SAND2011-7546C

Infinite Horizon Adversarial Patrolling on Networks

Yevgeniy Vorobeychik*
Sandia National Laboratories
Livermore, CA
yvorobe@sandia.gov

Bo An and Milind Tambe
University of Southern California
Los Angeles, CA
{boa,tambe}@usc.edu

ABSTRACT

Defender-Attacker Stackelberg games are the foundations of tools deployed for computing optimal patrolling strategies in adversarial domains such as the Federal Air Marshals Service and the United States Coast Guard, among others. In Stackelberg game models 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* 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 provide an approximate MILP formulation that uses discrete defender move probabilities. (3) We experimentally demonstrate the efficacy of an NLP-based approach, and systematically study the impact of network topology on the results. (4) 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. (5) We present an alternative model in which we replace graph constraints on defender moves with transition costs, and provide NLP and MILP formulations for several variants of this problem.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed artificial intelligence—*Intelligent agents*

General Terms

Algorithms, Performance, Economics, Security

Keywords

Game theory, Security, Stackelberg Games, Patrolling, MDP

1. 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) [10, 3]. 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 [13, 8, 11].

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 [3], 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 (Section 7 is an exception).

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 3), and offer several alternative formulations that leverage the general non-linear programming approach for computing equilibria in zero-sum stochastic games [9] (Section 4). (2) We provide an approximate MILP formulation that uses discrete defender move probabilities (Section 4). (3) We additionally offer an experimen-

*Sandia National Labs, Livermore, CA. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

tal 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). (4) 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 6). (5) We present a model in which there are no hard constraints on defender moves; instead the defender faces transition costs that depend on the source and destination pairs (Section 7). We provide and evaluate NLP and MILP formulations for this setting as well.

2. 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) [1, 2].

More recent work by Basilico *et al.* [5, 6, 4, 7] studied general-sum patrolling games in which the attacker is assumed to be infinitely patient, but the execution of an attack can take an arbitrary number of time steps (in our setting, in contrast, an attack takes a single step to execute). 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 5.1). 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.

3. 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, 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 [3]. 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.*

3.1 APG as a Stochastic Game

The adversarial patrolling game can be formulated as a *stochastic game* [9]. 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 finite space. Fortunately, in two-player discounted stochastic games there exists an equilibrium in stationary Markovian policies [9]. 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.

4. OPTIMAL PATROLLING POLICIES ON NETWORKS

4.1 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 equi-

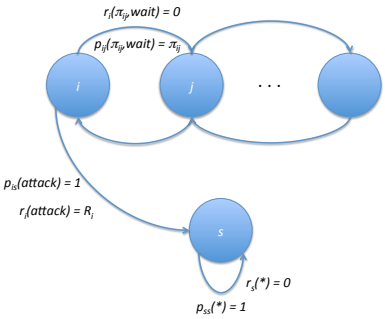

Figure 1: Schematic illustration of APG as a stochastic game, showing example targets-states i and j, as well the absorbing state s. $p_{ij}(\cdot)$ denotes the transition probability, as a function of the probability π_{ij} that the defender moves from i to j and whether or not the attacker chooses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

librium in general zero-sum stochastic games by Filar and Vrieze [9] 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 are expected attacker values 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.

[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.

4.2 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 [11]:

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

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

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

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.

4.3 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. (Arguably, this is actually the most natural way to formulate APGs as a stochastic game. However, as we will see in Section 7, the approach that explicitly represents attacker decisions becomes necessary when the problem has non-zero-sum elements.)

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.

4.4 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:

$$v_j - Z(1 - d_{ijl}) \leq w_{ijl} \leq v_j + Z(1 - d_{ijl}) \quad (4a)$$
$$-Z d_{ijl} \leq w_{ijl} \leq Z d_{ijl}. \quad (4b)$$

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 a MILP:

$$\min_{d,v,w} \sum_i v_i \quad (5a)$$

s.t. :

$$d_{ijl} \in \{0,1\} \qquad \forall\, i,j \in T, l \in \mathcal{L} \quad (5b)$$
$$\sum_l d_{ijl} = 1 \qquad \forall i,j \in T \quad (5c)$$
$$\sum_j \sum_l p_l d_{ijl} = 1 \qquad \forall\, i \in T \quad (5d)$$
$$\sum_l p_l d_{ijl} \leq A_{ij} \qquad \forall\, i,j \in T \quad (5e)$$
$$v_i \geq R_{ij} \qquad \forall\, i,j \in T \quad (5f)$$
$$v_i \geq \delta \sum_j \sum_l p_l w_{ijl} \qquad \forall i \in T \quad (5g)$$
$$-Z d_{ijl} \leq w_{ijl} \leq Z d_{ijl} \qquad \forall\, i,j \in T, l \in \mathcal{L} \quad (5h)$$
$$w_{ijl} \geq v_j - Z(1 - d_{ijl}) \qquad \forall\, i,j \in T, l \in \mathcal{L} \quad (5i)$$
$$w_{ijl} \leq v_j + Z(1 - d_{ijl}) \qquad \forall\, i,j \in T, l \in \mathcal{L}, \quad (5j)$$

where $\mathcal{L} = \{0, \ldots, L\}$. Constraints 5d-5g correspond directly to the constraints 2c-2f in the NLP formulation, only with the discrete probabilities replacing π_{ij}. Constraints 5h-5j 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 4.2. 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.

