Adversarial Patrolling Games

Yevgeniy Vorobeychik
Sandia National Laboratories
Livermore, CA

Bo An and **Milind Tambe**
University of Southern California
Los Angeles, CA

Abstract

Defender-Attacker Stackelberg games are the foundations of tools deployed for computing optimal patrolling strategies in adversarial domains such as the United states Federal Air Marshals Service and the United States Coast Guard, among others. In Stackelberg game models of these systems the attacker knows only the probability that each target is covered by the defender, but is oblivious to the detailed timing of the coverage schedule. In many real-world situations, however, the attacker can observe the current location of the defender and can exploit this knowledge to reason about the defender's future moves. We study Stackelberg security games in which the defender sequentially moves between targets, with moves constrained by an exogenously specified graph, while the attacker can observe the defender's current location and his (stochastic) policy concerning future moves. We offer five contributions: (1) We model this *adversarial patrolling game (APG)* as a stochastic game with special structure and present several alternative formulations that leverage the general non-linear programming (NLP) approach for computing equilibria in zero-sum stochastic games. We show that our formulations yield significantly better solutions than previous approaches. (2) We extend the NLP formulation for APG allow for attacks that may take multiple time steps to unfold. (3) We provide an approximate MILP formulation that uses discrete defender move probabilities. (4) We experimentally demonstrate the efficacy of an NLP-based approach, and systematically study the impact of network topology on the results. (5) We extend our model to allow the defender to construct the graph constraining his moves, at some cost, and offer novel algorithms for this setting, finding that a MILP approximation is much more effective than the exact NLP in this setting.

Introduction

Game theoretic approaches to security based on Stackelberg game models have received much attention in recent years, with several finding deployment in real-world settings including LAX (Los Angeles International Airport), FAMS (United States Federal Air Marshals Service), TSA (United States Transportation Security Agency), and USCG (United States Coast Guard) (Jain et al. 2010; An et al. 2011). At the backbone of these applications are defender-attacker Stackelberg games in which the defender first commits to a randomized security policy, and the attacker uses surveillance to learn about the policy before attacking. The analysis of Stackelberg security games has focused primarily on computing Strong Stackelberg equilibrium (SSE), i.e., the optimal strategy for the defender (von Stengel and Zamir 2004; Conitzer and Sandholm 2006; Kiekintveld et al. 2009).

To date, the Stackelberg game models for all real-world security applications assume that attacker knows the probability that each target is covered by the defender, but is oblivious to the actual sequence of defender moves. For example, the defender may in fact visit targets according to some fixed (but randomly generated) patrolling schedule, but the attacker is presumed to be unable to observe the defender's location at any point during the patrol. In many realistic settings, such as USCG (An et al. 2011), it is likely that the attacker can in fact observe the patrol while it is in progress (e.g., the coast guard ships can be quite overt). Thus, a more plausible model in such a setting would allow the attacker to observe both the randomized policy of the defender (i.e., probability distribution over moves) as well as current defender location. We formally model this setting as an *adversarial patrolling game*, or APG, and present methods for computing an optimal stochastic patrolling policy for the defender when the planning horizon is infinite and the attacker is impatient (i.e., exponentially discounts future payoffs). Through most of the paper, we additionally assume that the game is strictly competitive: that is, the goal of the defender is to minimize the attacker's expected (discounted) utility.

This paper provides five contributions in the adversarial patrolling setting: (1) We present a formal adversarial patrolling game (APG) model, show that it can be easily cast as a stochastic game (Section), and offer several alternative formulations that leverage the general non-linear programming approach for computing equilibria in zero-sum stochastic games (Filar and Vrieze 1997) (Section). (2) We extend the NLP formulation to capture settings in which attacks can take an arbitrary number of time steps. (3) We provide an approximate MILP formulation that uses discrete defender move probabilities (Section). (4) We additionally offer an experimental comparison that demonstrates that an NLP-based approach is highly efficacious in our setting, and offer a systematic study of the effect of network topology on the efficacy of defense (Section). (5) We present a model of network design in which the defender can first build edges along which patrol decisions are subsequently made. We

present a formal model, as well as algorithms for this setting, and find that a MILP-based approach is much superior to NLP (Section).

Related Work

Some of the earliest work on adversarial patrolling settings was done in the context of robotic patrols, but involved a very simple defense decision space (for example, with a set of robots moving around a perimeter, and a single parameter governing the probability that they move forward or back) (Agmon, Krause, and Kaminka 2008; Agmon, Urieli, and Stone 2011).

More recent work by Basilico *et al.* (Basilico, Gatti, and Amigoni 2009; Basilico et al. 2010; Basilico and Gatti 2011; Bosansky et al. 2011) studied general-sum patrolling games in which they assumed that the attacker is infinitely patient, but the execution of an attack can take an arbitrary number of time steps. However, the resulting formulations rely in a fundamental way on the assumption that both players are infinitely patient, and cannot be easily generalized to handle an impatient attacker. Moreover, Basilico *et al.* only consider a restricted attacker strategy space, and, additionally, their formulation may involve extraneous constraints which result in suboptimal solutions. Indeed, our experiments demonstrate that in the special case of our setting when rewards are undiscounted, our approach yields substantially higher defender utility (see Section). Finally, we consider, in addition to the baseline patrolling problem, two extensions, one involving network design, and another in which patrol moves incur costs. To our knowledge, neither of these extensions has previously been considered in the context of adversarial patrolling.

