Security Games for Controlling Contagion

Jason Tsai, Thanh H. Nguyen, Milind Tambe

University of Southern California, Los Angeles, CA 90089
{jasontts, thanhhng, tambe} @usc.edu

Abstract

Many strategic actions carry a ‘contagious’ component beyond the immediate locale of the effort itself. Viral marketing and peacekeeping operations have both been observed to have a spreading effect. In this work, we use counterinsurgency as our illustrative domain. Defined as the effort to block the spread of support for an insurgency, such operations lack the manpower to defend the entire population and must focus on the opinions of a subset of local leaders. As past researchers of security resource allocation have done, we propose using game theory to develop such policies and model the interconnected network of leaders as a graph.

Unlike this past work in security games, actions in these domains possess a probabilistic, non-local impact. To address this new class of security games, we combine recent research in influence blocking maximization with a double oracle approach and create novel heuristic oracles to generate mixed strategies for a real-world leadership network from Afghanistan, synthetic leadership networks, and a real social network. We find that leadership networks that exhibit highly interconnected clusters can be solved equally well by our heuristic methods, but our more sophisticated heuristics outperform simpler ones in less interconnected social networks.

Introduction

Many adversarial domains exhibit ‘contagious’ actions for each player. For example, word-of-mouth advertising / viral marketing has been widely studied by marketers trying to understand why one product or video goes ‘viral’ while others go unnoticed (Trusov, Bucklin, and Pauwels 2009). Recent work has even shown that peacekeeping operations in one nation reduces the probability of conflict arising in nearby areas by 70% (Beardsley 2011).

Counterinsurgency (COIN) is the contest for the support of the local leaders in an armed conflict and can include a variety of operations such as providing security and giving medical supplies (U.S. Dept. of the Army and U.S. Marine Corps 2007). Just as in word-of-mouth advertising and peacekeeping operations, these efforts carry a social effect beyond the action taken that can cause advantageous ripples through the neighboring population (Hung 2010). Moreover, multiple intelligent parties attempt to leverage the same social network to spread their message, necessitating an adversary-aware approach to strategy generation.

We use a game-theoretic approach to the problem and develop algorithms to generate resource allocations strategies for such large-scale, real world networks. We model the interaction as a graph with one player attempting to spread influence while the other player attempts to stop the probabilistic propagation of that influence by spreading their own influence. This ‘blocking’ problem models situations faced by governments/peacekeepers combatting the spread of terrorist radicalism and armed conflict with daily/weekly/monthy visits with local leaders to provide support and discuss grievances (Howard 2011).

This follows work in security games from recent years (Basilico and Gatti 2011; Jain et al. 2011; Letchford and Vorobeychik 2011; Bosanský et al. 2011; Dickerson et al. 2010; Paruchuri et al. 2008; Conitzer and Sandholm 2006). While some works have also modeled interactions on a graph, we extend the approach into a new area where actions carry a ‘contagion’ effect. The problem is a type of influence blocking maximization (*IBM*) problems (Budak, Agrawal, and Abbadi 2011; He et al. 2011), which are a competitive extension of the widely studied influence maximization problem (Chen, Wang, and Wang 2010; Kimura et al. 2010). Past work in influence blocking maximization has looked only at the best-response problems and has not produced algorithms to generate the game-theoretic equilibria necessary for this repeated-interaction domain.

A major contribution of this work is opening up a new area of research that combines recent research in security games and in influence blocking maximization. Drawing from recent work in security games, we propose using a double oracle algorithm where each oracle produces a single player’s best-response to the opponent’s strategy and incrementally creates the payoff matrix being solved. This approach allows us to leverage advances in *IBM* research that has focused entirely on fast best-response calculations.

We begin by proving approximation quality bounds on the double oracle approach when one of the oracles is approximated and combine this with a greedy approximate oracle to produce a more efficient approximate algorithm. To further increase scalability, we introduce two heuristic oracles, LSMI and PAGERANK, that offer much greater efficiency. We conclude with an experimental exploration of a variety of combinations of oracles, testing runtime and quality on random scale-free graphs, a real-world leadership network in Afghanistan, synthetic leadership networks, and a real-world social network. We find that the performance of the PAGERANK oracle suffers minimal loss compared to LSMI in leadership networks that possess clusters of highly interconnected nodes, but performs far worse in sparsely interconnected real-world social networks and scale-free graphs. Finally, an unintuitive blend of oracles offers the best combination of scalability and solution quality.

