Sensor Planning as a Game
Ron Parr
CPS 570
based upon

Multi-Step Multi-Sensor Hider-Seeker Games
Erik Halvorson, Vincent Conitzer, Ronald Parr
Duke University - Department of Computer Science

Motivation

- Several unmanned aerial vehicles searching for mobile target
- Where should they aim their cameras?
- How should the target move?
- This problem is circular:
  - Different aims → Hider should move differently
  - Different moves → Seeker should aim differently
- Goal: Find a game-theoretic equilibrium

Hider-Seeker Games

Where should I go to hide?

Where should we search?

Source: HowStuffWorks.com

Hider-Seeker Games

I'll go to F

Search aims 1 and 2...

Source: HowStuffWorks.com
Hider-Seeker Games

Now I'll go to H

Hider Captured!

Search aims 1 and 4...

Where should I go to hide?

Where should we search?

Hider-Seeker Games

I'll play A → F → H

We will search {1,2} then {1,4}

Hider-Seeker Games

I'll play A → F → H

We will search {1,2} then {1,4}
Hider-Seeker Games

I’ll play A → F → H

T = 2

= potential aims

= chosen aims

We will search {1,2} then {1,4}

Hider-Seeker Games

I’ll play A → F → H

Hider gets a utility of -1

T = 2

= potential aims

= chosen aims

Seeker gets a utility of 1

Mixed Strategy Example

I’ll play A → G → H or A → F → C w.p. ½ each

We will search {1,2} then {1,4}

Mixed Strategy Example

I’ll play A → G → H or A → F → C w.p. ½ each

T = 2

= potential aims

= chosen aims

We will search {1,2} then {1,4}
Mixed Strategy Example

We will search \{1, 2\} then \{1, 4\}

Mixed Strategy Example

\[ T = \frac{1}{2} \]

Mixed Strategy Example

\[ T = \frac{1}{2} \]

Solving Two-player Zero-sum Games

• \( \sigma_h \) is a best response (BR) to \( \sigma_h \) if it maximizes the seeker’s expected utility.
  
  \[ BR(\sigma_h) = \max_{\sigma_h} \sum_{s_h \in S_h} \sigma_h(s_h)u(s, s_h) \]

• \( BR(\sigma_h) \) for the hider is defined similarly
  
  • Goal: Find equilibrium strategies \( (\sigma^*_h, \sigma^*_s) \) such that:
    
    \[ \sigma^*_h \in BR(\sigma^*_s) \quad \sigma^*_s \in BR(\sigma^*_h) \]

• Since game is zero-sum, can solve with linear programming

LP for Zero-Sum Games

\[
\begin{align*}
\min & \quad u \\
\text{subject to} & \quad \sum_{s_h \in S_h} \sigma_h(s_h) = 1 \\
& \quad u \geq \sum_{s_h \in S_h} \sigma_h(s_h) \cdot u(s, s_h)
\end{align*}
\]

LP solution contains hider’s mixed strategy as \( \sigma_h(s_h) \) variables
Dual variables give the seeker’s mixed strategy
Difficulties

- LP has $\Theta(|S_h|)$ variables and $\Theta(|S_s|)$ constraints
- Both exponential in $T$
- $|S_s|$ also exponential in $c$
- One solution: Use row and column generation in the LP
- Existing work by McMahan et al. (2003)
- Idea: Start with small subset of strategies
  - Solve game with the restricted set
  - Generate best responses, add to set of strategies
  - Repeat until convergence

Preview

- Seeker and hider BR are both NP-hard, even when seeker is very limited
- Exact hider BR can be computed via a mixed integer program
- Seeker BR can be approximated easily

Double-Oracle Algorithm

- Initialize $F_s$ as an arbitrary, small set of seeker strategies
- Initialize $F_h$ as an arbitrary, small set of hider strategies
- Repeat:
  - $(\sigma_s, \sigma_h) =$ solution to game where seeker can mix over $F_s$ and hider over $F_h$
  - $s_s = BR(\sigma_h)$
  - $s_h = BR(\sigma_s)$
  - If $u(s_s, \sigma_h) = u(\sigma_s, \sigma_h) = u(\sigma_s, s_h)$, return $(\sigma_s, \sigma_h)$
  - Else, $F_s \leftarrow F_s \cup \{s_s\}$ and $F_h \leftarrow F_h \cup \{s_h\}$

Double-Oracle Animation

First LP

$$\begin{align*}
\min \\ u \\
\text{subject to} \\
\sigma_h(s_h) = 1 \\
u \geq \sigma_h(s_h) \cdot u(\sigma_s, s_h)
\end{align*}$$