Adversarial Patrolling

Formally, an *adversarial patrolling game (APG)* can be described by the tuple $\{T, U_d^c(i), U_d^u(i), U_a^c(i), U_a^u(i), \delta, G\}$, where T is the set of n targets patrolled by the defender, $U_d^c(i)$ and $U_d^u(i)$ are the utilities to the defender if an attacker chooses a target $i \in T$ when it is patrolled and not, respectively, while $U_a^c(i)$ and $U_a^u(i)$ are the corresponding attacker utilities, $\delta \in (0, 1)$ is the discount factor (in some cases, we also allow $\delta = 1$), and $G = (T, E)$ is a graph with targets as vertices and E the set of directed edges constraining defender patrolling moves between targets. It is useful to consider the representation of this graph as an adjacency matrix A, where $A_{ij} = 1$ if and only if there is an edge from target i to target j. Below we consider a zero-sum game setting, where $U_d^c(i) = -U_a^c(i)$ and $U_d^u(i) = -U_a^u(i)$.

The game proceeds in a (possibly infinite) sequence of steps in which the defender moves between targets (subject to the constraints imposed by G), while the attacker chooses the time and target of attack. The defender's (stochastic) patrolling policy is a schedule π which can in general be an arbitrary function from all observed history (i.e., the sequence of targets patrolled in the past) to a probability distribution over the targets patrolled in the next iteration. The attacker is presumed to know the defender's policy π at the time of decision. At each time step t the attacker observes the defender's current location i and may choose to wait or to attack an arbitrary target $j \in T$. If an attacker waits, he receives no immediate utility, while attacking a target j gains the attacker $U_a^c(i)$ if it is covered by the defender at time $t + 1$ and $U_a^u(i)$ if it is not. We denote the attacker's policy by a. We say that a policy (π or a) is *Markovian* if it only depends on the current location of the defender, and we call it *stationary Markovian* if it additionally has no dependence on time.

We use v_i to denote the expected discounted value to the attacker upon observing the defender at target i. Where relevant, we assume that the defender always starts at target 0, and the aim of the defender is, consequently, to minimize v_0, which the attacker attempts to maximize.

Example 1. ***USCG's Patrolling Problem as an APG****: USCG safeguards important infrastructure at US coasts, ports, and inland waterway. Given a particular port and a variety of critical infrastructure that an adversary may choose to attack, USCG conducts patrols to detect an adversary and protect this infrastructure. However, while the adversary has the opportunity to observe patrol patterns, limited security resources imply that USCG patrols cannot be at every location at all times (An et al. 2011). In the APG framework, USCG is the defender, while a terrorist group (for example) is an attacker who can conduct surveillance and can both observe the current location of patrols and obtain a good estimate of the stochastic patrolling policy deployed.*

APG as a Stochastic Game

The adversarial patrolling game can be formulated as a *stochastic game* (Filar and Vrieze 1997). A stochastic game is defined by a set of states, a set of players, each taking actions from a finite collection, transition probabilities between states which depend on joint player actions, and, finally, utility (reward) functions of players determined by current state and actions jointly selected by the players.

In our setting, states correspond to the set of targets T, as well as an absorbing state s. Defender actions in each state are the targets j that he can move to in a single time step, while attacker actions are to wait or to attack (for the moment, we will assume that we can compute expected utilities when attacker chooses to attack; we deal with the issue of which targets are attacked below). The state transitions are actually deterministic, conditional on player actions: if the attacker chooses to attack, the system always transitions to the absorbing state s; otherwise, the next target is completely determined by the defender's action. Finally, if the attacker waits, our baseline model involves zero reward accruing to both players. Letting R_i denote the expected utility to attacker of attacking in state i; the defender's utility in the zero-sum model is then $-R_i$. The stochastic game has an infinite horizon, and in our model the attacker's discount factor is δ. Figure 1 offers a schematic illustration of APG as a stochastic game. Since it's a zero-sum game, the defender will aim to minimizes the expected attacker utility (starting from state 0, as we had assumed).