Related Work

Recent work in game-theoretic security allocation have also dealt with domains that were modeled as graphs (Basilico and Gatti 2011; Jain et al. 2011; Halvorson, Conitzer, and Parr 2009), however their actions were all deterministically defined and did not feature a probabilistic contagion component. This 'spreading' aspect of the problem is very closely related to influence maximization. Influence maximization saw its first treatment in computer science as a discrete maximization problem by Kempe et al. (2003) who proposed a greedy approximation, followed-up by numerous proposed speed-up techniques (Chen, Wang, and Wang 2010; Kimura et al. 2010; Leskovec et al. 2007). Although these are one-player models, we draw from their methods to create more efficient best-response oracles in our work.

Influence blocking maximization problems, which we use to model our domain, have been explored with both independent cascade and linear threshold models of propagation (Budak, Agrawal, and Abbadi 2011; He et al. 2011). Both of these works only explored the defender's best-response problem. Some research exists on competitive influence maximization where *all* players try to maximize their own influence instead of limiting others' (Bharathi, Kempe, and Salek 2007; Kostka, Oswald, and Wattenhofer 2008; Borodin, Filmus, and Oren 2010). Furthermore, these works focus on complexity results instead of equilibrium strategy generation. Hung et al. (2011) and Howard (2010) also address the COIN problem. However, Hung et al. (2011) assume a static adversary and Howard (2010) only solves for local pure strategy equilibria. These are very restrictive assumptions that do not reflect real constraints of the adversary.

Example Domain and Problem Definition

The counterinsurgency domain we focus on includes one party that attempts to subvert the population to their cause and another party that attempts to thwart the first party's efforts (Hung, Kolitz, and Ozdaglar 2011; Howard 2011; Hung 2010). We assume that each side can carry out operations such as provide security or give medical supplies to sway the local leadership's opinion. Furthermore, local leaders will impact other leaders' opinions of the two parties. Specifically, one leader will convert other leaders to side with their affiliated party with some predetermined probability, giving each party's actions a 'spreading' effect. Since resources for COIN operations are very limited relative to the size of the task, each party is faced with a resource allocation task. Hung (2010) models the leadership network of a single district in Afghanistan (based on real data) with 73 nodes and notes that recent organizational assignments show that a single battalion operates in 4-7 districts and divides into 3-4 platoons per 1-2 districts. This translates into 5-30 teams responsible for a network with 300-500 nodes.

We model the counterinsurgency domain as a two-player influence blocking maximization problem, which allows us to draw from the extensive influence maximization literature. An *IBM* takes place on an undirected graph $G = (V, E)$. One player, the attacker, will attempt to maximize the number of nodes supporting his cause on the graph while the second player, the defender, will attempt to minimize the attacker's influence. Vertices represent local leaders that each player can attempt to sway to their cause, while edges represent the influence of one local leader on another. Specifically, each edge, $e = (n, m)$, has an associated probability, p_e , which dictates the chance that leader n will influence leader m to side with n's chosen player. Since the graph is undirected, this is also the probability that m influences n to side with m's chosen player. Only uninfluenced nodes can be influenced.

The two players each choose a subset of nodes as their actions ($S_a, S_d \subseteq V$), which we will also call 'sources', where the allowable size of the subset is given for each player ($|S_a| = r_a$, $|S_d| = r_d$). Each node in $S_a \cap S_d$ has a 50% chance of being influenced by each player, while all other nodes in S_a support the attacker and all other nodes in S_d support the defender. The influence then propagates via a synchronized independent cascade, where at time step t_0 only the initial nodes have been influenced and at t_1 each edge incident to nodes in $S_a \cup S_d$ is 'activated' with probability p_e. Uninfluenced nodes incident to activated edges become supporters of the influencing node's player. If a single uninfluenced node is incident to activated edges from both player's nodes, the node has a 50% chance of being influenced by each player.

For a given pair of actions, the attacker's payoff is equal to the *expected* number of nodes influenced to the attacker's side and the defender's payoff is the opposite of the attacker's payoff. We denote the function to calculate the expected number of attacker-influenced nodes as $\sigma(S_a, S_d)$. Each player chooses a mixed strategy, ρ_a for the attacker and ρ_d for the defender, over their pure strategies (subsets of nodes of size r_a or r_d) to maximize their expected payoff. This mixed strategy is a policy by which COIN teams can randomize their deployment each day/week/month, depending on the frequency of missions. The focus of the rest of this work will be to develop optimal, approximate, and heuristic oracles that can be used in double oracle algorithms to generate strategies for real-world social networks.

Double Oracle Approach

The most commonly used approach for a zero-sum game is a naïve Maximin strategy. This involves precalculating the payoffs for every pair of player actions to determine the entire payoff matrix after which a Maximin algorithm can solve for a Nash equilibrium. Since this is a zero-sum game, a Maximin solution produces policies that are optimal under both a simultaneous-move as well as the leader-follower Stackelberg framework that has been used in much of game-theoretic resource allocation in the recent past (Yin et al. 2010). However, a naïve maximin method admits two faults.

