, v)|v) = 1$. In other words, the hider's mixed strategy is to stand still with probability 1.

Suppose there exists a solution the original problem. Thus it is possible to place p balls of radius r in the plane to cover all the points in P. Each of these balls is a closed square of radius r and, since the points in P have integer coordinates, each ball can cover at most the points contained by a $k \times k$ square. Thus each ball can be converted into a square covering at least the same points, and since Φ contains all squares of size $k \times k$, there exists a subset capturing the hider with probability 1. Thus there is a "yes" solution to the instance of SeekerBR, as this subset corresponds with a seeker strategy giving expected utility 1.

Now suppose that there is a seeker strategy giving expected utility 1. This strategy will search $c = p$ squares of size $k \times k$ and each grid square (x, y) with $\pi(x, y) > 0$ will be contained in one of the searched squares. Since each of the grid squares with $\pi(x, y) > 0$ corresponds to a point in P, placing radius r balls at the centers of the searched aims will cover the exact same points. Thus, if there is a "yes" solution to the instance of SeekerBR, there is also a "yes" solution to the original instance of p-center. □

We now show that SeekerBR is intractable even with a single sensor if $T > 1$.

Theorem 2 *SeekerBR is NP-complete, even if $c = 1$, and all aims are of size 1.*

Proof The reduction is from SAT. Given a CNF boolean formula over n literals and clauses Q, construct the graph G as follows: For each literal x_i, create the nodes x_i, $\bar{x}_i$ and H_i. Connect each of x_i, $\bar{x}_i$ and H_i to each of x_{i+1}, $\bar{x}_{i+1}$ and H_{i+1}. Additionally, create a starting node S which is connected to x_1, $\bar{x}_1$ and H_1. See Figure 3 for an example graph.

The set of hider actions is then built from the clauses. For each clause $q \in Q$, create a path starting at S and gradually moving through the x_i, $\bar{x}_i$ and H_i nodes in the following manner. In step 1, if x_1 appears in clause q, the path follows the edge from S to the vertex x_1. If $\bar{x}_1$ appears in q, the path instead follows the edge between S and $\bar{x}_1$. If neither literal

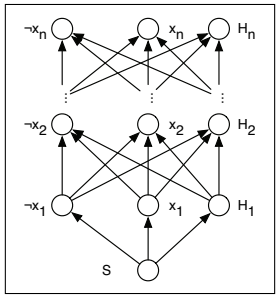

Figure 3: The graph constructed by the seeker BR reduction for a SAT instance with n literals.

appears in q, then the path follows the edge from S to H_1. Similarly, at time i, the path will move to x_{i+1} if x_{i+1} appears in clause q, $\bar{x}_{i+1}$ if $\bar{x}_{i+1}$ appears in q, or H_{i+1} if neither appears in q. The hider's behavioral strategy is then to pick a clause uniformly at random and follow the corresponding path. Set $\pi(S) = 1$, $z = 1$, $c = 1$ and Φ to be the sets of size 1 containing only x_i or $\bar{x}_i$ (for any i).[4]

This instance of SeekerBR has a seeker strategy capturing the hider with probability 1 if and only if the original boolean formula is satisfiable. Suppose that the formula is satisfiable and $\vec{\mathbf{x}}$ is a satisfying assignment. This assignment can be converted into a seeker pure strategy; the strategy will search the vertex x_i in step i if x_i is set to true in $\vec{x}$ and $\bar{x}_i$ if x_i is set to false in $\vec{\mathbf{x}}$. Since $\vec{\mathbf{x}}$ is a satisfying assignment, searching the corresponding vertices will capture each hider path in the graph, as at least one of the path's vertices will be searched.

Now suppose that there exists a seeker strategy that captures the hider with probability 1. This strategy can be converted into a satisfying assignment by performing the opposite conversion: If the seeker searches x_i in step i, then set x_i to true, otherwise (if the seeker searches $\bar{x}_i$) set x_i to false. Suppose (for contradiction) that the assignment does not satisfy the formula and q is a clause that is unsatisfied. Since this clause was converted directly into a hider path, and the hider has some probability of choosing this path, the seeker strategy must search a node on this path to capture the hider with probability 1. Suppose that a hider following the path corresponding to q is captured in step k; the corresponding assignment to x_k must then satisfy the clause q, showing the contradiction.