5. 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.[2] 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 [12] 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 [12], 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.

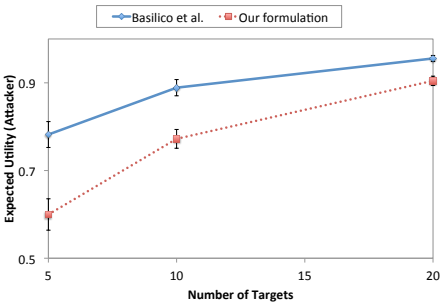

Figure 2: Comparison between our NLP formulation and that developed by Basilico et al. The graph is Erdos-Renyi with $p = 0.1$.

5.1 Comparison to Basilico et al.

Basilico et al. [5] 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 [5] 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

[2]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 $\delta = 1$. For details, see http://aamas.webs.com/appendix_apg.pdf.

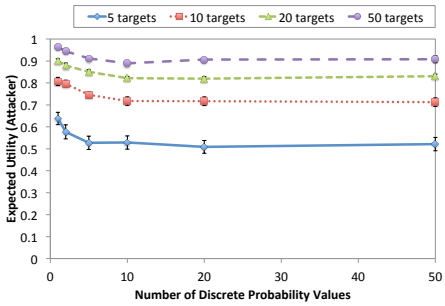

Figure 3: MILP objective value as a function of granularity of discretization in ER(0.1).

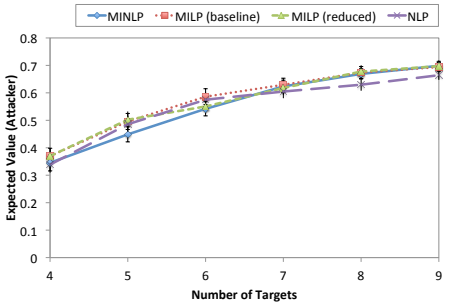

Figure 4: Comparison of average attacker utility achieved using MINLP, two versions of MILP, and NLP formulations, using the Cycle topology.

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.

5.2 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 varying 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.

5.3 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 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.

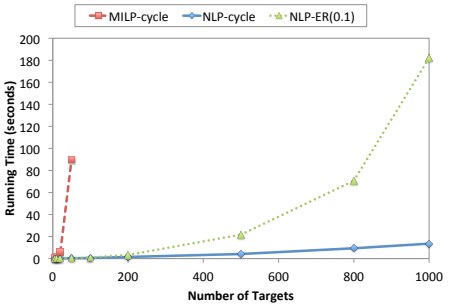

Figure 5: Running time comparison between MILP and NLP on Cycle and ER(0.1) graphs. We omit MINLP which does not scale, and the two MILP formulations yield similar results, so we only present MILP (baseline) here.

5.4 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 [12]. 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 [12], since the parameter m governs the network density, while γ governs the homogeneity of the degree distribution.

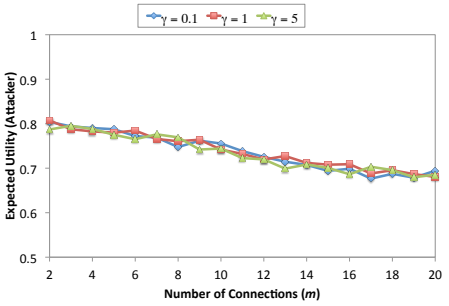

Figure 6: Comparison of attacker utility under different network topologies, with 20 targets. All graphs here use the generalized Preferential Attachment model.

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.

6. 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.[3]

6.1 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 4, and iterate. Intuitively, what we do here is shortcircuit 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.[4]

[3]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.

[4]For proof, see http://aamas.webs.com/appendix_apg.pdf.

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.

6.2 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

$$\pi_{ij}(1 - A_{ij}) = 0 \quad \forall\, i, j \in I \tag{6}$$

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 6.1. *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 6. 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}.[5]

6.3 Experiments: Network Design

In this section, we compare the MILP formulation for network design, which we refer to as MILP (ND), and the non-linear programming formulation in Section 6.2, 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.