Since the game has an infinite horizon, the policy of the defender can in general be impossible to represent in fi-

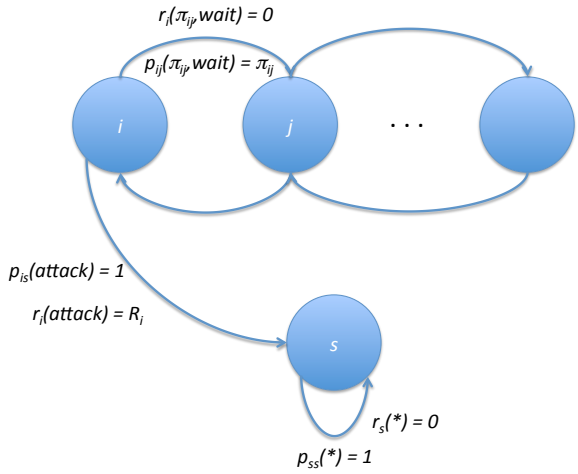

Figure 1: Schematic illustration of APG as a stochastic game, showing example targets-states i and j, as well the absorbing state s. $p_{ij}(\cdot)$ denotes the transition probability, as a function of the probability π_{ij} that the defender moves from i to j and whether or not the attacker chooses "wait" or "attack". Note that if the attacker attacks, the stochastic game transitions to the absorbing state with probability 1, independent of π_{ij}.

nite space. Fortunately, in two-player discounted stochastic games there exists an equilibrium in stationary Markovian policies (Filar and Vrieze 1997). Below we therefore focus exclusively on stationary policies, and denote by π_{ij} the (stationary) probability that the defender moves from target i to j.

Optimal Patrolling Policies on Networks

Partial Formulation of the Defender's Optimization Problem

Since APG is a special case of a stochastic game, we can adapt the non-linear programming formulation for computing a Nash equilibrium in general zero-sum stochastic games by Filar and Vrieze (Filar and Vrieze 1997) to our setting.[1] One minor addition to their formulation that we find useful below is to represent the constraints on the defender's action imposed by the graph G as a set of constraints

$$\pi_{ij} \leq A_{ij} \quad \forall\, i, j \in I. \tag{1}$$

Recalling that R_i is the expected attacker utility from attacking in state i, and v_i is the expected attacker value of starting in state i, we can formulate the defender's problem as the following mathematical program, if we suppose that R_i is known:

$$\min_{\pi, v} \sum_i v_i \tag{2a}$$

s.t. :

$$\pi_{ij} \geq 0 \qquad \forall\, i, j \in T \tag{2b}$$

$$\sum_j \pi_{ij} = 1 \qquad \forall\, i \in T \tag{2c}$$

$$\pi_{ij} \leq A_{ij} \qquad \forall\, i, j \in T \tag{2d}$$

$$v_i \geq R_i \qquad \forall\, i \in T \tag{2e}$$

$$v_i \geq \delta \sum_j \pi_{ij} v_j \qquad \forall i \in T. \tag{2f}$$

Constraints 2b and 2c simply constrain defender policy to be a valid probability distribution. The key constraints 2e and 2f are easiest to think about if we fix defender policy π and just consider the MDP faced by the attacker. The right-hand-side of Constraint 2e corresponds to expected utility of attacking (which is just the immediate reward R_i, since attack moves the MDP into an absorbing state s and no further earnings follow), while right-hand-side of Constraint 2f is the expected value of waiting (immediate reward is 0 for a waiting action). The constraints then arise because the state v_i must be the expected utility of making the best action choice, and minimizing the objective ensures that these values bind to *some* action in every state.

Note that formulation 2 contains non-linear non-convex constraints, and is therefore in general NP-Hard to solve. Typically, algorithms for such problems compute locally optimal solutions, and have no global optimality guarantees. Consequently, both the computation speed, and solution quality are empirical questions, which we explore in experiments below.

Formulation 2 is partial, since it leaves open the question of how R_i are computed. We address this question in two ways below.

Explicitly Representing Attacker Choices

The simplest approach to computing R_i is to explicitly represent the attacker choice of target in each state i, if he chooses to attack in that state (recall that we identify targets with "states"). Let $a_{ij} = 1$ if target j is attacked in state i and $a_{ij} = 0$ otherwise. Past literature on security games has offered a standard approach for computing the optimal attack value using a set of constraints (Kiekintveld et al. 2009):

$$a_{ij} \in \{0, 1\} \quad \forall\, i \in T \tag{3a}$$

$$\sum_j a_{ij} = 1 \quad \forall\, i \in T \tag{3b}$$

$$0 \leq R_i - [(1 - \pi_{ij}) U_a^u(j) + \pi_{ij} U_a^c(j)] \leq (1 - a_{ij}) Z \quad \forall\, i, j \in T \tag{3c}$$

where Z is a very large number. Adding these constraints to the partial formulation 2 gives us the first non-linear program for computing optimal defense strategies in APGs, albeit with integer variables. We refer to this formulation simply as MINLP.

[1] Vital to this adaption is the fact that in zero-sum settings which we focus on below, the distinction between Stackelberg equilibrium and a Nash equilibrium of the corresponding simultaneous-move game is not important.

Implicit Attacker Choices

In the MINLP formulation above, the integer variables arise because we use them to identify the targets chosen by the attacker in each state. Since we are actually interested in the defender's optimization problem, we do not need to know the specific decisions the attacker would make, but only need to compute the attacker's expected value. To do so, let the attacker's set of actions upon observing defender at target i be the union of "wait" and the set of targets j to attack. We can then replace the constraints 2e in the partial formulation 2 with the following set of constraints:

$$v_i \geq (1 - \pi_{ij})U_a^u(j) + \pi_{ij}U_a^c(j) \quad \forall\, i, j \in T.$$

This removes both the variable R_i, and the need for introducing integer variables that explicitly represent the choices of targets to attack. While the resulting program is still non-convex, it now has no integer variables, and, indeed, many fewer variables altogether. We refer to this formulation as NLP.

