Stackelberg Surveillance

Bikramjit Banerjee and Landon Kraemer
School of Computing, The University of Southern Mississippi
118 College Dr. #5106, Hattiesburg, MS 39406, U.S.A
E-mail: Bikramjit.Banerjee@usm.edu,Landon.Kraemer@eagles.usm.edu

Keywords: Stackelberg games, security games, camera surveillance

Received: October 23, 2014

Bayesian Stackelberg game theory has recently been applied for security-resource allocation at ports and airports, transportation, shipping and infrastructure, modeled as security games. *We model the interactions in a camera surveillance problem as a security game, and show that the Stackelberg equilibrium of this game can be formulated as the solution to a non-linear program (NLP). We provide two approximate solutions to this formulation: (a) a linear approximation based on an existing approach (called ASAP), and (b) a hill-climbing based policy search approximation. The first reduces the problem to a single (but difficult) linear program, while the second reduces it to a set of (easier) linear programs. We consider two variants of the problem: one where the camera is visible, and another where it is contained in a tinted enclosure. We show experimental results comparing our approaches to standard NLP solvers.*

Povzetek: V zadnjih letih se v varnostnih nalogah pogosto uporablja metode za iskanje ravnotežja. V prispevku je predstavljena teorija iger na osnovi Bayesa in Stackelberga.

1 Introduction

Bayesian Stackelberg game theory has recently been applied for security-resource allocation at ports and airports, transportation, shipping and infrastructure [7]. Stackelberg games are played by two players: a "leader" and a "follower". In security applications, these players are referred to as "defender" and "attacker" respectively. Typically a defender (leader) acts first by committing to a patrolling/inspection strategy, which is observed by an attacker (follower) of some type. The attacker then plays a best response, such as attack what it predicts to be the weakest/least protected asset, which also determines the defender's payoff besides the attacker's own payoff. The task for the defender is to play its *expected*-payoff-maximizing strategy, knowing the game payoffs (both its and the various types of attackers' payoffs) and the distribution over attacker types. These games are typically non-zero-sum, i.e., one player's loss does not numerically equal the other player's gain. The defender's optimal strategy incorporates randomization because security-resources are typically limited, i.e., not every asset can be simultaneously protected. In the rest of this article, we shall employ the security application terminology and refer to the players as "defender" and "attacker", instead of the general leader-follower game theoretic terminology.

In this article, we formulate a camera surveillance problem as a security game. In a typical camera surveillance scenario, a few fixed cameras are located in strategic spots, each with large coverage and concomitantly low resolution, that often fail to give sufficient details of forensic value after a crime. We consider unmanned surveillance, where no active control of the cameras is possible. For instance, in our university campus there are two cameras in each computer lab, yet articles have been stolen and never have the (fuzzy, grainy) footage enabled post facto identification of any perpetrator. The reliance on short focus setting (i.e., low zoom) aims to balance between the amount of data collected and coverage of the surveilled area. Therefore, attempts to solve the problem by increasing the number of cameras increases the amount of data collected (besides cost), or by increasing the resolution of each camera increases the cost and demand on technology for large surveilled areas.

Our goal is to allow cameras to operate at long focus settings (i.e., high zoom) to capture greater details for forensic value. However, this would lead to reduction in space coverage (unless we deploy a lot of cameras to regain coverage, but this would also blow up the amount of collected data). To address this problem, we allow the cameras to *turn* (i.e., move from one pan/tilt setting to another) at a Stackelberg-randomized schedule, regaining coverage *in time*, without increasing the amount of collected data compared to the fixed cameras scenario.

Figures 1, 2 illustrate the problem and our approach. The target scenario is of unmanned video recordings (from fixed cameras) that may be called for a closer look *post facto*, e.g., for a crime investigation after the crime has occurred. Figure 1 shows a snapshot from such a (real) video recording, where a vehicle is identified as the object of interest. However, zooming in post facto (Figure 2) does not help; not only the license plate but also the make/model are not discernible. The alternative envisaged in this article is a Stackelberg optimized schedule of $(pan,\ tilt)$ settings

Figure 1: A snapshot from a security video.

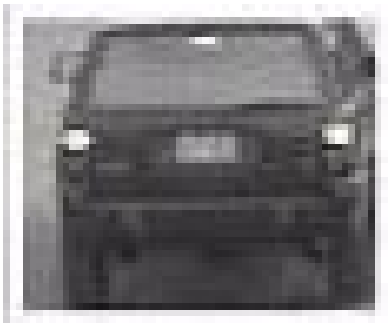

Figure 2: Zooming in to an object of interest in Figure 1 hardly gives any information.

for a camera always operating at a high zoom setting, so that there would be a high likelihood of the camera catching details of the vehicle—thus being of post facto forensic value—even if the vehicle was actively trying to evade it. Instead of actually optimizing likelihoods, we optimize the expected payoff of the camera by assigning differing values to different strategic locations (in an abstract waypoint graph). For instance, in Figure 1, the choke-point (around the bend) could be valued highly, resulting in more frequent coverage of it.