In the next set of experiments, we fixed the cost c_{ij} for every edge to be a fixed value c, which we vary between 0 and 0.1. Figure 7 shows the attacker expected utility and algorithm runtime for varying costs per edge c and number of targets. Interestingly, at costs as low as 0.005, the expected utility is already nearly optimal (that is, we do essentially as well as when $c = 0$). For cost between 0.005 and 0.01, we see the peak in computational burden: edge costs are now non-negligible, but good solutions can still be obtained if only the most important edges are built.

[5]See http://aamas.webs.com/appendix_apg.pdf for details.

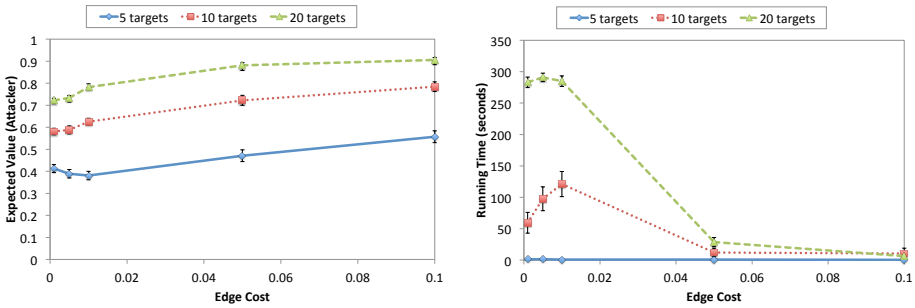

Figure 7: Network design: objective value and runtime for different edge costs and numbers of targets. Results from solving the MILP (ND) formulation (capped at 300 seconds).

7. TRANSITION COSTS

In many realistic settings, it may be unreasonable to expect the graph over which patrolling takes place to be truly fixed. Instead, we may posit that each directed edge (i, j) has some associated cost c_{ij} for the patroller to traverse, and the defender must decide at each point in time the most cost-effective way to patrol among all targets, depending on which target he is patrolling at the moment. (Notice that this setting is again a departure from our zero-sum assumption. In the sequel, we assume that stationary Markovian strategies nevertheless still suffice.) As an example, consider a border patrol setting: only a subset of targets is connected via easily traversable paths (e.g., roads), and in principle moves between targets separated by unfavorable terrain are not impossible, just substantially more costly. Depending on target value, patrol may at times wish to avail themselves of the more costly alternative routes.

Without loss of generality, suppose that the network is completely connected and remove the network constraint (Equation 1) from the optimization.[6] Since the game is no longer zero-sum, the NLP formulations we have used cannot be easily extended to compute a Stackelberg equilibrium in this setting, since we would need to introduce integer variables that explicitly represent attacker decisions. However, we can extend a *modified* version of this problem, where the defender is concerned only in *worst-case expected long-run costs* (just *worst-case costs* below), which would be incurred if the attacker chose to wait indefinitely. This objective may be especially reasonable if the defender does not actually know at which point during his patrol the attacker would arrive, or be ready to attack. We tackle the problem involving worst-case costs first in Section 7.1, and deal with the more complicated case that incorporates attacker decision in calculating the costs in Section 7.2.

7.1 Maximizing Worst-Case Costs

Our NLP formulation can be modified as follows to solve this problem. First, introduce variables C_i to be the sum of expected future costs starting at target i. Since we start the problem at target 0, we modify the objective function to be

$$\min(1-\alpha)v_0 + \alpha \sum_{i \neq 0} v_i + C_0.$$

Just as in the case of network design, we can prove that α can be made sufficiently small here to obtain an arbitrarily good approximation of an optimal solution. In order to compute C_0, we introduce a set of equality constraints that express the recursive relationship between expected cost starting at target i and costs that will accrue depending on where the defense policy will move the defender in the next time period:

$$C_i = \sum_j \pi_{ij} c_{ij} + \delta \sum_j \pi_{ij} C_j \quad \forall i \in I. \tag{7}$$

We can make analogous modifications to the MILP formulations above, and subsequently linearize the corresponding constraints.[7]

7.2 Maximizing Expected Realized Costs

As initially formulated, the problem we really wish to solve is for the defender to maximize his expected costs that are actually realized, accounting for attacker decisions. We noted that the NLP formulations become difficult to extend to solve this problem. We therefore extend an MILP formulation. It turns out to be especially convenient to use the formulation which explicitly represents attacker decisions about which target to attack in each state. This we had captured above using a set of constraints 3. Recall that R_i denotes the value to the attacker of attacking upon observing the defender at target i. We can in fact generate a similar set of constraints to represent attacker decisions whether to attack or wait in state i. Let b_i be a binary variable which is 1 if and only if the attacker waits in state i.

$$0 \leq v_i - R_i \leq b_i Z \qquad \forall\, i \in T \tag{8a}$$

$$0 \leq v_i - \delta \sum_j \pi_{ij} v_j \leq (1 - b_i) Z \qquad \forall\, i \in T. \tag{8b}$$

Constraint 8a computes the value to the attacker of attacking, while Constraint 8b computes the value of waiting, and since b_i is binary, only one of the right-hand-side inequalities will bind.