First, the payoff for a pair of player actions, (S_a, S_d), is the value of $\sigma(S_a, S_d)$, which is the expectation of the propagation process outlined previously. As shown by Chen et al. (2010), calculating the analogous expectation in a basic influence maximization game exactly is $\#P$-Hard. Since influence maximization is a special case of influence blocking maximization, it is trivial to show that calculating $\sigma(\cdot)$ exactly is also $\#P$-Hard. The standard method for estimating these expectations is a Monte Carlo approach that was adapted for the *IBM* problem by Budak et al. (2011) and which we also adopt here. It involves simulating the propagation process thousands of times to reach an accurate estimate of the expected outcome. Although it runs in time polynomial in the size of the graph and is able to achieve arbitrarily accurate estimations, the thousands of simulation trials required for accurate results causes this method to be extremely slow in practice.

Second, the Maximin algorithm stores the entire payoff matrix in memory which can be prohibitive for large graphs. For example, with 1000 nodes and 50 resources per player, each player has $\binom{1000}{50}$ actions. To overcome similar memory problems, double oracle algorithms have been proposed in the past (Jain et al. 2011; Halvorson, Conitzer, and Parr 2009) and form the basis for our work.

Double oracle algorithms for zero-sum games use a Maximin linear program at the core, but the payoff matrix is grown incrementally by two oracles. This process is shown in Algorithm 1. **D** is the set of defender actions generated so far, and **A** is the set of attacker actions generated so far. MaximinLP(**D**,**A**) solves for the equilibrium of the game that only has the pure strategies in **D** and **A** and returns ρ_d and ρ_a, which are the equilibrium defender and attacker mixed strategies over **D** and **A**. DefenderOracle($\cdot$), generates a defender action that is a best response against ρ_a among *all* possible actions. This action is added to the set of available pure strategies for the defender **D**. A similar procedure then occurs for the attacker. Convergence occurs when neither best-response oracle generates a pure strategy that is superior to the given player's current mixed strategy against the fixed opponent mixed strategy. The number of attacker and defender actions in the payoff matrix varies with convergence speed, but is generally much smaller than the full matrix. It has been shown that with two optimal best-response oracles, the double oracle algorithm converges to the Maximin equilibrium (McMahan, Gordon, and Blum 2003).

Now we prove an approximate double oracle setup that admits a quality guarantee. We denote the defender and attacker's mixed strategies at convergence as ρ_d and ρ_a. The defender's expected utility given a pair of mixed strategies is $u_d(\rho_d, \rho_a)$. Assume that the *defender*'s oracle, D_{AR}, is an α-approximation of the optimal best-response oracle, D_{BR}, so that $D_{AR}(\rho_a) \geq \alpha \cdot D_{BR}(\rho_a)$. The following theorem is a generalization of a similar result in Halvorson et al. 2009.

Algorithm 1 DOUBLE ORACLE ALGORITHM

```
Initialize **D** with random defender allocations.
Initialize **A** with random attacker allocations.
**repeat**
    $(\rho_d, \rho_a)$ = MaximinLP(**D**,**A**)
    **D** = **D** $\cup$ {DefenderOracle($\rho_a$)}
    **A** = **A** $\cup$ {AttackerOracle($\rho_d$)}
**until** convergence
**return** $(\rho_d, \rho_a)$
```

Theorem 1. *Let (ρ_d, ρ_a) be the output of the double oracle algorithm using an approximate defender oracle and let (ρ_d^*, ρ_a^*) be the optimal mixed strategies. Then: $u_d(\rho_d, \rho_a) \geq \alpha \cdot u_d(\rho_d^*, \rho_a^*)$.*

Proof. Since we know D_{AR} is an α-approximation, $u_d(\rho_d, \rho_a) \geq u_d(D_{AR}(\rho_a), \rho_a) \geq \alpha \cdot u_d(D_{BR}(\rho_a), \rho_a)$. Since (ρ_d^*, ρ_a^*) is a maximin solution, we know that $\forall \rho_d', \rho_a' : u_d(\rho_d^*, \rho_a') \geq u_d(\rho_d^*, \rho_a^*) \geq u_d(\rho_d', \rho_a^*)$. Thus: $u_d(D_{BR}(\rho_a), \rho_a) \geq u_d(\rho_d^*, \rho_a) \geq u_d(\rho_d^*, \rho_a^*)$, implying $u_d(\rho_d, \rho_a) \geq \alpha \cdot u_d(\rho_d^*, \rho_a^*)$. □

Oracles

A major advantage of double oracle algorithms is the ability to divide the problem into best-response components. This allows for easily creating variations of algorithms to meet runtime and quality needs by combining different oracles together. Here, we present four oracles that we can combine to create a suite of algorithms.