Our formulation of the camera surveillance problem as a security game shows that the Stackelberg equilibrium is given by the solution to a mixed integer non-linear program. We evaluate two first-cut approximate approaches: (a) a linear approximation approach that reduces the mixed integer non-linear program (MI-NLP) to a single (but difficult) mixed integer linear program (MILP), and (b) a policy search approach that generates many mixed integer linear programs that are potentially easier to solve because they have fewer integer variables (and constraints). Our experiments show that indeed the policy search approach is more scalable and produces higher quality solutions, compared not only to (a) but also to some standard NLP solvers.

2 Problem formulation

Henceforth, we will refer to the camera as the "defender" and any target of future interest as the "attacker". We define the camera surveillance problem as a tuple $\langle L, O, T_a, T_d, R_a, R_d \rangle$, where

- L is a set of potential locations of the attacker (i.e., vertices in a waypoint graph),
- O is a set of defender orientations (i.e., discrete pan-tilt settings),
- $T_a(\ell)$ denotes the set of (neighboring) locations that the attacker can reach from location $\ell \in L$ in one step,
- $T_d(o)$ denotes the set of (neighboring) orientations that the defender can reach from orientation $o \in O$ in one step,
- $R_a(\ell, o)$ and $R_d(\ell, o)$ denote the rewards received by the attacker and defender, respectively, when the attacker is in location $\ell \in L$ and the defender orientation is $o \in O$.

We assume discretized time, and that the defender and attacker can change (or not) their current orientation/location simultaneously on each tick. Since the defender wants to cover the current location of the attacker, but the latter wants to evade the defender, $R_d(\ell, o) > 0$ and $R_a(\ell, o) < 0$ iff ℓ is *covered* in orientation o. Otherwise, we assume $R_d = 0$ and $R_a \geq 0$. R_a can also vary according to attacker types. As in general security games, this application is not zero/constant-sum. This is easily seen from the fact that the defender's valuation of assets that it covers may differ from the attacker's, which may further vary by attacker types. For instance, a shoplifter in a superstore may value a flash drive more than a 65" television. On the other hand, a vandal's valuation of a television may be higher than a pricier piece of wood furniture, but negligible for a flash drive. However, it is reasonable to assume that the defender has a fixed valuation for all assets—a departure from traditional security games where the defender's valuations can vary by attacker types. In this article, we assume a single attacker type, but our methodology can be easily extended to multiple attacker types.

Given a problem model $\langle L, O, T_a, T_d, R_a, R_d \rangle$, the goal of the defender is to find a policy that maximizes its expected reward, assuming that the attacker will always play the best response to the defender's policy. Since the defender is unable to observe the attacker's location (it doesn't know who the attacker is), its policy is based on its current orientation only. If it was a deterministic policy that always mapped an orientation to the same neighboring orientation, then an attacker's best response could deterministically allow it to stay out of the defender's view. In fact, as in general security games, this application also presents *resource constraint*, where the resource is the ability to cover waypoints at high enough resolution to be of high forensic value later. Therefore, we use a stochastic policy representation for the defender. It is represented as a real-valued transition function, giving a probability distribution over its next (neighboring) orientation given its current orientation $\zeta : O \times O \mapsto [0, 1]$. The attacker's policy is a Boolean function that gives a mapping from

location-orientation pairs to subsequent (neighboring) locations, $\sigma : L \times O \times L \mapsto \{0,1\}$. Thus we assume that the attacker can observe the defender's current orientation in its decision making.

Although the problem can be considered episodic from the perspective of a particular attacker (with well-defined source and sink vertices in its waypoint graph), from the defender's perspective it is a continuing task, with no horizon, because it never knows the attacker (except after the fact). Since we solve the problem from the defender's perspective with indefinite attacker, we consider the *steady state* of the Markov chain over $L \times O$ occupancies (i.e. the state space) for a given joint policy $\langle \zeta, \sigma \rangle$. Notice that from the defender's perspective the states of the Markov chain are positive recurrent. Now for the particular joint policy $\langle \zeta, \sigma \rangle$, the steady-state probability distribution over $L \times O$ is given by the solution to the following set of recursive equations:

$$P(\ell', o'|\zeta, \sigma) = \sum_{\ell \in L, o \in O} \zeta(o, o')\sigma(\ell, o, \ell')P(\ell, o|\zeta, \sigma). \tag{1}$$

We make a key assumption that the above Markov chain is *irreducible*. This is a reasonable assumption in any surveillance domain because typical "blind spots"—locations that the defender cannot cover (such as restrooms), or the attacker cannot access—can simply be omitted from the attacker's waypoint graph. Thus the system of Equations 1 must have a unique solution.