Once we have identified whether the attacker attacks or waits in each state i, we can compute total discounted cost C_i starting at each state i:

$$C_i = (1 - b_i)\left(\sum_j \pi_{ij} c_{ij}\right) + b_i \left(\sum_j \pi_{ij} c_{ij} + \delta \sum_j \pi_{ij} C_j\right)$$
$$= \sum_j \pi_{ij} c_{ij} + \delta \sum_j b_i \pi_{ij} C_j.$$

The next step is to replace the variables π_{ij} with their discrete counterparts, obtaining

$$C_i = \sum_j \sum_l p_l d_{ijl} c_{ij} + \delta \sum_j \sum_l p_l b_i d_{ijl} C_j.$$

Finally, while we have a non-linear constraint $b_i d_{ijl} C_j$, we can linearize it in a similar fashion as above, since b_i and d_{ijl} are integer variables. Letting $h_{ijl} = b_i d_{ijl} C_j$, we ensure that h_{ijl} satisfies the

[6] This is without loss of generality because for any edge with $A_{ij} = 0$ we can set the cost $c_{ij} = \infty$.

[7] For details, see http://aamas.webs.com/appendix_apg.pdf.

following set of constraints:

$$-Zb_i \leq h_{ijl} \leq Zb_i \; \forall \, i,j,l \tag{9a}$$

$$-Zd_{ijl} \leq h_{ijl} \leq Zd_{ijl} \; \forall \, i,j,l \tag{9b}$$

$$C_j - Z(2 - d_{ijl} - b_i) \leq h_{ijl} \leq C_j + Z(2 - d_{ijl} - b_i) \; \forall \, i,j,l. \tag{9c}$$

There is just one more loose end: the set of constraints that compute b_i above contains the original, non-discretized variables π_{ij}, and one of them has a non-linear term. Thus, we first let $\pi_{ij} = \sum_l p_l d_{ijl}$ throughout, and then recall that we had already linearized the term $\delta \sum_j \sum_l p_l d_{ijl} v_j$ by introducing $w_{ijl} = d_{ijl} v_l$ above, imposing constraints in Equation 4 on w.[8]

7.3 Experiments: Transition Costs

In our experiments pertaining to the formulation that uses transition costs instead of a fixed graph, we generate the cost for each edge (i, j) i.i.d. from a uniform distribution on an interval $[0, c]$, where c is a parameter that we vary (we call it *cost upper bound*). The single exception is that we set the cost of staying at a given target to be 0, which seem natural in most realistic settings.

7.3.1 Worst-Case Costs

Our first finding is that the quality of solutions is not very different between the NLP and MILP formulations. Running time of NLP, however, is several orders of magnitude faster. We therefore use the NLP formulation to study the impact of cost upper bound c on solution quality and runtime. The results of this study are somewhat reminiscent of the observations we made in the network design setting. Runtime peaks at the cost upper bound of 0.01, which already results in substantial costs to the defender for moving between targets, though a nearly optimal solution relative to 0 cost is still achievable, with the defender having to be rather clever about which edges to traverse.

7.3.2 Realized Costs

Our final set of results concerns the model which trades off actual defender transition costs between targets with the desire to limit attacker's utility. Figure 8 shows the attacker utility as well as total defender expenditures for 5 targets.[9] As expected, attacker value increases with defense costs, but rather gradually. Interestingly, the total costs of defense start gradually falling after reaching a peak around $c = 0.75$, presumably as some of the costs become so high so that the corresponding arcs are not worth taking no matter what target the value is.

8. 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 mixed-integer non-linear programs, while NLP is much faster than MILP, where applicable. On the other hand, we found that MILP computes much better solutions than NLP in the network design problem. Additionally, the "transition costs" model in which the defender is concerned with realized (rather than worst-case) costs does not lend itself to an easy NLP adaptation. Instead, we extended the MILP formulation which explicitly represents attacker target choices to compute approximate solutions in this case.

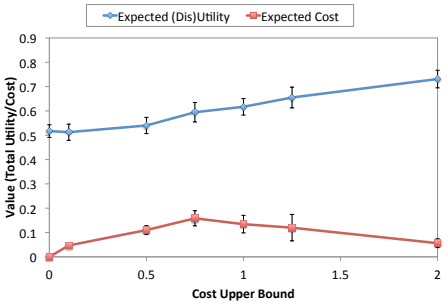

Figure 8: Attacker value v_0 and defender total expenditures, as a function of cost upper bound c in the "transition costs" model. Solved using the MILP with 6 discrete probability levels, for 5 targets. All results are based on at least 60 samples.