An important observation about the NLP formulation is that it only involves n non-linear constraints, far fewer than a NLP formulation to compute equilibria in general zero-sum stochastic games. As we demonstrate below, this NLP therefore scales extremely well with the number of targets.

Attacks Taking Multiple Time Steps

An important assumption in the formulations above is that once the attacker chooses to attack, the actual attack commences on the next time step. We now consider a generalization in which attacks take an arbitrary number of steps $h \geq 0$ to unfold. We extend the above NLP formulation to account for this, noting that since the extension can no longer be captured as a stochastic game with targets as states (modulo the absorbing state), and, therefore, it is an open question whether stationary Nash equilibria exist. In the formulation below, we assume that stationary defense policies suffice.

$$\min_{\pi,v,q,qs} \sum_i v_i \tag{4a}$$

s.t. :

$$\pi_{ij} \geq 0 \qquad \forall\, i, j \in T \tag{4b}$$

$$\sum_j \pi_{ij} = 1 \qquad \forall\, i \in T \tag{4c}$$

$$\pi_{ij} \leq A_{ij} \qquad \forall\, i, j \in T \tag{4d}$$

$$q_{ij}^1 = \pi_{ij} \qquad \forall\, i, j \in T \tag{4e}$$

$$q_{ij}^t = \sum_k \pi_{ik} q_{ij}^{t-1} \qquad \forall\, i, j \in T$$

$$t \leq h \tag{4f}$$

$$qs_{ij} = \sum_{t=1}^{h} q_{ij}^t \qquad \forall\, i, j \in T \tag{4g}$$

$$v_i \geq (1 - qs_{ij})U_a^u(j) + qs_{ij}U_a^c(j) \qquad \forall\, i, j \in T \tag{4h}$$

$$v_i \geq \delta \sum_j \pi_{ij} v_j \qquad \forall i \in T. \tag{4i}$$

The formulation is largely the same as above, but involves several additional constraints. Constraints 4e and 4f compute the probability q_{ij}^t that a target j is visited in exactly t steps (for any t), starting from i. Constraint 4g the computes the probability qs_{ij} that j is visited in $1..h$ steps. We then use qs_{ij} in place of π_{ij} to compute attacker utility from attacking in modified Constraint 4h.

Observe that stationarity in the space of targets no longer suffices in general. To see this, consider a simple example with 3 targets (1, 2, and 3), and suppose that there are edges between 1 and 2, and between 2 and 3 only. Finally, suppose that $h = 4$ and the attacker is infinitely patient. Then a non-stationary policy in which the defender moves from 1 to 2 to 3 to 2 to 1 is optimal, since the attacker will always be caught. On the other hand, no stationary policy exists which guarantees that we always catch the attacker: if the defender is at target 2, and the policy is deterministic, he will necessarily leave some target uncovered; if the policy at 2 is stochastic, there is strictly positive probability that the attacker will not be caught.[2]

Mixed Integer Programming Formulation

Even the simplified formulation above is still non-linear non-convex, and while there exist solvers for such problems, there are few guarantees about global solution quality that can be provided: in general, only a local optimum is found. Since we seek a globally optimal solution for the defender, we wish ideally to recast the problem in a form which at least guarantees that globally optimal solutions are approximately achieved.

In this section, we arrive at a MILP (mixed integer linear programming) formulation for an approximate defender optimization problem by discretizing the unit interval into which defense move probabilities will fall into L intervals ($L+1$ allowable probability levels). Let variables $p_l \in [0, 1]$ be the discrete levels in the unit interval with $p_0 = 0$ and $p_L = 1$, and define $d_{ijl} \in \{0, 1\}$ to be binary variables such that $d_{ijl} = 1$ indicates a particular discrete probability choice p_l. We must naturally have a constraint that only one such choice can be made:

$$\sum_l d_{ijl} = 1.$$

Next, we replace π_{ij} with $\sum_l p_l d_{ijl}$ throughout. While constraint 2f remains bi-linear, one of the variables is now binary, and the constraint can therefore be linearized as follows. Define a new set of variables w_{ijl} and let $w_{ijl} = d_{ijl}v_j$. Enforcing the following constraints on w_{ijl} replaces the bi-linear constraint above with an equivalent set of linear constraints:

$$v_j - Z(1 - d_{ijl}) \leq w_{ijl} \leq v_j + Z(1 - d_{ijl}) \tag{5a}$$

$$-Zd_{ijl} \leq w_{ijl} \leq Zd_{ijl}. \tag{5b}$$

Let $R_{ij} = (1 - \sum_l p_l d_{ijl})U_a^u(j) + \sum_l p_l d_{ijl} U_a^c(j)$. We can now rewrite the entire leader optimization program as

[2] We thank Zhengyu Yin for suggesting this example.