Also note that SeekerBR is in NP because it is easy to verify that a seeker strategy gets utility of at least z. Specifically, given the seeker strategy $a_s = (a_s(1), a_s(2), \ldots, a_s(T))$,

[4]The set Φ can be extended to include all the vertices in the graph by creating two hidden nodes H_i^1 and H_i^2 for each literal. In this new graph, the hider would decide (randomly) to go to either H_i^1 or H_i^2 every time he moved to H_i in the old graph. This new set of paths can still be represented compactly by representing the randomized transitions as a set of possible successor vertices and the probabilities of moving to each successor. For example, $x_1, \bar{x}_2, \{[H_3^1, 1/2], [H_3^2, 1/2]\}, x_4, x_5, \ldots$ would be one such path with a random transition in step 3.

Variables: $\forall v \in V\ \forall 1 \le t \le T: \quad searched^t(v)$
$\forall a_h \in A_h: \quad cover(a_h)$
$\forall \phi \in \Phi, 1 \le t \le T: \quad aim^t(\phi)$ **(binary)**

Maximize $\sum_{v \in V} \sum_{a_h \in \alpha_h(v)} \pi(v) \cdot cover(a_h) \cdot \sigma_h(a_h|v)$

Subject to:

$\forall a_h \in A_h$	$cover(a_h) \le \sum_{1 \le t \le T} searched^t(a_h(t))$
$\forall 1 \le t \le T, v \in V$	$searched^t(v) \le \sum_{\phi \in \Phi: v \in \phi} aim^t(\phi)$
$\forall 1 \le t \le T$	$\sum_{\phi \in \Phi} aim^t(\phi) \le c$
$\forall 1 \le t \le T, v \in V$	$0 \le searched^t(v) \le 1$
$\forall a_h \in A_h$	$0 \le cover(a_h) \le 1$
$\forall 1 \le t \le T, \phi \in \Phi$	$aim^t(\phi) \in \{0, 1\}$

Figure 4: A mixed integer program for OptSeekerBR. In this MIP, the variable $aim^t(\phi)$ indicates that the strategy includes the aim ϕ in step t, $searched^t(v)$ indicates that at least one of the aims searches the vertex v in step t, and $cover(a_h)$ indicates that a hider using the action a_h would be captured by the seeker best response.

one can verify that:

$$\sum_{1 \le t \le T} \sum_{v \in V} \sum_{a_h \in A_h; \exists t' a_h(t') \in \phi \in a_s(t')} \sigma_h(a_h|v) \cdot \pi(v) \ge z$$

This can be computed in time $O(T|V||\sigma_h|)$, where $|\sigma_h|$ is the size of σ_h on the input tape. Note that T, although exponential in $|T|$ (the size of T in the input), is bounded by the size of a single path in the input tape, thus ensuring that this verification can be accomplished in time polynomial in the size of the input. Thus, SeekerBR is in NP, and the problem is NP-complete. □

4.3 MIP for the seeker BR problem

The seeker BR optimization problem (OptSeekerBR) can be formulated as a mixed integer program. Any optimal solution to the MIP in Figure 4 is a solution to OptSeekerBR. The MIP has three variable types. The $aim^t(\phi)$ variables are explicitly binary, but the MIP solver will also assign binary values to $searched^t(v)$ and $cover(a_h)$ even when they are declared as continuous variables. This is because increasing these variables further will never decrease the objective, and if they can be increased beyond 0, then they can be increased all the way to 1.

4.4 Approximate seeker BR

The MIP for the true seeker BR takes considerable run time in practice, motivating the development of an approximate BR. We present a greedy BR algorithm in Figure 5. The algorithm greedily picks the aim that captures the most hider probability mass. All the paths covered by the chosen aim (and the mass on them) are then removed from consideration, and the process repeats until no more aims are possible.

Theorem 3 *ApproxSeekerBR is a 2-approximation to OptSeekerBR.*

Proof Let G_i be the ith aim chosen by ApproxSeekerBR and let Ξ_i be the set of *new* hider paths captured by G_i (i.e. those not captured by $G_1 \dots G_{i-1}$). Let O be an optimal solution and let O_i be the ith aim in O (ordered such that O_i is a view in the same step as G_i). Let Ξ_i^* be the set of new hider paths

ApproxSeekerBR (σ_h, Φ, π)

// Computes a seeker pure strategy which approximately maximizes his expected utility in response to σ_h.
// σ_h : A hider mixed strategy.
// Φ : The set of aims available to the seeker.
// π : The prior over hider starting positions.