EXACT Oracle

The first oracle is an optimal best-response oracle. Our oracle, which we call EXACT , determines the best-response by iterating through the entire action set for a given player. For each action, the expected payoff against the opponent's strategy is calculated, which requires n calculations of $\sigma(\cdot)$, where n is the size of the support for the opponent's mixed strategy. In this oracle, $\sigma(\cdot)$ is evaluated via the Monte Carlo estimation method.

This oracle can be used for both the defender and the attacker to create an incremental, optimal algorithm that can potentially be superior to Maximin because of the incremental approach. However, the oracle will perform redundant calculations that can cause it to run slower than Maximin when the equilibrium strategy's support size is very large.

APPROX Oracle

Here we describe approximate oracles that draw from research in influence maximization, competitive influence maximization, and influence blocking maximization. Budak et al. (2011) showed that the best-response problem for the blocker is submodular when both players share the same probability of influencing across a given edge. Thus, a greedy hill-climbing approach provides the highest marginal gain in each round provides a $(1 - \frac{1}{e} - \epsilon)$-approximation[1].

[1] The ϵ-error of the Monte Carlo estimation exists in the Maximin approach as well and can be made arbitrarily small with suf-

This is outlined in Algorithm 2, where MCEst($\cdot$) is the Monte Carlo estimation of $\sigma(\cdot)$, ρ_a is the current attacker mixed strategy, and Action()/Prob() retrieve a pure strategy, S_a, and its associated probability. The Lazy-Forward speedup to the greedy algorithm introduced by Leskovec et al. (2007) to tackle influence maximization problems is also implemented, but we do not show it in Algorithm 2 for clarity.

For the attacker problem, we note that given a fixed blocker strategy, the best-response problem of the maximizer in an *IBM* is exactly the best-response problem of the last player in a competitive influence maximization from Bharathi et al. (2007), which they showed to be submodular. Thus, the attacker's best-response problem can also be approximated with a greedy algorithm with the same guarantees. These oracles are referred to as APPROX .

By combining an APPROX oracle for the defender and an EXACT oracle for the attacker, we can create an algorithm that generates a strategy for the defender more efficiently than an optimal one and guarantees a reward within $(1 - \frac{1}{e})$ of the optimal strategy's reward by Theorem 1. An algorithm with two APPROX oracles no longer admits quality guarantees, but the iteration process still maintains the best-response reasoning crucial to adversarial domains.

Algorithm 2 APPROX -DefBR(ρ_a)

```
$S_d = \emptyset$
while $|S_d| < r_d$ do
    for $v \in (V - S_d)$ do
        U($n$) = - $\sum_{i=1}^{\rho_a.\text{Size()}} \rho_a$.Prob($i$) $\cdot$
            MCEst($\rho_a$.Action($i$),$S_d \cup \{v\}$)
    end for
    $v^* = \arg\max_{v \in V}$ U($n$)
    $S_d = S_d \cup \{v^*\}$
end while
```

LSMI Oracle

We introduce our main heuristic oracle, LSMI, which is also the name of the heuristic it is based on: Local Shortest-paths for Multiple Influencers (LSMI($\cdot$)). This oracle uses APPROX oracle's Algorithm 2. However, LSMI($\cdot$) is used to replace the MCEst($\cdot$) function and provides a fast, heuristic estimation of the marginal gain from adding a node to the best response. The heuristic is based on two assumptions: very low probability paths between two nodes are unlikely to have an impact and the highest probability path between two nodes estimates the relative strength of the influence. The probability associated with a path is defined as $p = \prod_e p_e$ over all edges e on the path. We then combine these heuristic influences from two players in a novel, efficient way.

The two heuristic assumptions have been applied successfully for one-player influence maximization in various forms, one of the most recent being Chen et al. (2010). When calculating the influence of a node, they only consider nodes reachable via a path with an associated probability of at least some θ. Also, they assume that each source will only affect nodes via the highest probability path. To improve the accuracy of this estimation, they disallow other sources from being on the path since the closer source's influence will supersede the further source's along the same path. We use these ideas as well, but Chen et al. (2010)'s approach to the critical step of combining these influences efficiently relies on there being only one type of influence. In a two-player situation such as ours, there are two probabilities associated with each node, and the winning influencer depends not only on the probability but on the distance to sources as well. This ordering effect is a new issue that necessitates a novel approach to influence estimation.