The attacker's best-response to a particular defender policy ζ, is given by

$$\sigma^{\zeta} = \operatorname*{argmax}_{\sigma} \frac{1}{1-\gamma} \sum_{\ell \in L, o \in O} P(\ell, o|\zeta, \sigma)R_a(\ell, o), \tag{2}$$

where $\gamma \in [0, 1)$ is a discount factor. The defender's goal, then, is to find

$$\zeta^* = \operatorname*{argmax}_{\zeta} \frac{1}{1-\gamma} \sum_{\ell \in L, o \in O} P(\ell, o|\zeta, \sigma^{\zeta})R_d(\ell, o). \tag{3}$$

The above solution based on the steady state is a departure from traditional security games, and the key reason why the resulting program turns out to be non-linear (elaborated at the end of the next section). We formulate the solution program in the next section. In the rest of this article, we will ignore the multiplicative constant involving the discount factor in the objective functions only.

3 Mixed Integer Non-linear Program (MI-NLP)

We use the following variables to formulate the optimization problem for Equation 3:

- $\zeta : O \times O \mapsto [0, 1]$ variables represent the defender's policy, i.e. $\zeta(o, o')$ gives the likelihood that the defender will transition to orientation $o' \in O$ from orientation $o \in O$.
- $\sigma : L \times O \times L \mapsto \{0, 1\}$ represent the attacker's deterministic policy, i.e. if the attacker would transition to location $l' \in L$ from state $(l, o) \in L \times O$, then $\sigma(l, o, l') = 1$.
- $X : L \times O \mapsto [0, 1]$ variables represent the steady state joint occupation probabilities from Equation 1, i.e. $P(\ell, o|\zeta, \sigma)$.
- $v : L \times O \times L \mapsto \Re$ variables that represent the attacker's optimal expected value function for transitioning to location ℓ' from state ℓ, o:

$$v(\ell, o, \ell') = R_a(\ell, o)X(\ell, o) + \gamma \sum_{o' \in T_d(o)} \zeta(o, o') \max_{\ell''} v(\ell', o', \ell'')$$

- $maxV : L \times O \mapsto \Re$ variables represent optimal v,

$$maxV(\ell, o) = \max_{\ell'} v(\ell, o, \ell'). \tag{4}$$

- $maxVE : L \times O \times O \mapsto \Re$ variables represent the products

$$maxVE(\ell, o, o') = \zeta(o, o')maxV(\ell, o'). \tag{5}$$

The linear objective function is

$$\text{Maximize: } \sum_{\ell \in L, o \in O} X(\ell, o)R_d(\ell, o)$$

and the constraints include (in addition to Equation 5)

- the steady state probabilities, $\forall \ell' \in L, o' \in O$

$$X(\ell', o') = \sum_{\substack{\ell \in T_a^{-1}(\ell'),\\ o \in T_d^{-1}(o')}} \zeta(o, o')\sigma(\ell, o, \ell')X(\ell, o) \tag{6}$$

- the linearization of Equation 4, $\forall \ell \in L, o \in O, \ell' \in T_a(\ell)$ and for a large enough M,

$$maxV(\ell, o) \geq v(\ell, o, \ell')$$
$$maxV(\ell, o) \leq v(\ell, o, \ell') + M(1 - \sigma(\ell, o, \ell'))$$

- the resulting linearized v function, $\forall \ell \in L, o \in O, \ell' \in T_a(\ell)$,

$$v(\ell, o, \ell') = R_a(\ell, o)X(\ell, o) + \gamma \sum_{o' \in T_d(o)} maxVE(\ell', o, o')$$

- probability distribution and mutual exclusivity constraints:

$$\forall o \in O, \sum_{o' \in T_d(o)} \zeta(o, o') = 1 \tag{7}$$
$$\sum_{\ell \in L, o \in O} X(\ell, o) = 1$$
$$\forall \ell \in L, o \in O, \sum_{\ell' \in T_a(\ell)} \sigma(\ell, o, \ell') = 1$$

In most security games, the game is assumed to be one-shot, i.e., the game ends for both players when the attacker succeeds or fails. However, in our case the game never really ends for the defender, as the attacker is indefinite. As a consequence, our formulation has a steady state term $X(\ell, o)$, which does not appear in traditional security games. We shall see in the next section that this new term $X(\ell, o)$, particularly its presence in Constraint 6, is the only roadblock to completely linearizing the above MI-NLP. Thus the use of steady state in our solution turns out to be the key reason why our formulation is non-linear.