$\Xi \leftarrow \emptyset$
$\forall 1 \le t \le T, a_s(t) \leftarrow \emptyset$
repeat until $\forall 1 \le t \le T, |a_s(t)| = c$
for each t **such that** $|a_s(t)| < c$
$z^*(t) \leftarrow \max_{\phi \in \Phi} \sum_{v \in V} \pi(v) \sum_{\substack{a_h \in \alpha_h(v) \\ \sigma_h(a_h|v) > 0 \\ a_h \notin \Xi \\ a_h(t) \in \phi}} \sigma_h(a_h|v)$
$\phi(t) \leftarrow \text{argmax}_{\phi \in \Phi} \sum_{v \in V} \pi(v) \sum_{\substack{a_h \in \alpha_h(v) \\ \sigma_h(a_h|v) > 0 \\ a_h \notin \Xi \\ a_h(t) \in \phi}} \sigma_h(a_h|v)$
$t^* \leftarrow \text{argmax}_{1 \le t \le T} z^*(t)$
$a_s(t^*) \leftarrow a_s(t^*) \cup \{\phi(t^*)\}$
$\Xi \leftarrow \Xi \cup \{a_h \in \bigcup_{v \in V} \{a_h \in \alpha(v) | \sigma_h(a_h|v) > 0\} | a_h(t^*) \in \phi(t^*)\}$
return $a_s = (a_s(1), a_s(2), \dots, a_s(T))$

Figure 5: An approximation algorithm for OptSeekerBR.

captured by O_i (again, those not captured by $O_1 \dots O_{i-1}$). Let Z^G be the expected seeker utility from following the approximate strategy and Z^* be the expected seeker utility of the optimal BR. In this proof, we will use the shorthand $\sigma_h(\Xi_i)$ (and $\sigma_h(\Xi_i^*)$) to mean:

$$\sigma_h(\Xi_i) = \sum_{v \in V} \pi(v) \cdot \sum_{a_h \in \Xi_i} \sigma_h(a_h|v)$$

$\sigma_h(\Xi_i)$ gives the marginal probability (marginalizing out the hider's starting location) that the hider chooses a path in Ξ_i. Clearly:

$$Z^G = \sum_i \sigma_h(\Xi_i) \text{ and } Z^* = \sum_i \sigma_h(\Xi_i^*)$$

Define the loss to be the difference between Z^* and Z^G:

$$\text{loss} = \sum_i \left[\sigma_h(\Xi_i^*) - \sigma_h(\Xi_i)\right] \le \sum_i \max\{0, \sigma_h(\Xi_i^*) - \sigma_h(\Xi_i)\}$$

Now consider some k such that $\sigma_h(\Xi_k^*) \ge \sigma_h(\Xi_k)$; in other words, an aim contributing to the final summation in the previous statement. When the aim G_k was chosen by ApproxSeekerBR, there was no other aim ϕ which had a larger contribute to $\sigma_h(\Xi_k)$, despite the fact that O_k was available. Thus, enough paths in Ξ_k^* were viewed by $G_1, \dots G_{k-1}$ that G_k appeared as good of an option as O_k. In other words, $\sigma_h(\Xi_k) \ge \sigma_h(\Xi_k^* - (\bigcup_{j<k} \Xi_j))$. Let $\Delta_k = \Xi_k^* \cap (\bigcup_{j<k} \Xi_j)$, i.e. Δ_k is the set of paths in Ξ_k^* that were viewed by previous aims in the greedy algorithm. Now, since $\sigma_h(\Xi_k) \ge \sigma_h(\Xi_k^* - (\bigcup_{j<k} \Xi_j))$,

$$\begin{aligned} \text{loss} &\le \sum_i \max\{0, \sigma_h(\Xi_i^*) - \sigma_h(\Xi_i)\} \\ &\le \sum_i \max\{0, \sigma_h(\Xi_i^*) - \sigma_h(\Xi_i^* - (\bigcup_{j<k} \Xi_j))\} \end{aligned}$$

Now the second quantity in the max is exactly equal to $\sigma_h(\Delta_i)$. Inserting this:

$$\text{loss} \le \sum_i \max\{0, \sigma_h(\Delta_i)\} = \sum_i \sigma_h(\Delta_i)$$