L-Eval($\cdot$), described in Algorithm 3, is our new algorithm for determining the expected influence of the local neighborhood around a given node. LSMI (n, S_a, S_d) estimates the marginal gain of n by finding the difference between calling L-Eval($\cdot$) with and without n and replaces the MCEst($\cdot$) function in Algorithm 2. For the defender oracle, instead of a call of MCEst($S_a, S_d \cup n$):

$$\begin{aligned} &\text{LSMI}(S_a, S_d, n) = \\ &\quad \text{L-Eval}(V, S_a, S_d \cup \{n\}) \text{ - L-Eval}(V, S_a, S_d), \\ &\qquad\qquad\qquad \text{s.t. } V = \text{GetVerticesWithin}\theta(n). \end{aligned}$$

GetVerticesWithinθ() is a modified Dijkstra's algorithm that measures path-length by hop-distance, tie-breaks with the associated probabilities of the paths, and stores all nodes' shortest hop-distance and associated probability to the given node. It does not add a new node to the search queue if the probability on the path to the node falls below θ.

In L-Eval($\cdot$), V is the set of n's local nodes and S_a/S_d are the attacker/defender source sets. Due to the addition of n, we must recalculate the expected influence of each $v \in V$. First, we determine all the nearby nodes that impact a given v by calling GetVerticesWithin$\theta(v)$. Since only sources exert influence, we intersect this set with the set of all sources and compile them into a priority queue ordered from lowest hop-distance to greatest. p_a and p_d represent the probability that the attacker/defender successfully influences the given node. From the nearest source, we aggregate the conditional probabilities in order. If the next nearest source is an attacker source, then p_a is increased by the probability that the new source succeeds, conditional on the failure of all closer defender and attacker sources. The probability that all closer sources failed is exactly (1 - p_a + p_d). If the next nearest source is a defender source, then a similar update is performed. The algorithm iterates through all impacted nodes and returns the total expected influence.

Although the estimated marginal gain of LSMI can be arbitrarily inaccurate, choosing the best action only requires that the *relative* marginal gain of different nodes be accurate. We show in the Experiments section that LSMI does a very good job of this in practice as evidenced by the high reward achieved by LSMI-based algorithms.

PAGERANK Oracle

PageRank is a popular algorithm to rank webpages (Brin and Page 1998), which we adapt here due to its frequent use in influence maximization as a benchmark heuristic. The underlying idea is to give each node a rating that captures the

ficient simulations(Kempe, Kleinberg, and Tardos 2003).

Algorithm 3 L-Eval(V, S_a, S_d)

```
$InfValue = 0$
for $v \in (V - S_a - S_d)$ do
    N = GetVerticesWithin$\theta(v) \cap (S_a \cup S_d)$
    /* Prioritize sources by *lowest* hop-distance to $v$*/
    S =makePriorityQueue(N)
    $p_a = 0, p_d = 0$
    while S $\neq \emptyset$ do
        $s =$ S.poll()
        if ($s \in S_a$) then
            $p_a = p_a + (1 - p_a - p_d) \cdot$ Prob($s, v$), $p_d = p_d$
        else /* $s$ must be in $S_d$ */
            $p_d = p_d + (1 - p_a - p_d) \cdot$ Prob($s, v$), $p_a = p_a$
        end if
    end while
    $InfValue = InfValue + p_a$
end for
return $InfValue$
```

power it has for spreading influence that is based on its connectivity. For the purposes of describing PageRank, we will refer to directed edges $e_{u,v}$ and $e_{v,u}$ for every undirected edge between u and v. For each edge $e_{u,v}$, set a weight $w_{u,v} = p_e/p_v$ where $p_v = \sum_e p_e$ over all edges incident to v. The rating or 'rank' of a node u, $\tau_u = \sum_v w_{u,v} \cdot \tau_v$ for all non-source nodes v adjacent to u. The exclusion of source nodes is performed because u cannot spread its influence through a source node.

For our oracles, since the defender's goal is to minimize the attacker's influence, the defender oracle will focus on nodes incident to attacker sources $N_a = \{n | n \in V \wedge \exists e_{n,m}, m \in S_a\}$. Specifically, ordering the nodes of N_a by decreasing rank value, the top r_d nodes will be chosen as the best response. In the attacker's oracle phase, the attacker will simply choose the nodes with the highest ranks. Although PageRank is very efficient, we expect its quality to be low, since the attacker oracle fails to account for the presence of a defender and the defender oracle only searches through nodes directly incident to the attacker's source nodes. We will refer to oracles based on this heuristic as PageRank .