9. REFERENCES

[1] Noa Agmon, Sarit Krause, and Gal A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *IEEE International Conference on Robotics and Automation*, pages 2339–2345, 2008.

[2] Noa Agmon, Daniel Urieli, and Peter Stone. Multiagent patrol generalized to complex environmental conditions. In *Twenty-Fifth National Conference on Artificial Intelligence*, 2011.

[3] Bo An, James Pita, Eric Shieh, Milind Tambe, Christopher Kiekintveld, and Janusz Marecki. Guards and protect: Next generation applications of security games. In *SIGECOM*, volume 10, pages 31–34, March 2011.

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

[5] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Eighth International Conference on Autonomous Agents and Multiagent Systems*, pages 57–64, 2009.

[6] Nicola Basilico, Davide Rossignoli, Nicola Gatti, and Francesco Amigoni. A game-theoretic model applied to an active patrolling camera. In *International Conference on Emerging Security Technologies*, pages 130–135, 2010.

[7] Branislav Bosansky, Viliam Lisy, Michal Jakov, and Michal Pechoucek. Computing time-dependent policies for patrolling games with mobile targets. In *Tenth International Conference on Autonomous Agents and Multiagent Systems*, pages 989–996, 2011.

[8] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, EC '06, pages 82–90, New York, NY, USA, 2006. ACM.

[9] Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, 1997.

[8]The complete MILP formulation is provided in the appendix at http://aamas.webs.com/appendix_apg.pdf.

[9]We used a time limit of 300 seconds for CPLEX to solve these problems. Doubling the time limit does not appreciably change the results.

[10] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordóñez. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, 40:267–290, July 2010.

[11] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *Seventh International Conference on Autonomous Agents and Multiagent Systems*, 2009.

[12] Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[13] Bernhard von Stengel and Shmuel Zamir. Leadership with commitment to mixed strategies. Technical Report LSE-CDAM-2004-01, CDAM Research Report, 2004.

APPENDIX

A. PROOFS

A.1 Proof of Theorem 6.1

Suppose $\pi_{ij} > 0$. The only way for the constraint to equal zero in this case is to force $A_{ij} = 1$. Alternatively, suppose that $\pi_{ij} = 0$. Then the value of A_{ij} is unrestricted. However, since $A_{ij} \geq 0$, any positive value of A_{ij} would carry a cost, and have no benefit to the objective value, since $\pi_{ij} = 0$ and this link is effectively unused. Therefore in an optimal solution, $A_{ij} = 0$.

B. FINITE HORIZON PROBLEMS

Abstractly, suppose we wish to maximize discounted rewards over a finite time horizon τ, with δ as the discount factor, but have a solution that maximizes the discounted rewards over an infinite time horizon. The following proposition bounds the quality of the solution to an infinite-horizon problem in terms of the finite-horizon objective.

PROPOSITION B.1. *Suppose that single-period utility is at most $\bar{U}$. Let π^* be an optimal policy for the infinite-horizon problem. Let $v^*(\tau)$ and $\pi^*(\tau)$ be the optimal objective and policy for the problem in which the horizon is τ. Let $v(\tau, \pi)$ be the value of the finite-horizon objective when a policy π is used.*

$$v(\tau, \pi^*) \geq v^*(\tau) - \frac{\bar{U}\delta^{\tau+1}}{1-\delta}.$$

Results of this kind are well known, but we nevertheless prove this proposition for completeness.

PROOF. Define $u(\pi_t)$ to be a (stochastic) reward realization in period t under policy π.

$$\begin{aligned} v(T, \pi^*) &= E\left[\sum_{t=0}^{T} \delta^t u(\pi_t^*)\right] \\ &= E\left[\sum_{t=0}^{\infty} \delta^t u(\pi_t^*)\right] - E\left[\sum_{t=T+1}^{\infty} \delta^t u(\pi_t^*)\right] \\ &\geq E\left[\sum_{t=0}^{\infty} \delta^t u(\pi_t^*(T))\right] - E\left[\sum_{t=T+1}^{\infty} \delta^t \bar{U}\right] \\ &\geq v^*(T) - \frac{\bar{U}\delta^{T+1}}{1-\delta}. \end{aligned}$$

□

As an example, suppose that our optimality tolerance is ϵ. Letting

$$\frac{\bar{U}\delta^{T+1}}{1-\delta} \leq \epsilon,$$

we can obtain a lower bound for the length of the horizon T given δ and ϵ:

$$T \geq \frac{\log(\epsilon(1-\delta)) - \log(\bar{U}\delta)}{\log \delta}.$$

So, if $\delta = 0.5$ and $\bar{U} = 1$, time horizon only needs to be 8 periods for an infinite-horizon solution to be within 0.01 of the finite horizon optimum.