$S_{h_i}$ $i$th hider strategy generated
$S_{s_i}$ $i$th seeker strategy generated.
Double-Oracle Animation

After one iteration

\[
\begin{align*}
\min \quad & u \\
\text{subject to} \quad & \sigma_h(s_{h_0}) + \\
& u \geq \sigma_h(s_{h_0}) \cdot u(s_{h_0}, s_{h_0}) + \\
& u \geq \sigma_h(s_{h_0}) \cdot u(s_{h_1}, s_{h_1}) + \\
& \vdots + \\
& u \geq \sigma_h(s_{h_0}) \cdot u(s_{h_k}, s_{h_k}) \\
& \sigma_h(s_{h_0}) \cdot u(s_{h_1}, s_{h_1}) + \\
& \vdots + \\
& \sigma_h(s_{h_0}) \cdot u(s_{h_k}, s_{h_k}) \\
& \cdots \\
& = 1
\end{align*}
\]

\(s_{h_0} = i\text{th hider strategy generated}\)

\(s_{h_i} = i\text{th seeker strategy generated}\).

\(s_{h_i} = i\text{th hider strategy generated}\)

\(s_{h_i} = i\text{th seeker strategy generated}\).

\[12/1/15\]

Double-Oracle Algorithm

- For multi-step problem, we need subroutines to compute:
  - Seeker BR to a given hider mixed strategy
  - Hider BR to a given seeker mixed strategy
- We give:
  - \text{Intractability results} for both problems
  - \text{Mixed integer program} for hider BR
  - \text{Approximation algorithm} for seeker BR

\[\text{Double-Oracle Animation}\]
After $k$ iterations

$$\min_{u} \text{subject to} \quad \sigma_h(s_{h_0}) + u \geq \sigma_h(s_{h_0}) \cdot u(s_{s_0_1}, s_{h_0}) + \sigma_h(s_{h_0}) \cdot u(s_{s_2}, s_{h_0}) + \ldots + \sigma_h(s_{h_{k-1}}) \cdot u(s_{s_k}, s_{h_0})$$

$= 1$

**Theorem:**
Seeker BR is NP-hard, even when $T=1$, $G$ is a grid, and the aims are the $k$-by-$k$ squares.

**Proof sketch:**
Reduction from the $p$-center problem.

**Theorem:**
Seeker BR is NP-hard, even if $c = 1$ and the aims consist of only one vertex.

**Proof sketch:**
Reduction from SAT.

**Approximation Algorithm**

I will pick uniformly from:

- $A \rightarrow B \rightarrow E$
- $A \rightarrow C \rightarrow E$
- $A \rightarrow D \rightarrow E$
- $A \rightarrow D \rightarrow F$
- $A \rightarrow F \rightarrow F$

How should I respond?
I can search one vertex per step.

I’ll search $E$ in the second period since it captures hider w.p. 3/5.
Approximation Algorithm

I will pick uniformly from:

A → B  E
A → C  E
A → D  E
A → D  F
A → C  F

I’ll search E in the second period since it captures hider w.p. 3/5.
I’ll search C in the second period since it captures hider w.p. 1/5 (given that I also search D).

My strategy: \( \{C\}, \{E\} \) gives utility 4/5 against \( \sigma_h \)

Approximation Algorithm

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X is a set of already captured paths, initially ( \emptyset )</td>
<td>( \forall t, s_h(t) \leftarrow \emptyset )</td>
</tr>
<tr>
<td>Repeat until all (</td>
<td>s_h(t)</td>
</tr>
<tr>
<td>Add any captured paths to ( X )</td>
<td>( s_h(t^<em>) \leftarrow X \cup {s_h : s_h(t^</em>) \in \phi_{t^*}} )</td>
</tr>
<tr>
<td>Return ( s_h )</td>
<td></td>
</tr>
</tbody>
</table>

This algorithm gives a 2-approximation to the true seeker BR.