In the above equation, the max disappears because $\sigma_h(\Delta_i)$ is strictly positive. Since each Ξ_i^* is disjoint (from the other Ξ_j^*), Δ_i is also disjoint from the other Δ_j. Thus each hider path appears in at most one Δ_i and, since each is a subset of $\bigcup_i \Xi_i$, we have the following:

$$\text{loss} \leq \sum_i \sigma_h(\Delta_i) \leq \sum_i \sigma_h(\Xi_i)$$

Rearranging terms:

$$\text{loss} = \sum_i \sigma_h(\Xi_i^*) - \sum_i \sigma_h(\Xi_i) \leq \sum_i \sigma_h(\Xi_i)$$

$$\Rightarrow \sum_i \sigma_h(\Xi_i^*) \leq 2 \sum_i \sigma_h(\Xi_i)$$

Thus, ApproxSeekerBR is a 2-approximation. □

4.5 Complexity of the hider BR problem

The previous several subsections studied the seeker BR problem for the hider-seeker game. This subsection considers the BR problem for the hider. Recall that a hider action is a single path of length T through the graph, and a hider pure strategy is a mapping s_h from hider types (vertices) to actions (paths). A hider best response is a pure strategy which, given a seeker mixed strategy σ_s, minimizes the seeker's expected utility. We show that deciding if there exists a hider strategy that results in less than a given utility for the seeker is NP-complete.

Definition Given a graph $G = (V, E)$, rational z, T, a hider starting location v_0, and a seeker mixed strategy $\sigma_s : A_s \rightarrow [0, 1]$ over the set of seeker actions A_s (with c sensors), HiderBR asks whether there exists a pure hider strategy starting in v that gives the seeker expected utility at most z. σ_s is represented as a list of only the seeker strategies which receive positive probability in σ_s, each of which consists of T size c lists of searches. *OptHiderBR* is the optimization version of HiderBR which, given the same input (without z), computes an optimal hider best-response to σ_s starting in v_0.

Theorem 4 *HiderBR is NP-complete, even if $c = 1$, and Φ contains only sets of size 1.*

Proof The reduction is from MINSAT, which is defined as: *Given a set of m boolean clauses Q over the set of literals $X = \{x_1, x_2, \ldots, x_n\}$, does there exist an assignment to the literals in X such that at most k clauses are satisfied?* Kohli et al. [1994] demonstrated that this problem is NP-complete.

Our reduction is fairly similar to the reduction for the seeker best response problem. Given an instance of MINSAT, create two graph nodes for each literal: x_i^+ and x_i^-, corresponding to setting the literal x_i to true and false. Connect both of the nodes for x_{i-1} to both nodes for x_i. Additionally, create a starting node $v_0 = S$, which is connected to x_1^+ and x_1^-. See Figure 6 for an example graph.

Now convert the clauses into seeker actions. For each clause $q \in Q$, construct an action $a_s(q)$ which searches the assignments to literals which would satisfy the clause. For example, if x_i appears negated in the clause q, $a_s(q)$ would search x_i^- at time i. Note that not all seeker actions will search a vertex at each time step, and also that Φ is the set of vertices in the graph. The seeker's mixed strategy σ_s is to pick a clause uniformly at random and perform the associated search action. Set $z = k/m$.

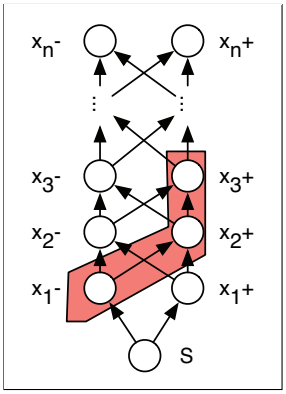

Figure 6: The graph constructed by the hider BR reduction for a MINSAT instance with n literals. Also shown is the seeker searching strategy for the clause $\bar{x}_1 \vee x_2 \vee x_3$.

These two problems are now equivalent for the following reasons. Suppose there exists a solution to this instance of HiderBR; this path will, at each time step i, visit either x_i^+ or x_i^-, and the seeker's probability of capture for this path is at most k/m. Since the seeker's mixed strategy selects actions uniformly at random (i.e., each with probability $1/m$), this path is "caught" by less than k distinct seeker actions. Thus, the corresponding assignment to the literals will satisfy at most k clauses in the set Q. Now suppose that there is an assignment to the literals which satisfies at most k clauses. This assignment can be converted directly into a hider path: Starting at S, go to vertex x_1^+ if x_1 is true in the assignment or x_1^- if x_1 is false, then proceed similarly for the remaining time steps. Since this assignment satisfies at most k clauses in Q, this path will be captured by at most k seeker actions, thus giving the seeker expected utility of at most k/m. Thus, HiderBR is NP-hard.