C. APPROXIMATION QUALITY OF NETWORK DESIGN FORMULATION

First, let us abstract the constraint set of the above optimization problem as some set $\mathcal{C}$. Let

$$u(\pi^*) = \min_{\pi, v, A \in \mathcal{C}} v_0 + \sum_{i,j \in T} A_{ij} c_{ij}$$

be the true minimal (optimal) solution, while

$$\hat{u}(\hat{\pi}) = \min_{\pi, v, A \in \mathcal{C}} (1-\alpha) v_0 + \alpha \sum_{j \neq 0} v_j + \sum_{i,j \in T} A_{ij} c_{ij},$$

where π^* and $\hat{\pi}$ are the optimal policy and a policy that optimizes the approximate objective, respectively.

THEOREM C.1. *Suppose that $0 \leq v_i \leq \bar{V}$ for all targets $i \in I$. Then*

$$u(\hat{\pi}) \leq u(\pi^*) + 2\alpha \bar{V}.$$

PROOF. Define $C = \sum_{i,j \in T} A_{ij} c_{ij}$.

$$\begin{aligned} u(\hat{\pi}) = v_0(\hat{\pi}) &= \hat{u}(\hat{\pi}) + \alpha v_0(\hat{\pi}) - \alpha \sum_{j \neq 0} v_j(\hat{\pi}) + C \\ &\leq (1-\alpha) v_0(\pi^*) + \alpha \sum_{j \neq 0} v_j(\pi^*) + \alpha v_0(\hat{\pi}) \\ &\qquad - \alpha \sum_{j \neq 0} v_j(\hat{\pi}) + C \\ &\leq u(\pi^*) + 2\alpha \bar{U}. \end{aligned}$$

□

In our setting the attacker receives a reward only once, when he actually attacks a target; consequently,

$$v_i \leq \max_j \max\{U_a^c(j), U_a^u(j)\}$$

for all targets $i \in I$. Thus, $\bar{V} = \max_j \max\{U_a^c(j), U_a^u(j)\}$. If we further let $\max\{U_a^c(j), U_a^u(j)\} \leq 1$ for all targets j (this is true in all our experiments below), $\bar{V} = 1$, and our approximation incurs an additive error $\leq 2\alpha$.

More generally, suppose that a reward a player receives in each time period is bounded by $\bar{U}$. Since v_i is the expected sum of discounted rewards,

$$v_i = E\left[\sum_{t=0}^{\infty} \delta^t u_t\right] \leq \bar{U} E\left[\sum_{t=0}^{\infty} \delta^t\right] = \frac{\bar{U}}{1-\delta}.$$

Thus,

$$u(\hat{\pi}) \leq u(\pi^*) + \alpha \frac{\bar{U}}{1-\delta}.$$

Consequently, in order to achieve an error tolerance ϵ, we need

$$\alpha \leq \frac{\epsilon(1-\delta)}{\bar{U}}.$$

D. ALTERNATIVE MILP NETWORK DESIGN FORMULATION

In the MILP formulation the alternative set of graph constraints would take the form

$$\sum_l p_l (1 - A_{ij}) d_{ijl} = 0 \quad \forall\, i, j \in I.$$

The basic problem with this set of constraints is when we make A_{ij} a problem variable, they become non-linear. However, since both variables involved are binary, they are easy to linearize. Let $z_{ijl} = d_{ijl}(1 - A_{ij})$ and replace these constraints with linear analogues:

$$\sum_l p_l z_{ijl} = 0 \quad \forall\, i, j \in I. \tag{10}$$

num targets	original	alt, binary	alt, real
5	0.82±0.092	0.61±0.073	0.68±0.068
10	12.14±6.90	9.57±5.10	24.85±11.66
20	28.73±7.36	54.63±13.37	300 (capped)

Table 2: Runtime comparison between three MILP network design formulations: baseline (the original constraint), and two alternative (a linearized non-linear constraint) formulations, one forcing variables to be binary (alt, binary) and another allowing them to be real valued (alt, real).

Finally, we can express the condition that $z_{ijl} = d_{ijl}(1 - A_{ij})$ as a set of linear constraints:

$$0 \leq z_{ijl} \leq d_{ijl} \qquad \forall\, i,j,l \qquad (11a)$$

$$d_{ijl} - A_{ij} \leq z_{ijl} \leq 1 - A_{ijl} \qquad \forall\, i,j,l \qquad (11b)$$

So far we seemed to only have succeeded in introducing more variables to the problem. The punchline, however, is that we can now relax all variables A_{ij} to be real, rather than binary variables.