4 Linear approximation

Note that in the above NLP, all constraints are linear except for Equations 6 and 5. Constraint 6's non-linearity lies in the summands, which are products of three variables $\zeta(o, o')$,$\sigma(\ell, o, \ell')$, and $X(\ell, o)$. In order to focus on the summands individually, we represent each $\zeta(o, o')\sigma(\ell, o, \ell')X(\ell, o)$ summand with a new variable $\beta(\ell, o, \ell', o') \in [0, 1]$ and rewrite constraint 6 as

$$\forall \ell' \in L, o' \in O, X(\ell', o') = \sum_{\substack{\ell \in T_a^{-1}(\ell'), \\ o \in T_d^{-1}(o')}} \beta(\ell, o, \ell', o') \quad (8)$$

and add the constraint

$$\sum_{\ell \in L, o \in O, \ell' \in T_a(\ell), o' \in T_d(o)} \beta(\ell, o, \ell', o') = 1. \quad (9)$$

Now we constrain the $\beta(\ell, o, \ell', o')$ variables without reintroducing non-linearity. First, we note that since $\sigma(\ell, o, \ell')$ is binary, the product $\zeta(o, o')\sigma(\ell, o, \ell')X(\ell, o)$ will be zero when $\sigma(\ell, o, \ell') = 0$, and it will be $\zeta(o, o')X(\ell, o)$ when $\sigma(\ell, o, \ell') = 1$. While the former is relatively simple, the latter is still non-linear.

In determining a practical way to linearize $\zeta(o, o')X(\ell, o)$, it is useful to consider Constraint 5, which also needs to be linearized. Constraint 5 contains the product $\zeta(o, o')maxV(\ell, o')$. Note that if $\zeta(o, o')$ were constant, then both of these expressions would be linearlized. To this end, we invoke the *limited randomization* approach (ASAP) from [4] where the defender's mixed strategy is limited to be integer multiples of $1/k$ for a predetermined integer k. That is, we introduce a set of discrete, constant "snap" points $S = \{\frac{0}{|S|-1}, \frac{1}{|S|-1}, \ldots, \frac{|S|-1}{|S|-1}\}$ and a set of indicator variables $I_S : S \times O \times O \in \{0, 1\}$ constrained as follows: $\forall s \in S, o \in O, o' \in T_d(o)$,

$$\begin{aligned} \zeta(o, o') &\geq s \cdot I_S(s, o, o') \\ \zeta(o, o') &\leq s + M(1 - I_S(s, o, o')) \end{aligned} \quad (10)$$

in addition to: $\forall o \in O, o' \in T_d(o), \sum_{s \in S} I_S(s, o, o') = 1$. Together, these constraints ensure that $\zeta(o, o') \in S$. Note that Constraint 7 is still required to ensure that $\zeta(o, *)$ is a proper distribution.

To demonstrate how this discretization is used, we first address the non-linear Constraint 5. We want to define $maxVE(\ell, o, o')$ to be equal to $s \cdot maxV(\ell, o')$ for the snap point corresponding to $\zeta(o, o')$, i.e. s s.t. $I_S(s, o, o') = 1$. Thus we replace Constraint 5 with the following pair: $\forall s \in S, \ell \in L, o \in O, o' \in T_d(o)$,

$$\begin{aligned} maxVE(\ell, o, o') &\leq s \cdot maxV(\ell, o') + M(1 - I_S(s, o, o')) \\ maxVE(\ell, o, o') &\geq s \cdot maxV(\ell, o') - M(1 - I_S(s, o, o')). \end{aligned}$$

Next we use a similar approach to appropriately constrain the $\beta(\ell, o, \ell', o')$ variables. First, we bound $\beta(\ell, o, \ell', o')$ from above with the following: $\forall \ell \in L, o \in O, \ell' \in T_a(\ell), o' \in T_d(o)$,

$$\begin{aligned} \beta(\ell, o, \ell', o') &\leq \sigma(\ell, o, \ell') \quad (11) \\ \forall s \in S, \beta(\ell, o, \ell', o') &\leq s \cdot X(\ell, o) + M(1 - I_S(s, o, o')). \end{aligned}$$