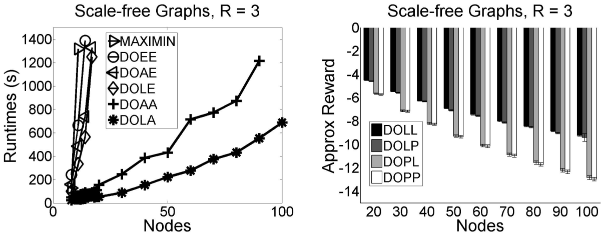
Experiments

In this section, we show experiments on both synthetic and real-world leadership and social networks. We evaluate the algorithms on scalability and solution quality. One advantage of double oracle algorithms is the ease with which the oracles can be changed to produce new variations of existing algorithms. This allows us to simulate various attacker/defender best-response strategies and test our heuristics' performance more thoroughly.

Ideally, we would report the performance of our mixed strategy against an optimal best-response as a worst-case analysis. However, due to scalability issues with the Exact best-response oracle, rewards for larger graphs can only be calculated against an approximate best-response generated by the Approx oracle. Unless otherwise stated, each datapoint is an average over 100 trials and the games created used contagion probability on edges of 0.3, 20,000 Monte Carlo simulations per estimation, and an LSMI $\theta = 0.001$.

Algo Label	Def. Oracle	Att. Oracle	Nodes	R
DOEE	Exact	Exact	15	3
DOAE	Approx	Exact	20	3
DOAA	Approx	Approx	100	3
DOLE	LSMI	Exact	20	3
DOLA	LSMI	Approx	100-200	3
DOLL	LSMI	LSMI	450	20
DOLP	LSMI	PageRank	700	20
DOPE	PageRank	Exact	40	3
DOPA	PageRank	Approx	200-300	3
DOPL	PageRank	LSMI	1000+	20
DOPP	PageRank	PageRank	1000+	20

Table 1: Algorithms evaluated



(a) Runtime, slower algorithms (b) Quality, faster algorithms

Figure 1: Scale-free, less than 100 nodes, 3 resources

In addition to the optimal Maximin algorithm, we also test the set of double oracle algorithms listed in Table 1, where Nodes and R(esources) indicate the approximate problem complexity the algorithm can handle within 20 minutes based on experiments with scale-free graphs.

Random Scale-Free Graphs

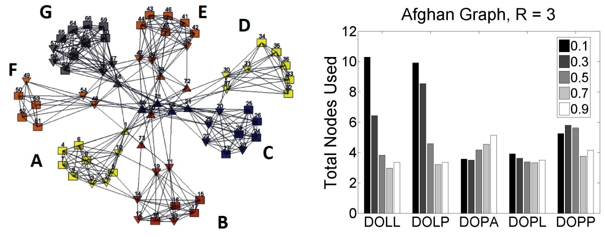
Scale-free graphs have commonly been used as proxies for real-world social networks because the distribution of node degrees in many real world networks have been observed to follow a power law (Clauset, Shalizi, and Newman 2009). We conduct experiments on randomly generated scale-free graphs of various sizes and show runtime and quality results in Figure 1. With only 3 resources, we see most algorithms incapable of scaling past 100 nodes (faster algorithms like DOLL not shown as they hug the x-axis). Experiments with larger graphs with more resources were only possible on algorithms consisting only of LSMI and PageRank oracles. Quality comparison only larger graphs between the four possible such algorithms in Figure 1b reveal that algorithms with LSMI *defender* oracles vastly outperform ones with PageRank defender oracles. Quality is measured against an Approx best-response by an adversary.

Leadership Networks

In Hung (2010), a leadership network was created based on real data of a district in Afghanistan with 7 village areas, each with a few 'village leaders' with connections outside the village, and a cluster of 'district leaders' shown in the middle. We recreate the same network, shown in Figure 2a and run our algorithms on it. Although not shown, quality as measured against an APPROX attacker was very similar for all algorithms. Algorithms exceeding 20min are not shown.

Closer examination of defender strategies reveals a difference in the oracles' approach. Since the PAGERANK defender oracle considers only attacker-adjacent nodes with the highest rank, most of its strategies focus on two high-degree district leaders (neither are maximal degree nodes) and on a regular member of the highest population Village G. In this graph structure, where sets of nodes are fully connected, this strategy works very well because the attacker's best response will often be the highest degree district leader and a node in Village G. This approach is more conservative than LSMI , which directly chooses the attacker's source nodes since the 50% chance of wiping out an attacker source provides slightly higher utility. The attacker oracles all select from the same set of four high-degree nodes. Aside from the highest-degree district leader and Village G nodes, an additional high-degree village leader far from Village G is also used. This result suggests that not only connectivity, but also strategic spacing provided by our algorithms is a key point for the maximizer's target selection.