THEOREM D.1. *Suppose that $A_{ij} \geq 0$ is unrestricted in the alternative formulation above and $c_{ij} > 0$. An optimal solution A_{ij} is always binary-valued.*

PROOF. First we show that $A_{ij} \in [0,1]$. Since $z_{ijl} \geq 0$, constraint 11b (right-hand-side) implies that $A_{ij} \leq 1$. Since $d_{ijl} \leq 1$, $z_{ijl} \leq 1$, and combined with constraint 11b (left-hand-side) it implies that $A_{ij} \geq 0$.

Since $z_{ijl} \geq 0$, constraint 10 implies that either $p_l = 0$ or $z_{ijl} = 0$. Since for any i,j we can only choose a single l with a positive d_{ijl} (constraint 5c), let $\hat{l}$ be such l with $d_{ij\hat{l}} = 1$.

Case 1: suppose $p_{\hat{l}} > 0$ and, therefore, $z_{ij\hat{l}} = 0$. Then $d_{ij\hat{l}} - A_{ij} \leq 0$, or $A_{ij} \geq d_{ij\hat{l}} = 1$. Thus, if I choose a positive probability of defending a target j when at target i, $A_{ij} = 1$.

Case 2: suppose that $p_{\hat{l}} = 0$, that is, probability of defending j starting at i is 0. This means that if (i,j) is ever built, it is not used. Since $\pi_{ij} > 0$ by our assumption, $A_{ij} > 0$ would have no value, but incur a non-zero cost. Thus, in an optimal solution $A_{ij} = 0$. □

Note that we can do away with the assumption that $c_{ij} > 0$ by simply forcing all $A_{ij} = 1$ when $c_{ij} = 0$.[10]

While this alternative MILP formulation has fewer binary constraints, Table 2 suggests that the additional effort was in vain: in fact, the original MILP network design formulation scales considerably better. Surprisingly, simply adding the binary designation to A_{ij} variables actually speeds the alternative formulation up when the number of targets is large!

E. MILP FORMULATION OF THE PROBLEM WITH TRANSITION COSTS

E.1 Worst-Case Costs

In the MILP formulation for the transition costs setting where the defender is only concerned with worst-case costs, we can linearize the constraints in Equation 7, since probabilities are already discretized. First, let us write the discrete analogues:

$$C_i = \sum_j \sum_l p_l d_{ijl} c_{ij} + \delta \sum_j \sum_l p_l d_{ijl} C_j \quad \forall i \in I.$$

Next, let $h_{ijl} = d_{ijl} C_j$, giving us the constraint

$$C_i = \sum_j \sum_l p_l d_{ijl} c_{ij} + \delta \sum_j \sum_l p_l h_{ijl} \quad \forall i \in I. \qquad (12)$$

Finally, we impose the following constraints on h_{ijl}:

$$C_j - Z(1 - d_{ijl}) \leq h_{ijl} \leq C_j + Z(1 - d_{ijl}) \qquad \forall\, i,j,l \qquad (13a)$$

$$-Z d_{ijl} \leq h_{ijl} \leq Z d_{ijl} \qquad \forall\, i,j,l. \qquad (13b)$$

[10]Note that since our formulation actually linearizes the constraint under the assumption that A_{ij} is binary (otherwise linearization need not work), a formal proof must proceed from the linearized formulation, and we cannot rely on Theorem 6.1.

E.2 Realized Costs

Here we present the complete MILP formulation in the transition costs setting when the defender attempts to minimize both the defender utility and realized costs, accounting for the attacker's decisions (wait or attack). Since it's not a zero-sum game, this formulation is now a substantial deviation from the original NLP formulation for zero-sum games, and, indeed, there is no clear extension of the original NLP to cover this setting.

Let $R_{ij} = [(1 - \pi_{ij})U_a^u(j) + \pi_{ij} U_a^c(j)]$. The full MILP formulation is then

$$\min \quad (1-\alpha)v_0 + \alpha \sum_{i \neq 0} v_i + C_0 \qquad (14a)$$

s.t. :

$$d_{ijl} \in \{0,1\} \qquad \forall\, i,j \in T, l \in \mathcal{L} \quad (14b)$$

$$\sum_l d_{ijl} = 1 \qquad \forall i,j \in T \quad (14c)$$

$$\sum_j \sum_l p_l d_{ijl} = 1 \qquad \forall\, i \in T \quad (14d)$$

$$C_i = \sum_j \sum_l p_l d_{ijl} c_{ij} + \delta \sum_j \sum_l p_l h_{ijl} \qquad (14e)$$