a MILP:

$$\min_{d,v,w} \sum_i v_i \tag{6a}$$

s.t. :

$$d_{ijl} \in \{0,1\} \qquad \forall\, i,j \in T, l \in \mathcal{L} \tag{6b}$$

$$\sum_l d_{ijl} = 1 \qquad \forall i,j \in T \tag{6c}$$

$$\sum_j \sum_l p_l d_{ijl} = 1 \qquad \forall\, i \in T \tag{6d}$$

$$\sum_l p_l d_{ijl} \leq A_{ij} \qquad \forall\, i,j \in T \tag{6e}$$

$$v_i \geq R_{ij} \qquad \forall\, i,j \in T \tag{6f}$$

$$v_i \geq \delta \sum_j \sum_l p_l w_{ijl} \qquad \forall i \in T \tag{6g}$$

$$-Zd_{ijl} \leq w_{ijl} \leq Zd_{ijl} \qquad \forall\, i,j \in T, l \in \mathcal{L} \tag{6h}$$

$$w_{ijl} \geq v_j - Z(1-d_{ijl}) \qquad \forall\, i,j \in T, l \in \mathcal{L} \tag{6i}$$

$$w_{ijl} \leq v_j + Z(1-d_{ijl}) \qquad \forall\, i,j \in T, l \in \mathcal{L}, \tag{6j}$$

where $\mathcal{L} = \{0,\ldots,L\}$. Constraints 6d-6g correspond directly to the constraints 2c-2f in the NLP formulation, only with the discrete probabilities replacing π_{ij}. Constraints 6h-6j linearize the bilinear term. We note that there is an analogous MILP formulation in which we explicitly represent the attacked targets, as described in Section . We refer to the MILP above as MILP (reduced), as compared to MILP (baseline) which refers to the latter program; MILP without a modifier refers to the former.

Finite Horizon Problems

An infinite horizon setting that we consider seems at first glance unrealistic: in typical security domains, there is a set time frame for patrols, and at the end of this time, the patrol must return to base, at which point the game restarts. Fortunately, it is not difficult to show that the solution to the infinite-horizon problem is an arbitrarily accurate approximation even if the actual time frame is finite.[3]

Experiments: Patrolling on Exogenous Graphs

In our experimental studies below we use a somewhat simplified model in which $U_a^c(i) = 0$ for all targets $i \in T$. We generate the values of successful attacks $U_a^u(i)$ i.i.d. from a uniform distribution on a unit interval. Throughout, we use $\delta = 0.95$, except where specified otherwise.[4] Finally, we use well-known generative models for networks to generate random instances of graphs over which the defender patrols. The first is an Erdos-Renyi model (Newman 2010) under which every directed link is made with a specified and fixed probability p; we refer to this model by ER(p), or simply ER. The second is Preferential Attachment (Newman 2010), which adds nodes in a fixed sequence, starting from an arbitrary seed graph with at least two vertices. Each node i is attached to m others stochastically (unless $i \leq m$, in which case it is connected to all preceding nodes), with probability of connecting to a node j proportional to the degree of j, d_j. In a generalized version of this model that we consider below, connection probabilities are $(d_j)^\gamma$, such that when $\gamma = 0$ we recover (roughly) the Erdos-Renyi model, $\gamma = 1$ recovers the "standard" PA model, and large values of γ correspond to highly inhomogeneous degree distributions. Finally, we also consider simple Cycles.

When the networks are relatively sparse (like a Cycle), and the number of targets large, the attacker can usually attack the most valuable target at time 0, and not face the tradeoff between the value of time and attack utility that we are trying to model. In our experiments, we therefore connected the starting target 0 to every other target, with network topology effective only on the rest of the targets. Alternatively, we may think of target 0 as a base, and the rest of the targets as initial deployments, which are unconstrained. Since target 0 is a kind of nominal target, we additionally set its utility to the attacker $U_a^u(0)$ to be 0.

All computational experiments were performed on a 64 bit Linux 2.6.18-164.el5 computer with 96 GB of RAM and two quad-core hyperthreaded Intel Xeon 2.93 GHz processors. We did not make use of any parallel or multi-threading capabilities, restricting a solver to a single thread, when relevant. Mixed integer linear programs were solved using CPLEX version 12.2, mixed integer non-linear programs were solved using KNITRO version 7.0.0, and we used IPOPT version 3.9.3 to solve non-linear (non-integer) programs in most cases (the one exception is identified below, where we also used KNITRO).

The results we report are based on 100 samples from both the attacker utility distribution and (when applicable) from the network generation model. Throughout, we report 95% confidence intervals, where relevant.

Comparison to Basilico et al.

Basilico et al. (Basilico, Gatti, and Amigoni 2009) presented a multiple math programming approach to adversarial patrolling for a setting very similar to ours. By setting $\delta = 1$, and reformulating the algorithm in (Basilico, Gatti, and Amigoni 2009) in a zero-sum setting and with a single-step attack, we can make a direct comparison between our algorithm (using the NLP formulation) and theirs. The results, shown in Figure 2, suggest that our approach yields significantly better solutions. The difference becomes less important as the number of targets increases: since in both approaches we only allow for one defender resource (defender can protect at most a single target at a time), and we assign relative values to targets uniformly randomly, on sparse graphs the attacker becomes increasingly likely to get the target he wants when the discount factor is 1, since the defender is eventually at least two hops away from the most valuable target.