Then, we bound $\beta(\ell, o, \ell', o')$ from below with: $\forall s \in S, \ell \in L, o \in O, \ell' \in T_a(\ell), o' \in T_d(o)$,

$$\begin{aligned} \beta(\ell, o, \ell', o') \geq s \cdot X(\ell, o) - \\ M[2 - I_S(s, o, o') - \sigma(\ell, o, \ell')] \end{aligned} \quad (12)$$

Finally, while we have related X to a summation over β in Constraint 8, we must add another similar constraint to ensure that $X(\ell, o)$ values are recursively consistent:

$$\forall \ell \in L, o \in O, X(\ell, o) = \sum_{\substack{\ell' \in T_a(\ell), \\ o' \in T_d(o)}} \beta(\ell, o, \ell', o') \quad (13)$$

Thus the MI-NLP reduces to an MILP, whose solution will be an approximation of the exact NLP solution. However, the above ASAP [4] approach, like ASAP itself, incorporates many more integer variables (and constraints) which poses a challenge to linear programming solvers. In the next section, we describe a different approximate approach.

5 Policy search

The above linear approximation of the original NLP requires simultaneous solution of both the defender's and attacker's policies (which are dependent upon each other), which can require significant computation time. In this section, we present a potentially more efficient alternative to this approach.

In a Stackelberg game, it is assumed that the attacker has full knowledge of the defender's policy, and therefore, an attacker's equilibrium policy is an optimal response to the defender's equilibrium policy. On the other hand, the defender's equilibrium policy is optimized with respect to the space of attacker responses. It is important to note that *when the defender's policy is given* (i.e., ζ are constants), the attacker's optimal policy is given by the following (mixed integer) *linear* program:

Algorithm 1 POLICYSEARCH($restarts, \delta$)

```
1: overallMax ← (−∞, ∅)
2: for r = 1 ... restarts do
3:    currentMax ← randomPolicy()
4:    while True do
5:       done ← true
6:       neighbors ←GETNEIGHBORS(currentMax, δ)
7:       for n ∈ neighbors do
8:          solution ← (evaluate(n), n)
9:          if solution_1 > currentMax_1 then
10:            currentMax ← solution
11:            done ← false
12:         end if
13:      end for
14:      if done then
15:         break
16:      end if
17:   end while
18:   if currentMax_1 > overallMax_1 then
19:      overallMax ← currentMax
20:   end if
21: end for
22: Return overallMax
```

$$\text{Maximize:} \sum_{\ell \in L, o \in O} X(\ell, o) R_a(\ell, o)$$

subject to all constraints in the MI-NLP formulation (including Equation 5 which is now linear because ζ is given), except Equation 6. To replace Equation 6, we use the following set of linear constraints: Equations 8, 9, 11, 13, and the following pair: $\forall \ell \in L, o \in O, \ell' \in T_a(\ell), o' \in T_d(o)$,

$$\beta(\ell, o, \ell', o') \leq \zeta(o, o') X(\ell, o) \quad (14)$$

$$\beta(\ell, o, \ell', o') + M(1 - \sigma(\ell, o, \ell')) \geq \zeta(o, o') X(\ell, o) \quad (15)$$

which are again linear for a given ζ.

To leverage this linearity, we separate the defender's decision problem from the (linear) attacker's problem, and perform a hill-climbing search for the former. This search, with multiple restarts, is given in Algorithm 1.

The variables $overallMax$, $currentMax$, and $solution$ in Algorithm 1 are all tuples, specifically pairs, where the second component is the defender's policy, and the first component is its value. For instance, $overallMax$ is initialized to the empty policy with a value of $-\infty$, in Line 1 of Algorithm 1. Subscripted notations in this algorithm, such as $currentMax_1$ in Line 9, refer to the first (value) component of the tuple. The function $evaluate(\zeta)$ returns the objective value of the defender, $\sum_{\ell,o} X^*(\ell, o) R_d(\ell, o)$, where X^* are computed by solving the attacker's MILP for the given defender's policy ζ, as described above in this section. Algorithm 2 specifies one way to generate a limited set of neighboring distributions of a given ζ, with only one dimension perturbed (by δ)

Algorithm 2 GETNEIGHBORS(ζ, δ)

```
1: neighbors ← ∅
2: indices ← T_d(1) × T_d(2) × ... × T_d(|O|)
3: for (i_1, i_2, ..., i_|O|) ∈ indices do
4:    ζ' ← ζ
5:    for o ∈ |O| do
6:       ζ'(o, i_o) ← ζ'(o, i_o) + δ
7:       for o' ∈ |O| do
8:          ζ' ← ζ'(o,o') / (1+δ)
9:       end for
10:   end for
11:   neighbors ← neighbors ∪ {ζ'}
12: end for
13: Return neighbors
```

at a time. Thus the parameter $\delta \in (0, 1]$ controls the distance between a given ζ and its neighbors. Note that the evaluation of neighbors can be executed in parallel. The function $randomPolicy()$ returns a random ζ and solution value, as determined by $evaluate(\zeta)$.