Hider BR

After \( k \) iterations
\[
\min u \quad \text{subject to}
\]

\[
\begin{align*}
\sigma_h(s_h) + & \quad \ldots \quad \sigma_h(s_h) \cdot u(s_h, s_h) \quad + \\
\geq & \quad \sigma_h(s_h) \cdot u(s_h, s_h) \quad + \\
\geq & \quad \sigma_h(s_h) \cdot u(s_h, s_h) \quad + \\
\geq & \quad \sigma_h(s_h) \cdot u(s_h, s_h) \quad + \\
\geq & \quad \ldots \quad \ldots \quad \ldots \\
\geq & \quad \sigma_h(s_h) \cdot u(s_h, s_h) + \sigma_h(s_h) \cdot u(s_h, s_h) = 1
\end{align*}
\]
Complexity of Hider BR

- Hider BR = path starting in minimizing probability of capture
- Theorem:
  Given a seeker mixed strategy, computing a path starting in vertex which minimizes the probability of capture is NP-hard, even if \( c = 1 \) and the aims consist of only one vertex.
- Proof Sketch:
  Reduction from MINSAT.

Solving for Hider BR

- Hider BR is intractable
- Can solve with a mixed-integer program
- Empirically reasonable runtime
- We will use this MIP in the double-oracle algorithm

Final Approximate Algorithm

- Initialize \( F_s \) as an arbitrary small set of seeker strategies
- Initialize \( F_h \) as an arbitrary small set of hider strategies
- Repeat:
  - \((\sigma_s, \sigma_h) = \) solution to game where seeker can mix over \( F_s \) and hider over \( F_h \)
  - \( s_s = BR_{approx}(\sigma_h) \)
  - \( s_h = BR_{MIP}(\sigma_s) \)
  - if \( u(s_s, \sigma_h) \leq u(\sigma_s, \sigma_h) = u(\sigma_s, s_h) \), return \( (\sigma_s, \sigma_h) \)
  - Else, \( F_s \leftarrow F_s \cup \{s_s\} \) and \( F_h \leftarrow F_h \cup \{s_h\} \)
- Result is a 2-approximation for the seeker

Generalization: Bayesian Version

- Assumed that hider starting vertex is known
- What if we only have a prior distribution over this?
  - The game becomes a Bayesian game
  - Need to generate hider BRs for each starting vertex
  - Linear program changes slightly
  - Hider strategy is now a behavioral strategy \( \sigma_h(s_h|v) \), giving probability of playing \( s_h \) given that he starts in \( v \)
  - Everything else remains the same
Experimental Results

• Simulated the hider-seeker game on a 100 x 100 grid.
  • Aims are 3 x 3 squares
  • 3 steps (T = 3)
  • Prior distribution is a mixture of truncated normals
• Two experiments:
  • Vary the number of cameras from 1 to 6 with three normal distributions
  • Vary the number of normals in prior from 1 to 6 with three sensors
• Also developed a BR scheduling heuristic

Runtime Plots

Suboptimality plots

Summary

• Hider-seeker games are difficult because of the enormous strategy spaces
• Can be approached using row and column generation techniques
  • Generate strategies for both hider and seeker
  • BR problems intractable for both hider and seeker
  • Hider BR solved with a MIP that solves quickly in practice
  • Greedy approximation algorithm for the seeker BR
• Appears to work well experimentally
Size of the action space

- For \( n \) aims with \( c \) sensors
- \( n \) choose \( c = O(n^c) \) actions
- For \( t \) times steps
- \( (N \text{ choose } c)^t = O(n^{ct}) \)

Why minimax

- Example where minimax is useful even without an adversary
- Suppose you want to guarantee a minimum level of performance in the presence of random events
- Treat nature as an adversary

Best Response vs. Constraint Generation

- Each LP constraint is a bound on one player's utility
- We have one such constraint for each action
- Missing constraints correspond to missing actions
- Most violated constraint = action with highest utility against current opponent strategy = best response
- Best response generation = constraint generation

Is Suboptimality what we Really Care About

- Yes and No
- Yes:
  - When algorithm terminates, we can guarantee performance within a constant factor of optimal
  - This will be true even if we fail to generate all opponent strategies
- No:
  - What we really care about is probability of catching the hider
  - Constant factor can still be pretty bad, e.g., 80% vs. 40%
Why do things get worse, then better?

- Note that vertical axis is just suboptimality, not performance on an absolute scale
- If you have only one sensor, greedy is optimal!
- If you have lots of sensors, hard to be suboptimal!

Example of suboptimality

- Consider a simple chain with a “breakout” node:

  ![Diagram of a simple chain with a breakout node]

  - Should sweep systematically from right to left while guarding the exit
  - Alternatively, a narrow corridor
  - More generally, a building with a few exists

Other questions

- Why does suboptimality decrease as the number of normals increases?
- Why is there that funny kink in the curve?
- Why CG and not some other trick?
- What is more important, demonstrating hardness or finding a good approximation?