HiderBR is in NP because it is easy to determine whether a given hider strategy gives the seeker expected utility less than z against σ_s. To verify this, given the hider strategy s_h, simply verify:

$$z \geq \sum_{1 \leq t \leq T} \sum_{v \in V} \sum_{a_s \in A_s : \exists t' s_h(v)(t') \in \phi \in a_s(t')} \sigma_s(a_s) \cdot \pi(v)$$

This can be checked in time $O(Tc|V||\sigma_s|)$, where $|\sigma_s|$ is the size of the representation of σ_s on the input tape. Since even a single seeker strategy in $|\sigma_s|$ is larger than T and c, this can be done in time polynomial in the size of the problem input. Thus, HiderBR is also in NP and is NP-complete. □

4.6 MIP for the hider BR problem

This subsection describes a mixed integer program which can be used for the optimization version of the hider BR problem for the specific case where the hider's starting location is known *a priori*. Recall that, in the hider-seeker game, a hider pure strategy is a mapping from starting vertices to length T paths. This subsection considers the problem of generating a single element of this mapping – the path corresponding to starting in v_0. Solving this problem for all $v \in V$ would yield the full best-response strategy for the hider.

Variables: $\forall a_s \in A_s : \quad caught(a_s)$
$\forall 0 \leq t \leq T, \forall v \in V : \quad in^t(v)$ **(binary)**

Minimize $\sum_{a_s \in A_s} \sigma_s(a_s) \cdot caught(a_s)$

Subject to:

$\forall 0 \leq t < T, v \in V$	$in^t(v) - \sum_{v' \in N(v)} in^{t+1}(v') \leq 0$
$\forall 0 \leq t \leq T$	$\sum_{v \in V} in^t(v) = 1$
$\forall a_s \in F_s, 1 \leq t \leq T,$ $v \in \phi \in a_s(t)$	$caught(a_s) \geq in^t(v)$
$\forall v \in V, 0 \leq t \leq T$	$in^t(v) \in \{0, 1\}$
$\forall a_s \in F_s$	$0 \leq caught(a_s) \leq 1$
	$in^0(v_0) = 1$

Figure 7: A mixed integer program for OptHiderBR. In this MIP, the variables $in^t(v)$ indicate that the hider's path passes through v in time step t and $caught(a_s)$ indicates that the hider is captured by the seeker action a_s.

Figure 7 presents a MIP for OptHiderBR. The first constraint ensures that the path follows the graph's connectivity (i.e., the hider can only go from v to some $v' \in N(v)$), and the second constraint ensures that the hider must be in exactly one location in each time step. The third constraint enforces the relationship between the $in^t(v)$ variables and the $caught(a_s)$ variables. As in the seeker BR MIP, the $caught(a_s)$ variables can be continuous: the mechanics of the MIP will ensure that they always take on binary values. In practice, this MIP solves quite quickly for reasonable values of T.

4.7 Final planning algorithm

Even though the hider BR problem is intractable, the MIP for the hider BR solves quickly enough in practice that it can be used in the double oracle algorithm. For the seeker BR problem, however, we use the approximate algorithm. Figure 8 presents a modified version of the double oracle algorithm that uses these algorithms as subroutines. Using these subroutines as oracles, the double-oracle algorithm converges when the hider's mixed strategy σ_h is a BR to the seeker's mixed strategy σ_s, and the seeker's mixed strategy σ_s gives an expected utility against σ_h that is at least as high as the expected utility obtained by the strategy chosen by ApproxSeekerBR against σ_h. Our final theoretical result bounds the gap between the utility (to the seeker) of the strategies produced by this approximation algorithm, and the utility in a true minimax equilibrium of the game.