Experiments varying contagion probability, shown in Figure 2b, show LSMI defender oracle algorithms randomizing over many more nodes at low contagion levels. This is because the attacker's initial set of nodes accounts for most of his expected utility, encouraging randomization over many nodes. PAGERANK ignores this since a given set of nodes is often adjacent to all sets of attacker-chosen nodes, while LSMI matches the increased node use directly.



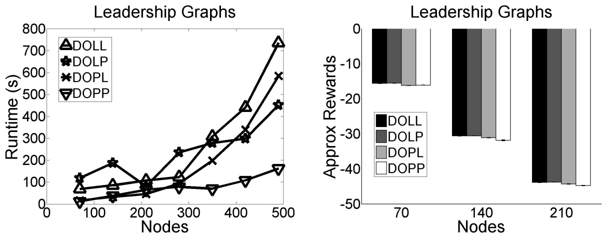
(a) Network from Hung (2010) (b) Nodes in defender strategy

Figure 2: Afghanistan leadership network results

As noted previously, a battalion is responsible for 4-7 districts, so we create synthetic graphs with multiple copies of a village structure (70 nodes each) and link all district leaders together to create multi-district graphs. In our experiments, for every district, each player is given 3 resources. Figure 3 shows runtime and solution quality against an APPROX attacker best-response. Since we create the graphs one district at a time, the graph sizes increase by 70 nodes at a time. The trend in rewards is once again that LSMI defender oracle algorithms very slightly outperform the others. All four algorithms scale to real-world problem sizes.

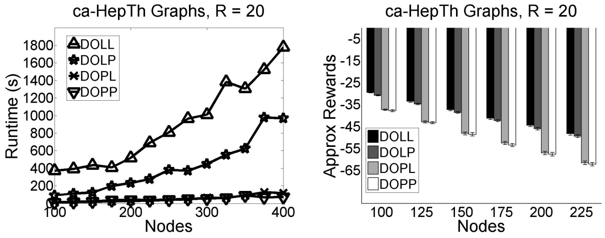
Social Networks

To evaluate our performance on social networks, we use the real-world network commonly used to evaluate influence maximization algorithms: High Energy Physics Theory collaboration network (ca-HepTh). We use this graph as an approximation for a general social network as opposed to the leadership network in the previous section which is hierarchical in structure. For the experiments conducted herein, we extract randomly generated subgraphs of varying sizes each of which is generated so that the degree of included nodes are proportional to their degree in the actual dataset.



(a) Runtime (b) Quality

Figure 3: Synthetic leadership network results



(a) Runtime (b) Reward

Figure 4: ca-HepTh results

The results shown in Figure 4 are very similar to the results from Figure1. Unlike in the leadership graphs, the PAGERANK defender oracle works poorly in social networks, just as in random scale-free graphs. Simply choosing the highest ranking neighbors may have minimal effect on the influence of an attacker source because many neighbors will *not* be interconnected, which was not the case in leadership networks.

Conclusion

With increasingly informative data about interpersonal connections, principled methods can finally be applied to inform strategic interactions in social networks. Our work combines recent research in influence blocking maximization, operations research, and game-theoretic resource allocation to provide the first set of solution techniques for a novel class of security games with contagious actions. Experiments on real-world leadership and social networks reveal that a simple PAGERANK oracle can provide high quality solutions for graphs with clusters of highly interconnected nodes, whereas more sophisticated techniques can be very beneficial in sparsely connected graphs. The methods used herein are a first step into a new area of research in game-theoretic security with applications ranging from product marketing to peacekeeping in warring states.

Acknowledgments

This research was supported by the United States Department of Homeland Security through the National Center for Risk and Economic Analysis of Terrorism Events (CREATE) under award number 2010-ST-061-RE0001. Any opinions, findings, conclusions or recommendations herein are those of the authors and do not reflect views of the United States Department of Homeland Security, or the University of Southern California, or CREATE.

References

[Basilico and Gatti 2011] Basilico, N., and Gatti, N. 2011. Automated abstractions for patrolling security games. In *AAAI*.

[Beardsley 2011] Beardsley, K. 2011. Peacekeeping and the contagion of armed conflict. *The Journal of Politics* 73(4):1051–1064.

[Bharathi, Kempe, and Salek 2007] Bharathi, S.; Kempe, D.; and Salek, M. 2007. Competitive influence maximization in social networks. In *WINE*, 306–311.

[Borodin, Filmus, and Oren 2010] Borodin, A.; Filmus, Y.; and Oren, J. 2010. Threshold models for competitive influence in social networks. In *WINE*, 539–550.

[Bosanský et al. 2011] Bosanský, B.; Lisý, V.; Jakob, M.; and Pechoucek, M. 2011. Computing time-dependent policies for patrolling games with mobile targets. In *AAMAS*, 989–996.