6 Tinted enclosure

We also consider a variant of the surveillance problem where the attacker is unable to observe the current orientation of the defender, perhaps because the camera is contained in a tinted enclosure. This requires the redefinition of $v(\ell, o, \ell')$ as $v(\ell, \ell')$ and simplifies its expression as

$$v(\ell, \ell') = \sum_{o \in O} R_a(\ell, o) X(\ell, o) + \gamma \max_{\ell''} v(\ell', \ell'')$$

Furthermore, $maxV(\ell, o)$ is redefined as $maxV(\ell)$, and the attacker's policy as $\sigma(\ell, \ell')$. Despite these, the key constraint 6 remains cubic, and the above approaches are still applicable. However, the handicap of the attacker means that the defender's policy values can be expected to improve in this setting.

7 Experiments

In the first batch of experiments, we evaluated the linear approximation (ASAP) and Policy Search, over 9 sets of random camera surveillance (non-tinted) problems, the sets defined by increasing number of waypoint vertices of attacker locations, $|L|$, ranging from 2 to 10. Each set contains 20 random problem instances defined by a random edge set in the waypoint graph (single component), and random functions R_a, R_d. In all instances, we restricted $|T_a(\ell)|, |T_d(o)|$ to 3.

For comparison, we also evaluated two standard NLP solvers, Bonmin and SNOPT, and used the (unapproximated) MI-NLP formulation given earlier. Bonmin is a full-fledged MI-NLP solver. SNOPT, on the other hand,

treats integer variables as continuous variables when solving MI-NLPs. All solvers except for SNOPT were run on cluster nodes with 12 processors (a mix of Intel Xeon X5570, X5670, and AMD Opteron processors with up to 12 cores each) and solvers were allowed to use all cores for threading/multiprocessing; however, Bonmin does not support multithreading. SNOPT was run on the NEOS Server [9], and its memory usage was not available.

The Policy Search solver was configured with $\delta = 0.01$ and $restarts = 4$, and the Linear Approximation solver was configured with 25 snap points (i.e. $|S| = 25$). Both of these approaches require MILP solvers, for which we used IBM's ILOG CPLEX. Since the Linear Approximation solver requires solution of a single MILP, we allocated 12 threads to a single CPLEX instance for this solver. On the other hand, since the Policy Search solver requires solution of numerous (but less difficult) MILPs, we allowed for 6 instances of CPLEX to run simultaneously, using 2 threads each. For each problem instance, we measured the expected value of the defender's policy produced by each solver, the amount of wall time (in seconds) required by each solver to find a solution, and the maximum amount of RAM used (in megabytes).

Figure 3 shows the average defender's objective value, runtimes and memory usage for the first 3 sets, $|L| = 2, 3, 4$. The performances of Bonmin and SNOPT clearly show the inadequacy of general purpose MI-NLP solvers for the camera surveillance problem. While the growth in the time and memory requirements of the Policy Search solver is orders of magnitude smaller than Bonmin, it produces higher quality solutions than Bonmin. SNOPT, on the other hand, is more efficient but it produces extremely poor solutions. The performance of the (ASAP based) Linear Approximation solver is clearly dominated by the Policy Search solver, hence we selected the latter for further evaluation. Furthermore, the time and memory requirements of both the Linear Approximation solver and Bonmin were prohibitively high for $|L| > 4$, so we did not run those experiments. For the problem sets $|L| = 5 \ldots 10$, we only compare Policy Search with SNOPT.

Figures 4 (left and middle) show the defender's value and computation times of Policy Search and SNOPT on the remaining problem sets. It is interesting that the computation times of SNOPT remains fairly constant. However the solutions continue to be extremely poor compared to Policy Search. SNOPT performs a general purpose approximation by treating Boolean variables as real. This, apart from its handling of non-linear constraints, is less suited than the more targetted MILP approximation performed by Policy Search.

The second batch of experiments reported in Figure 4 (right) shows the comparative policy values produced by Policy Search on the original (labeled "PS Visible") and the tinted (labeled "PS Tinted") variants of the surveillance problem. As expected, the average defender's policy value improves in the tinted variant and Policy Search is able to significantly pick up this improvement for $|L| > 2$. For comparison, the plot also shows the defender values when ζ is held fixed at the uniform random policy, verifying that Policy Search does produce non-trivial solutions.