Theorem 5 *Let (σ_s, σ_h) be the output of the approximate double oracle algorithm and let (σ_s^*, σ_h^*) be the true minimax solution. Then: $u(\sigma_s, \sigma_h) \geq 1/2 \cdot u(\sigma_s^*, \sigma_h^*)$.*

Proof Using the termination condition of the algorithm, and the fact that ApproxSeekerBR is a 2-approximation to the true BR: $u(\sigma_s, \sigma_h) \geq u(ApproxBR(\sigma_h), \sigma_h) \geq \frac{1}{2} \cdot u(BR(\sigma_h), \sigma_h)$. Because (σ_s^*, σ_h^*) is a minimax solution, $\forall \sigma_s', \sigma_h' : u(\sigma_s^*, \sigma_h') \geq u(\sigma_s^*, \sigma_h^*) \geq u(\sigma_s', \sigma_h^*)$. Combining these gives: $u(BR(\sigma_h), \sigma_h) \geq u(\sigma_s^*, \sigma_h) \geq u(\sigma_s^*, \sigma_h^*)$, which implies $u(\sigma_s, \sigma_h) \geq \frac{1}{2} \cdot u(\sigma_s^*, \sigma_h^*)$. □

5 Experimental results

Our algorithm works for games on arbitrary graphs, but a grid is a more natural representation for many real world scenarios. We used a 100×100 grid, and the prior over initial hider locations was a mixture of truncated Gaussians, each of which covered 21 grid squares. The seeker's set of aims Φ was the set of all 3×3 grid squares. We chose $T = 3$ and ran the double oracle algorithm using both the approximate seeker BR oracle and the true seeker BR oracle for 10 runs, each with a different random seed. For each run, the hider and seeker action spaces were initialized with a single randomly generated action (for the hider, there was one such action for each starting location). We also developed a best response scheduling technique which more evenly divided computation time between the hider and seeker best responses. Rather than generating best responses for both players at the same time, this technique generates a BR for the player whose previous BR value is furthest from the current equilibrium value. This does not affect the optimality guarantees, but we found that it greatly reduced total runtime to convergence.

Approximate Double Oracle Algorithm $(G,\ \Phi,\ T)$

// Computes an approximate minimax solution to the length T hider-seeker game on the graph G with aims Φ.
// G : A graph.
// Φ : A set of possible aims.
// T : The length of the game.

$j \leftarrow 0$
$F_s \leftarrow$ { **an arbitrary seeker action** }
$\forall v \in V, F_h(v) \leftarrow$ { **an arbitrary hider action in** $\alpha_h(v)$ }
repeat
  $(\sigma_s^{(j)}, \sigma_h^{(j)}) \leftarrow MinimaxSolution(F_s, F_h)$
  $a_s^{(j)} \leftarrow ApproxSeekerBR(\sigma_h^{(j)})$
  $\forall v \in V, a_h^{(j)}(v) \leftarrow HiderBRMIP(\sigma_s^{(j)}|v)$
  if $u(a_s^{(j)}, \sigma_h^{(j)}) \leq u(\sigma_s^{(j)}, \sigma_h^{(j)})$ **and** $u(\sigma_s^{(j)}, \sigma_h^{(j)}) = u(\sigma_s^{(j)}, a_h^{(j)})$
    break
  else
    $F_s \leftarrow F_s \cup \{a_s^{(j)}\}$
    $\forall v \in V, F_h(v) \leftarrow F_h(v) \cup \{a_h^{(j)}(v)\}.$
    $j \leftarrow j + 1$
return $(\sigma_s^{(j)}, \sigma_h^{(j)})$

Figure 8: The approximate double oracle algorithm for the hider-seeker game.

Figure 9 shows the results of running the approximate double oracle algorithm in two distinct experiments. In the first experiment (left) we varied the number of observations available to the seeker in each step. In the second experiment (right) we varied the number of Gaussians in the prior from 1 to 7 with $c = 3$. The first set indicates how our approach scales with c, and the second set shows scaling with $|V|$. All reported runtimes are in CPU seconds on a 3.0 GHz Core 2 architecture using CPLEX 10. The suboptimality plots (bottom row) show $u(BR(\sigma_h), \sigma_h) - u(\sigma_s, \sigma_h)$, where (σ_s, σ_h) is the output of the double oracle algorithm using ApproxBR to generate seeker actions. This bounds the seeker's loss from using an approximate BR. We did not compare with the full linear program that contains all hider and seeker strategies because it is impractical to generate and solve such a large LP.