[Brin and Page 1998] Brin, S., and Page, L. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks* 30:107–117.

[Budak, Agrawal, and Abbadi 2011] Budak, C.; Agrawal, D.; and Abbadi, A. E. 2011. Limiting the spread of misinformation in social networks. In *WWW*, 665–674.

[Chen, Wang, and Wang 2010] Chen, W.; Wang, C.; and Wang, Y. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*, 1029–1038.

[Clauset, Shalizi, and Newman 2009] Clauset, A.; Shalizi, C. R.; and Newman, M. E. J. 2009. Power-law distributions in empirical data. *SIAM Rev.* 51:661–703.

[Conitzer and Sandholm 2006] Conitzer, V., and Sandholm, T. 2006. Computing the optimal strategy to commit to. In *ACM EC-06*, 82–90.

[Dickerson et al. 2010] Dickerson, J.; Simari, G.; Subrahmanian, V.; and Kraus, S. 2010. A Graph-Theoretic Approach to Protect Static and Moving Targets from Adversaries. In *AAMAS*, 299–306.

[Halvorson, Conitzer, and Parr 2009] Halvorson, E.; Conitzer, V.; and Parr, R. 2009. Multi-step multi-sensor hider-seeker games. In *IJCAI*, 159–166.

[He et al. 2011] He, X.; Song, G.; Chen, W.; and Jiang, Q. 2011. Influence blocking maximization in social networks under the competitive linear threshold model technical report. *CoRR* abs/1110.4723.

[Howard 2011] Howard, N. J. 2011. *Finding optimal strategies for influencing social networks in two player games*. Masters thesis, MIT, Sloan School of Management.

[Hung, Kolitz, and Ozdaglar 2011] Hung, B. W. K.; Kolitz, S. E.; and Ozdaglar, A. E. 2011. Optimization-based influencing of village social networks in a counterinsurgency. In *SBP*, 10–17.

[Hung 2010] Hung, B. W. K. 2010. *Optimization-Based Selection of Influential Agents in a Rural Afghan Social Network*. Masters thesis, MIT, Sloan School of Management.

[Jain et al. 2011] Jain, M.; Korzhyk, D.; Vanek, O.; Conitzer, V.; Pechoucek, M.; and Tambe, M. 2011. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, 327–334.

[Kempe, Kleinberg, and Tardos 2003] Kempe, D.; Kleinberg, J. M.; and Tardos, É. 2003. Maximizing the spread of influence through a social network. In *KDD*, 137–146.

[Kimura et al. 2010] Kimura, M.; Saito, K.; Nakano, R.; and Motoda, H. 2010. Extracting influential nodes on a social network for information diffusion. *Data Min. Knowl. Discov.* 20(1):70–97.

[Kostka, Oswald, and Wattenhofer 2008] Kostka, J.; Oswald, Y. A.; and Wattenhofer, R. 2008. Word of mouth: Rumor dissemination in social networks. In *SIROCCO*, 185–196.

[Leskovec et al. 2007] Leskovec, J.; Krause, A.; Guestrin, C.; Faloutsos, C.; VanBriesen, J. M.; and Glance, N. S. 2007. Cost-effective outbreak detection in networks. In *KDD*, 420–429.

[Letchford and Vorobeychik 2011] Letchford, J., and Vorobeychik, Y. 2011. Computing randomized security strategies in networked domains. In *Proceedings of the Workshop on Applied Adversarial Reasoning and Risk Modeling (AARM) at AAAI*.

[McMahan, Gordon, and Blum 2003] McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the presence of cost functions controlled by an adversary. In *ICML*, 536–543.

[Paruchuri et al. 2008] Paruchuri, P.; Pearce, J. P.; Marecki, J.; Tambe, M.; Ordóñez, F.; and Kraus, S. 2008. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *AAMAS*.

[Trusov, Bucklin, and Pauwels 2009] Trusov, M.; Bucklin, R. E.; and Pauwels, K. 2009. Effects of word-of-mouth versus traditional marketing: Findings from an internet social networking site. *Journal of Marketing* 73.

[U.S. Dept. of the Army and U.S. Marine Corps 2007] U.S. Dept. of the Army and U.S. Marine Corps. 2007. *The U.S. Army/Marine Corps Counterinsurgency Field Manual 3-24*. University of Chicago Press.

[Yin et al. 2010] Yin, Z.; Korzhyk, D.; Kiekintveld, C.; Conitzer, V.; ; and Tambe, M. 2010. Stackelberg vs. nash in security games: Interchangeability, equivalence, and uniqueness. In *AAMAS*.