8 Related work

Most solution systems for targetted applications of security games, such as ARMOR (LAX airport security [5]), IRIS (federal air marshal service [8]) and GUARDS (transportation security administration [6]) formulate the resource allocation/scheduling problems as (mixed-integer) linear programming problems. Both the defender and the attacker's decision problems are formulated as linear programs. In the camera surveillance problem, although the attacker's decision problem is indeed a mixed integer linear program, the defender's decision problem becomes a non-linear program. Our problem formulation shares many key characteristics of security games, including multiple types of attackers, non-zero-sum utilities, and randomized optimal strategy for the defender, as discussed before.

The attacker's decision problem in this article, that of *stealth navigation*, has been addressed separately before. For instance, [1] considers the navigation of a robot through a field of dynamic obstacles (e.g., beams of search light) without colliding with any. They present a polynomial time algorithm for the case that the robot can move faster than any obstacle. In our time-discretized setting, however, the defender's view and the attacker's location can vary at every time step, hence at the same speed. Therefore, it is likely that no polynomial solution exists even for our attacker's decision problem. More recently, [3] has looked into stealth navigation of a defender, sneaking up on an invader, in order to be detected by the invader as late as possible, ideally no earlier than capture time. They apply heuristic approaches to determine the defender's plan based on a predetermined roadmap (waypoint graph). While we also use a waypoint graph to represent the decision problems, our objective and methodologies are very different.

The defender's decision problem has often been addressed in a distributed constrained optimization setting, such as a *sensor network*, where multiple fixed sensor nodes must coordinate to track (potentially multiple) targets [2, 10]. The sensors are assumed to have significant computational capabilities. By contrast, we consider a single defender and (mobile) attacker, and our approach is readily implementable on existing pan-tilt-zoom cameras since the computation occurs offline. Furthermore, with multiple defenders (cameras), it may become difficult for the attacker to observe their (joint) policies, therefore the attacker's policies may need to be represented in a potentially less interesting way for this application. However, with an appropriate formulation and improved approximation, it may be possible to extend the camera surveillance problem to multiple cameras (defenders) in the future.

Our take on the camera surveillance problem appears unique, and we are unaware of any existing study on the

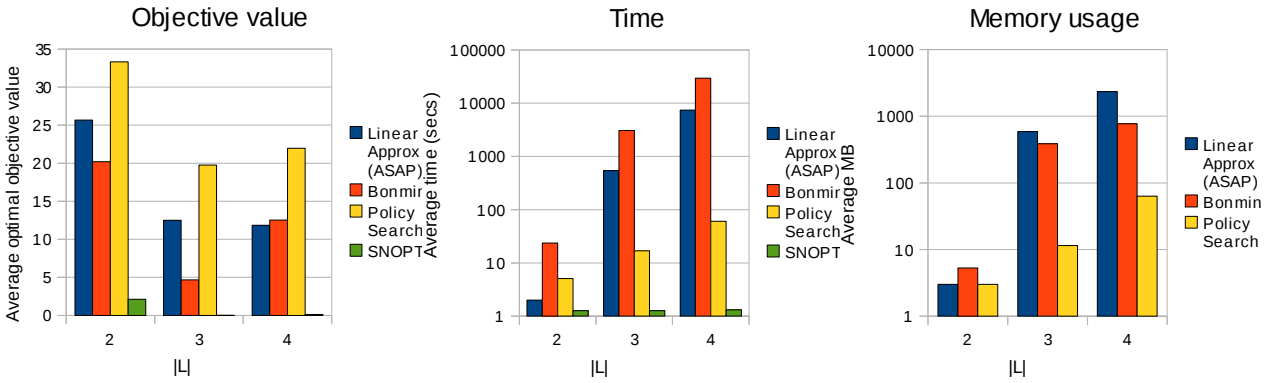

Figure 3: Plots of average (over 20 instances) metrics over the smallest 3 problem sets. (a) shows the defender's objective value only.