In the first experiment (varying c), Figure 9 (top left) demonstrates that ApproxBR results in a considerable reduction in runtime to convergence. Even for large numbers of

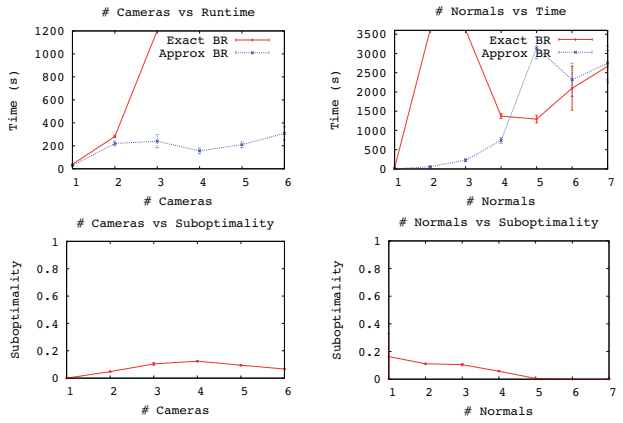

Figure 9: (top left and right) Run time vs. number of observations and number of Gaussians in the prior. Error bars show 1 standard deviation. (bottom left and right) Suboptimality of Approximate Double Oracle Algorithm vs number of observations and number of Gaussians. All experiments on a 100×100 grid with $T = 3$.

observations, the ApproxBR version of the double oracle algorithm converged in minutes, vs. hours when the true BR was used. Figure 9 (bottom left) demonstrates that the utility of the solution is not far from optimal. Suboptimality decreases with c because it is easier to catch the evader with more observations.

In the second experiment (varying the number of Gaussians in the prior), the runtime in Figure 9 (top right) shows widely varying behavior. Unlike the previous experiment, we should expect the run time to scale polynomially for both algorithms, though we should still expect ApproxBR to be faster. The five Gaussian case is an unexpected but reproducible anomaly. Our log files confirm that CPLEX spends significantly less time computing exact BRs for the four Gaussian case than for three Gaussians. The reason for this behavior is a subject of ongoing investigation. The suboptimality, Figure 9 (bottom right), is consistently low and decreasing.

6 Conclusion

This paper presented a multi-step, multi-sensor hider-seeker game where the hider is constrained to move on a graph and, at each time step, the seeker chooses up to a constant number of observations from a set of possible observations. We modeled this game as a two-player zero-sum Bayesian game, and showed that standard techniques for solving the game directly do not scale well with the number of steps or the number of sensors. Consequently, we applied a column and constraint generation technique which uses best response oracles for both players. We also showed, however, that computing the true best response is NP-hard for both the hider and seeker in the multi-step game and NP-hard for the seeker in the single-step game. We then developed an approximation algorithm for the seeker best response and a mixed-integer program for the hider best response. Experimentally, the combined algorithm fairly quickly results in a solution that is near optimal.

7 Acknowledgments

This work was partially supported by the Sloan Foundation, and by NSF IIS awards 0209088 and 0812113, NSF CAREER award 0546709, and by the DARPA CSSG program. Any opinions, findings, conclusions or recommendations are those of the authors only. The authors also wish to thank Pankaj Agarwal for many useful suggestions.

References

[Adler *et al.*, 2003] M. Adler, H. Racke, N. Sivadasan, C. Sohler, and B. Vocking. Randomized pursuit-evasion in graphs. *Combinatorics, Probability and Computing*, 12:225–244, 2003.

[Bertsimas and Tsitsiklis, 1997] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[Fowler *et al.*, 1980] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12(3):133–137, 1980.

[Isaacs, 1965] R. Isaacs. *Differential Games: A Mathematical Theory with Applications to Warfare, Pursuit, Control and Optimization*. Wiley, 1965.

[Isler *et al.*, 2004] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion with limited visibility. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1060–1069, 2004.

[Kohli *et al.*, 1994] R. Kohli, R. Krishnamurti, and P. Mirchandani. The minimum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7(2):275–283, 1994.

[Koller and Megiddo, 1992] D. Koller and N. Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior*, 4(4):528–552, 1992.

[McMahan *et al.*, 2003] H. B. McMahan, G. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *International Conference on Machine Learning (IMCL)*, 2003.

[Megiddo *et al.*, 1988] N. Megiddo, S. Hakimi, M. Garey, D. Johnson, and C. Papadimitriou. The complexity of searching a graph. *JACM*, 35(1):18–44, 1988.

[Parsons, 1976] T. Parsons. Pursuit-evasion in a graph. In Y. Alavi and D. Lick, editors, *Theory and Applications of Graphs*, pages 426–441. Springer-Verlag, 1976.

[von Stengel, 1996] B. von Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2):220–246, 1996.