[3] Details are at http://aamas.webs.com/appendix_apg.pdf.

[4] We considered other discount factors as well, but this one strikes the right balance: it creates interesting tradeoffs between attacking and waiting, and yet creates a setting that is significantly different from past work which only considers $\delta = 1$.

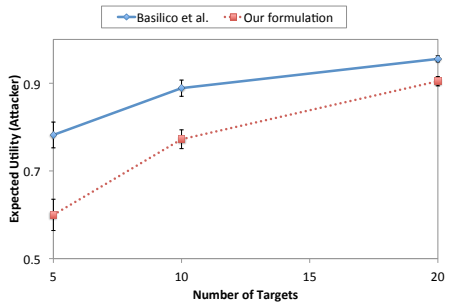

Figure 2: Comparison between our NLP formulation and that developed by Basilico et al. The graph is ER(0.1).

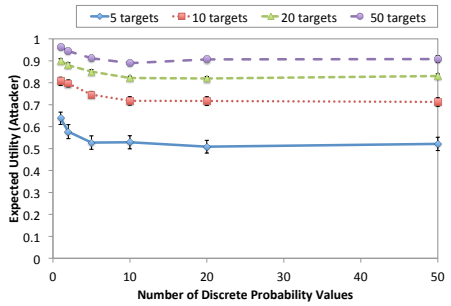

Figure 3: MILP objective value as a function of granularity of discretization in ER(0.1).

MILP Discretization

The size and, consequently, complexity of the MILP depends greatly on the fineness of discretization of the probability interval. While we can, perhaps, presume that a fine enough discretization would get us close to an optimal solution, computationally we cannot in all likelihood afford a very fine discretization. An important question, therefore, is: how much is enough? We address this question by considering a sequence of increasingly fine discretizations, starting at $L = 1$ ($p_0 = 0$ and $p_1 = 1$) and going up to $L = 50$ ($p_l \in \{0, 0.02, 0.04, \ldots, 1\}$). To ensure that whatever we find is not particular to a given setting, we also vary the number of targets between 5 and 50, as well as the network topology (Cycle, Erdos-Renyi, and Preferential Attachment).

The results, shown in Figure 3 for ER(0.1) networks, are quite reassuring: $L = 10$ seems to suffice across all the settings shown, and these results are also consistent with those obtained for Cycle and PA(2,1) networks. From this point on, results based on MILP formulation use $L = 10$, unless otherwise specified.

Comparison of the Alternative Formulations

We offered several alternative formulations of the defender's optimization problem: MINLP (the mixed integer non-linear programming approach in which we explicitly encode attacker target choices), NLP (non-linear program in which attacker target choices are implicit), and two MILPs, the first that does encode target choices, which we call "MILP (baseline)", and the second that does not, and which we refer to as "MILP(reduced)".

We compare all these formulations in terms of objective value (i.e. average v_0 over 100 random realizations of target values and network topologies) and average running time. The results in Figure 4 suggest that there is not a significant difference in efficacy of the programming approaches we propose. Running time, however, does in fact differentiate them. Experimentally we found that MINLP running time diverges rapidly from that of MILP: even with as few as 9 targets, KNITRO solver takes nearly 300 seconds, as compared to under 2 seconds solving the corresponding MILP approximation using CPLEX. Surprisingly, we found little

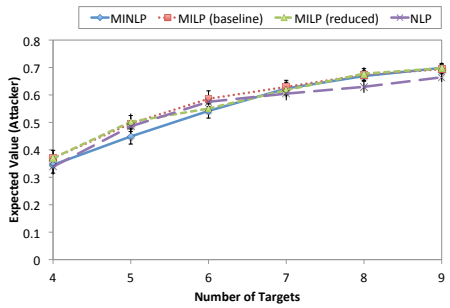

Figure 4: Comparison of average attacker utility achieved using MINLP, two versions of MILP, and NLP formulations, using the Cycle topology.

difference in running time between the two MILP formulations, but the difference between MILP and NLP formulations is rather dramatic. Figure 5 shows that the NLP formulation scales considerably better than MILP, solving instances with as many as 1000 targets in under 200 seconds (MILP already begins to reach its limit by $n = 50$). Interestingly, graph topology seems to play some role in determining the difficulty of the problem: Cycle graphs are solved much faster by NLP than Erdos-Renyi analogs.

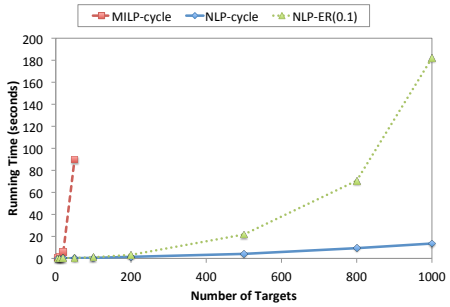

Figure 5: Running time comparison between MILP and NLP on Cycle and ER(0.1) graphs. We omit MINLP which does not scale, and the two MILP formulations yield similar results, so we only present MILP (baseline) here.