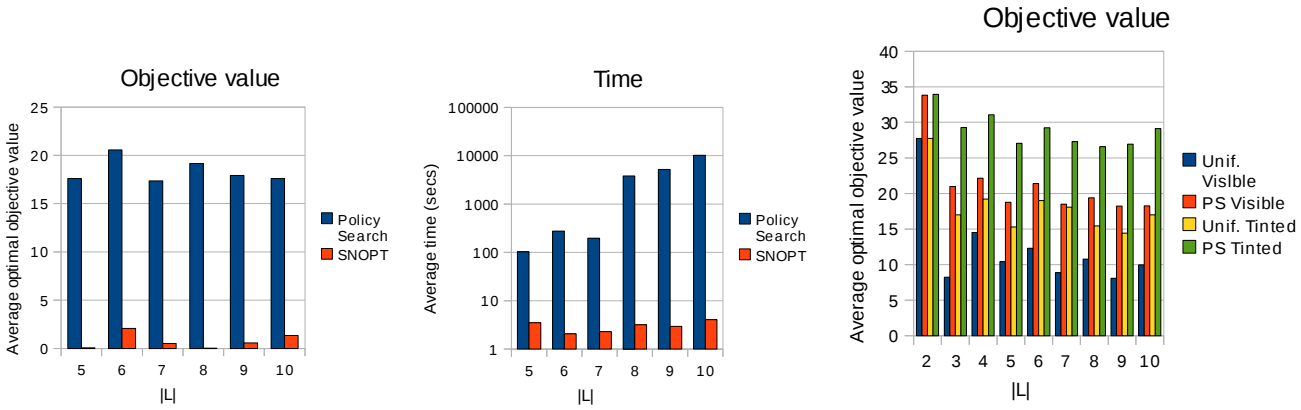

Figure 4: Left & Middle: Comparative solution quality and runtimes of Policy Search and SNOPT. Right: Comparative solution quality of Policy Search on Visible and Tinted formulations, also showing the quality of uniform random defender policy in both formulations.

Stackelberg view of this problem.

9 Conclusions and future work

We have formulated the tradeoff between the amount of collected data and the coverage of a surveillance camera, as a security game between a defender and an (indefinite) attacker. We have shown that this yields a mixed integer non-linear program, on which standard MI-NLP solvers are either prohibitively inefficient or produce poor policies. We have presented two simple approximate solutions: a linear approximation based on an existing aproach (ASAP), and a policy search (hill climbing) approach that leverages the segregation of the defender's decision problem from the attacker's. We have shown experimentally that the policy search approach is more scalable and produces higher quality solutions.

Although we were able to solve instances with up to 10 nodes in the waypoint graph, scaling up to larger and more complex waypoint graphs remains a challenge. One standard technique would be to reduce MILPs into linear programs by assuming that the Boolean variables can be fractional. While this version would be able to scale much better than our current POLICYSEARCH algorithm, it would produce an approximation whose error-bound is unknown at this time. This is a potential avenue for future exploration.

This article lays the foundation for the development of more competent solution approaches in the future, particularly better quality approximations. For instance, a potential avenue for future work is to determine the conditions under which the camera surveillance security game can be posed as a semi-definite program (SDP). It would be interesting to investigate whether standard interior point SDP solvers can produce higher quality solutions more efficiently than Policy Search.

Acknowledgment

The authors thank the anonymous reviewers for helpful comments and suggestions. This work was funded in part by the U.S. Army under grant #W911NF-11-1-0124.

References

[1] K. Fujimura and H. Samet. Planning a time-minimal motion among moving obstacles. *Algorithmica*, 10:41–63, 1993.

[2] V. Lesser, C. Ortiz, and M. Tambe, editors. *Distributed Sensor Networks: A multi-agent perspective*. Kluwer, 2003.

[3] J. Park, J.S. Choi, J. Kim, S. Ji, and B.H. Lee. Long-term stealth navigation in a security zone where the movement of the invader is monitored. *International Journal of Control, Automation and Systems*, 8(3):604–614, 2010.

[4] P. Paruchuri, J.P. Pearce, M. Tambe, F. Ordonez, and S. Kraus. An efficient heuristic approach for security against multiple adversaries. In *Proc. 6th Intl. AAMAS Conference*, 2007.

[5] J. Pita, M. Jain, J. Marecki, F. Ordonez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed ARMOR protection: The application of a game-theoretic model for security at the Los Angeles International Airport. In *Proc. 7th Intl. AAMAS Conference*, 2008.

[6] J. Pita, M. Tambe, C. Kiekintveld, S. Cullen, and E. Steigerwald. GUARDS: Game theoretic security allocation on a national scale. In *Proc. 10th Intl. AAMAS Conference*, 2011.

[7] M. Tambe. *Security and Game Theory*. Cambridge University Press, 2012.

[8] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordonez, and M. Tambe. IRIS-A tool for strategic security allocation in transportation networks. In *Proc. 8th Intl. AAMAS Conference*, 2009.

[9] Website. NEOS Server: State-of-the-art solvers for numerical optimization. http://www.neos-server.org/neos/.

[10] R. Zivan, R. Glinton, and K. Sycara. Distributed constraint optimization for large teams of mobile sensing agents. In *Proc. International Conference on Intelligent Agent Technology (IAT)*, pages 347–354, 2009.