$$0 \leq v_i - R_i \leq b_i Z \qquad \forall\, i \in T \quad (14f)$$

$$0 \leq v_i - \delta \sum_j \sum_l p_l w_{ijl} \leq (1 - b_i) Z \qquad \forall\, i \in T \quad (14g)$$

$$a_{ij} \in \{0,1\} \qquad \forall\, i \in T \quad (14h)$$

$$\sum_j a_{ij} = 1 \qquad \forall\, i \in T \quad (14i)$$

$$0 \leq R_i - R_{ij} \leq (1 - a_{ij}) Z \qquad \forall\, i,j \in T \quad (14j)$$

$$-Z d_{ijl} \leq w_{ijl} \leq Z d_{ijl} \qquad \forall\, i,j \in T, l \in \mathcal{L} \quad (14k)$$

$$w_{ijl} \geq v_j - Z(1 - d_{ijl}) \qquad \forall\, i,j \in T, l \in \mathcal{L} \quad (14l)$$

$$w_{ijl} \leq v_j + Z(1 - d_{ijl}) \qquad \forall\, i,j \in T, l \in \mathcal{L}, \quad (14m)$$

$$-Z b_i \leq h_{ijl} \leq Z b_i \qquad \forall\, i,j,l \quad (14n)$$

$$-Z d_{ijl} \leq h_{ijl} \leq Z d_{ijl} \qquad \forall\, i,j,l \quad (14o)$$

$$h_{ijl} \leq C_j + Z(2 - d_{ijl} - b_i) \qquad \forall\, i,j,l \quad (14p)$$

$$h_{ijl} \geq C_j - Z(2 - d_{ijl} - b_i) \qquad \forall\, i,j,l. \quad (14q)$$

This is a rather complicated formulation, so we now walk through the meaning of all the constraints (the objective has the two familiar elements, minimizing both the attacker value starting at target 0 and total cost starting at target 0. d_{ijl} are the discrete probability variables. Thus, Constraint 14c ensures that exactly one discrete probability level is chosen. Constraint 14d ensures that probabilities sum to 1. Constraint 14e computes the defender costs. The purpose of the variables h_{ijl} to linearize the non-linear term $b_i \pi_{ij} C_j$ in which b_i is the binary variable storing the attacker's wait/attack decision. Constraints 14f and 14g compute the value of state i, with b_i ensuring that only one of these constraint binds. Constraint 14i ensures that exactly one target is attacked, and the utility R_i of attacking in state i is computed in constraint 14j. The purpose of all the remaining constraints is to compute the linearization variables w_{ijl} and h_{ijl}.

F. EXPERIMENTS WITH DISCOUNT FACTOR

Here we study the impact of changing the discount factor δ on the attacker's expected utility and the runtime of the NLP model. Figure 9 shows that once the discount factor is at 0.5 or lower, it does not pay for the attacker to wait, and the utility is therefore insensitive to changing the discount factor in this region (recall that positive utility is attained only upon a successful attack in this setup, so attacking immediately implies that the discount factor plays no role, except to discourage waiting). Considering the upper range of discount factors, we can observe that when $\delta > 0.75$, the attacker can often gain a non-negligible value from waiting, and, on the other hand, the expected utility at $\delta = 0.95$ is still significantly below that for $\delta = 1$, suggesting that qualitative differences exist between the two regimes.

Inspecting the runtime plot (Figure 10) reveals no significant runtime differences as long as the discount factor is below 0.95, but runtime rises sharply when it is higher.

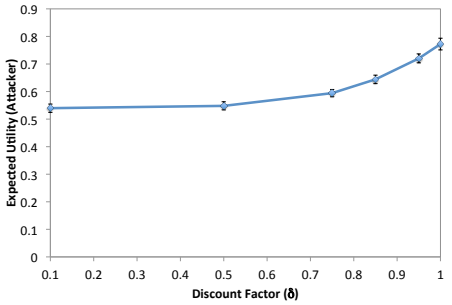

Figure 9: Results for objective value of attacker in the baseline model as we vary the discount factor δ between 0.1 (very impatient attacker) and 1 (no discounting). The NLP model (solved with IPOPT) is used throughout, and the number of targets is fixed at 10.

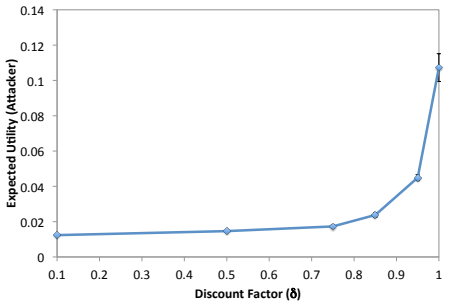

Figure 10: Runtime of the baseline NLP model (solved with IPOPT) as we vary the discount factor δ between 0.1 (very impatient attacker) and 1 (no discounting). The number of targets is fixed at 10.