The Impact of Network Topology

A perpetually interesting question in network science literature is how the characteristics of a particular process taking

place on a graph are affected by the specifics of graph topology (Newman 2010). In this section we study graph topologies from two perspectives: first, the density (the number of edges relative to number of nodes) of the graph, and second, homogeneity of the degree distribution. The (generalized) Preferential Attachment generative model of networks offers a convenient means to study both of these aspects of networks (Newman 2010), since the parameter m governs the network density, while γ governs the homogeneity of the degree distribution.

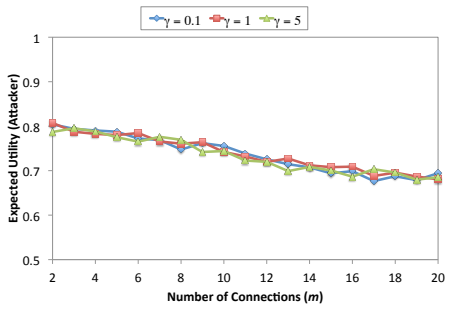

Figure 6: Comparison of attacker utility under different network topologies, with 20 targets. All graphs here use the generalized Preferential Attachment model.

Figure 6 shows the results. As we would expect, increasing the density (m) of the network gives the defender higher utility (lower to the attacker). Surprisingly, however, homogeneity of the degree distribution appears to have little effect.

Adversarial Patrolling Games: Network Design

Our experiments showed that network density plays an important role in determining the efficacy of patrolling. A natural question is: what if the defender can build the network? For example, in a border patrol setting, the defender may choose to build roads or clear certain areas to enable direct moves between important checkpoints. Such investments to improve patrolling efficacy will usually be costly (particularly if one includes maintenance costs), but may be well worth the investment if targets are important enough.

Formally, suppose that the defender will first decide which edges to construct, with a directed edge from i to j costing c_{ij}. (Observe that we can allow for existing edges by setting the corresponding costs $c_{ij} = 0$, and can incorporate constraints by letting $c_{ij} = \infty$.) Once the graph is constructed, the adversarial patrolling game commences just as described above, and, thus, in making the decisions about which edges to construct, the defender must account for the impact of the resulting graph on patrolling efficacy. Fortunately, the decision to build edges can be incorporated directly into the mathematical programming formulations above, with A_{ij} now becoming variables, rather than specified problem parameters.[5]

[5]There is a subtle issue in the network design problem: the result that we rely on to allow us to consider only stationary Markov policies for the defender assumes a zero-sum game, which this no longer is. However, the setting is a zero-sum game *once the edges have been formed*, and that is all that our theorem actually requires.

Baseline Network Design Formulation

One way to solve the network design problem would be to search exhaustively through all the networks: create a network, solve for defender utility using the approach from Section , and iterate. Intuitively, what we do here is short-circuit this approach by doing the entire optimization in one shot.

Let A_{ij} be binary variables with $A_{ij} = 1$ if and only if the defender builds an edge from i to j which he can subsequently use in patrolling decisions. The lone term involving A_{ij} in all our formulations above is linear in A_{ij}, and we therefore need to make no further modifications to the constraints. Since edges have a cost, we must change the objective to reflect the resulting cost-benefit tradeoffs. Therein lies a problem: our formulations above used $\sum_i v_i$ as an objective, while the defender's concern is only about v_0. Consequently, if we simply add a total incurred cost to $\sum_i v_i$ in the objective, the cost term will not be given sufficient weight, and the solution may be suboptimal: in fact, it is fundamentally the tradeoff between value and cost of building edges that we are trying to make here. The true objective of $v_0 + cost$, however, does not work either, since it will fail to correctly compute the values v_i of all states i, which are necessary to correctly obtain v_0: coefficients on all v_i must be strictly positive. We therefore offer the following approximate objective function:

$$\min \quad (1-\alpha)v_0 + \alpha \sum_{i \neq 0} v_i + \sum_{i,j} c_{ij} A_{ij},$$

where $\alpha > 0$ is some small real number, and the last term computes the total cost of building the graph. It can be shown that α can be scaled low enough to ensure that the resulting objective is arbitrarily close to optimal.

The modifications above can be made directly to both the NLP and MILP formulations of the adversarial patrolling problem. However, the modification introduces integer variables, which are especially problematic when with start with a non-linear program. Below we offer an alternative network design formulation in which no integer variables are present.

NLP Network Design Formulation

Above, we used the graph constraint from the basic APG formulations unchanged, and merely introduced A_{ij} as integer variables. Alternatively, we can modify the graph constraint to recover an equivalent formulation of the network design problem that contains no integer variables.

Consider the set of constraints

$$\pi_{ij}(1 - A_{ij}) = 0 \quad \forall\, i, j \in I \tag{7}$$

which are equivalent to those in Equation 1 (when $A_{ij} = 0$, π_{ij} are forced to be 0). While we have just replaced linear constraints with those that are non-linear, the win comes from the fact that we can now relax A_{ij} to be real-valued.

Theorem 1. *Suppose that $A_{ij} \geq 0$ is unrestricted and $c_{ij} > 0$. Further, suppose that we replace the linear graph Constraint 1 in the network design formulation with Constraint 7. Then an optimal solution A_{ij} is binary-valued.*

We note that we can make an analogous modification to the MILP network design formulation, but must subsequently linearize the new set of graph constraints. Nevertheless, we can prove that the resulting linearized version always results in binary-valued A_{ij}.

Experiments: Network Design

In this section, we compare the MILP formulation for network design, which we refer to as MILP (ND), and the nonlinear programming formulation in Section , which we refer to as NLP (ND).

The results in Table 1 offer a compelling case for the MILP network design formulation: attacker values achieved are not very different, but NLP-based approaches are clearly quite suboptimal in terms of design costs, building far more edges than optimal.

method	attacker value	design cost
MILP (ND) (CPLEX)	0.82±0.014	0.45±0.0058
NLP (ND) (IPOPT)	0.78±0.044	7.35±0.29
NLP (ND) (KNITRO)	0.77±0.021	3.14±0.084

Table 1: Comparison of attacker's expected value and defender's network design cost for the NLP (ND) formulation solved by IPOPT and KNITRO, and the MILP (ND) formulation. For all, the number of targets is 20 and per-edge cost is 0.02. For KNITRO, we used 4 restarts; we had not tried more, as even with 4 a significant fraction of instances (between 5 and 10%) simply stall.

Conclusion

We presented a model of discounted adversarial patrolling on exogenous networks, and demonstrated how to formalize it as a highly structured stochastic game. We then adapted a known non-linear programming formulation to our problem in two ways: the first introduced integer variables to compute the optimal attack utilities, following an approach commonly taken in the literature on Stackelberg games, while the second incorporated this decision directly into the NLP. Furthermore, we offered an alternative, albeit approximate, MILP formulation for this problem. Subsequently, we extended the baseline adversarial patrolling model to allow the defender to construct the graph constraining patrolling moves, at some per-edge cost, and offered NLP and MILP formulations to solve this problem. Finally, we presented a model in which the defender can move between an arbitrary pair of targets, but incurs a cost for each move which depends on the source and destination, and offered NLP and MILP formulations to solve several variants of this problem.

Our experiments verify that solutions which we compute are significantly better than those obtained using an alternative formulation applied to a special case of *undiscounted* zero-sum APGs. Overall, both NLP and MILP formulations compute solutions much faster than mixed-integer nonlinear programs, while NLP is much faster than MILP, where applicable. On the other hand, we found that MILP computed much better solutions than NLP in the network design problem.

References

Agmon, N.; Krause, S.; and Kaminka, G. A. 2008. Multi-robot perimeter patrol in adversarial settings. In *IEEE International Conference on Robotics and Automation*, 2339–2345.

Agmon, N.; Urieli, D.; and Stone, P. 2011. Multiagent patrol generalized to complex environmental conditions. In *Twenty-Fifth National Conference on Artificial Intelligence*.

An, B.; Pita, J.; Shieh, E.; Tambe, M.; Kiekintveld, C.; and Marecki, J. 2011. Guards and protect: Next generation applications of security games. In *SIGECOM*, volume 10, 31–34.

Basilico, N., and Gatti, N. 2011. Automated abstraction for patrolling security games. In *Twenty-Fifth National Conference on Artificial Intelligence*, 1096–1099.

Basilico, N.; Rossignoli, D.; Gatti, N.; and Amigoni, F. 2010. A game-theoretic model applied to an active patrolling camera. In *International Conference on Emerging Security Technologies*, 130–135.

Basilico, N.; Gatti, N.; and Amigoni, F. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Eighth International Conference on Autonomous Agents and Multiagent Systems*, 57–64.

Bosansky, B.; Lisy, V.; Jakov, M.; and Pechoucek, M. 2011. Computing time-dependent policies for patrolling games with mobile targets. In *Tenth International Conference on Autonomous Agents and Multiagent Systems*, 989–996.

Conitzer, V., and Sandholm, T. 2006. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, EC '06, 82–90. New York, NY, USA: ACM.

Filar, J., and Vrieze, K. 1997. *Competitive Markov Decision Processes*. Springer-Verlag.

Jain, M.; Tsai, J.; Pita, J.; Kiekintveld, C.; Rathi, S.; Tambe, M.; and Ordóñez, F. 2010. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces* 40:267–290.

Kiekintveld, C.; Jain, M.; Tsai, J.; Pita, J.; Ordóñez, F.; and Tambe, M. 2009. Computing optimal randomized resource allocations for massive security games. In *Seventh International Conference on Autonomous Agents and Multiagent Systems*.

Newman, M. 2010. *Networks: An Introduction*. Oxford University Press.

von Stengel, B., and Zamir, S. 2004. Leadership with commitment to mixed strategies. Technical Report LSE-CDAM-2004-01, CDAM Research Report.