Game-Theoretic Target Selection in Contagion-based Domains

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 scale-free graphs. 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 scale-free graphs.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Security, Performance

Keywords

Game theory, Social contagion, Influence maximization

1. 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 [26].

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 [27]. 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 [15]. 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 [14].

This follows work in security games from recent years [3, 17, 22, 24, 11]. 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 [7, 13], which are a competitive extension of the widely studied influence maximization problem [9, 19]. 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 a real-world leadership network in Afghanistan, synthetic leadership networks, and random scale-free graphs. We find that the performance of the basic 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 scale-free graphs. Finally, an unintuitive blend of the two oracles offers the best combination of scalability and solution quality.

2. RELATED WORK

Recent work in game-theoretic security allocation have also dealt with domains that were modeled as graphs [3, 17, 12], 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 and inoculation problems. Influence maximization, in which a player attempts to optimize a selection of beginning 'seed' nodes from which to spread his influence in a known graph, 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 [9, 19, 21]. Although these are one-player games, we draw inspiration from their techniques to address efficiency issues in our work.

Standard inoculation games feature a defender that attempts to protect nodes in a graph and, usually, a random outbreak of a disease on a node in the graph. These games typically model nature as the adversary, which chooses an initial set of nodes with some predefined probability distribution that the defender is optimizing against [1, 2, 8, 20]. Variations on this include distributed inoculation games where each node acts independently, in which results such as price of anarchy are generally considered [1, 8]. Inoculation games do not typically include an optimizing adversary, amounting to only an attacker *or* defender best-response problem. Influence blocking maximization problems, which we use to model our domain, have been explored with both independent cascade and linear threshold models of propagation [7, 13]. 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' [4, 5]. Furthermore, these works focus on complexity results instead of equilibrium strategy generation. Aside from influence blocking maximization, a number of researchers have also explored mutual maximization models where both players seek to maximize their own influence [5, 4]. Finally, 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 pure strategies. This forced predictability in a repeated-interaction situation is dangerous since a real adversary can directly ambush COIN teams. Additionally, it may be suboptimal since a real adversary has no such limitation.

3. 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 [16, 14, 15]. 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,

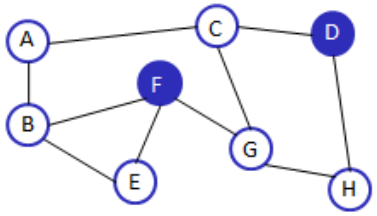


Figure 1: Example action for one player

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 ($|S_a| = r_a$, $|S_d| = r_d$). Figure 1 shows an example of an action for one player as the selection of the two nodes, D and F, filled in. The other player would similarly choose a set of nodes on the same graph from which to begin spreading his influence.

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. This process is detailed in Algorithm 1.

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 ex-

pected 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.

Algorithm 1 INFLUENCE PROP.: S_a, S_d, $G = (N, E)$

```
E* = ∅, E^active = ∅
A ← {s|s ∈ S_a ∧ s ∉ S_d}, D ← {s|s ∉ S_a ∧ s ∈ S_d}
for {s|s ∈ S_a ∩ S_d} do
    // randomly add s to one of the player's sets
    RandomAdd(s, A, D)
end for
N^new = A ∪ D
while N^new ≠ ∅ do
    for {(u, v)|u ∈ N^new,(u, v) ∉ E*} do
        // activate the edge based on its probability
        E^active.add(RandomActivate((u, v)))
        E*.add((u, v))
    end for
    N^new = ∅
    for {s|s ∉ A ∪ D, ∃(u, s) ∈ E^active} do
        N^new.add(s)
        // Add s to appropriate set
        AddToSet(s, A, D)
    end for
end while
```

4. 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. However, this naïve approach 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 [17, 12] 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 2. **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 depending on the speed of convergence, 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 [23].

Algorithm 2 DOUBLE ORACLE ALGORITHM

```
Initialize D with random defender allocations.
Initialize A with random attacker allocations.
repeat
    (ρ_d, ρ_a) = MaximinLP(D,A)
    D = D ∪ {DefenderOracle(ρ_a)}
    A = A ∪ {AttackerOracle(ρ_d)}
until convergence
return (ρ_d, ρ_a)
```

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. Also, we denote the defender's expected utility given a pair of mixed strategies as $u_d(\rho_d, \rho_a)$. 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.

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 $u_d(\rho_d, \rho_a) \geq \alpha \cdot u_d(\rho_d^*, \rho_a^*)$. □

5. 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.

5.1 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[1].

[1]The ϵ-error of the Monte Carlo estimation exists in the Maximin approach as well, but can be made arbitrarily small with sufficient simulations[18].

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.

5.2 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 $(1 - \frac{1}{e})$-approximation. This is outlined in Algorithm 3, 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 speed-up 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 3 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 3 APPROX -DefBR(ρ_a)

```
S_d = ∅
while |S_d| < r_d do
    for n ∈ (N − S_d) do
        U(n) = Σ_{i=1}^{ρ_a.Size()} ρ_a.Prob(i) ·
            MCEst(ρ_a.Action(i), S_d ∪ {n})
    end for
    n* = arg max_{n∈N} U(n)
    BR = BR ∪ {n*}
end while
```

5.3 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 3. 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 4, 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 3. For the defender oracle, instead of a call of MCEst$(S_a, S_d \cup n)$:

$$LSMI(S_a, S_d, n) =$$
$$\text{L-Eval}(V, S_a, S_d \cup \{n\}) \text{ - L-Eval}(V, S_a, S_d),$$
$$\text{s.t. } V = \text{GetVerticesWithin}\theta(n).$$

GetVerticesWithin$\theta()$ 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 $p_a + p_d$. p_d remains unchanged. 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.

Algorithm 4 L-Eval(V, S_a, S_d)

$InfValue = 0$
for $v \in (V - S_a - S_d)$ **do**
  $\mathbf{N}$ = GetVerticesWithin$\theta(v) \cap (S_a \cup S_d)$
  /* Prioritize sources by *lowest* hop-distance to v*/
  $\mathbf{S}$ =makePriorityQueue($\mathbf{N}$)
  $p_a = 0, p_d = 0$
  while S $\neq \emptyset$ **do**
    $s = \mathbf{S}$.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 $InfValue$

5.4 PAGERANK Oracle

PageRank is a popular algorithm to rank webpages [6], 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 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 .

6. 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

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.

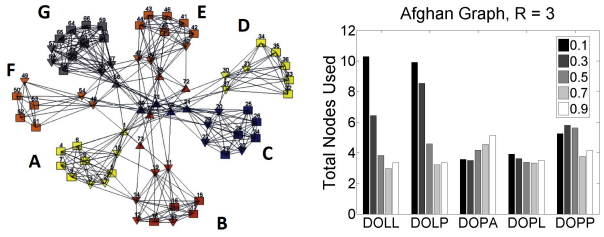
6.1 Leadership Networks

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 occurs 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 responds by matching the increase node use directly.

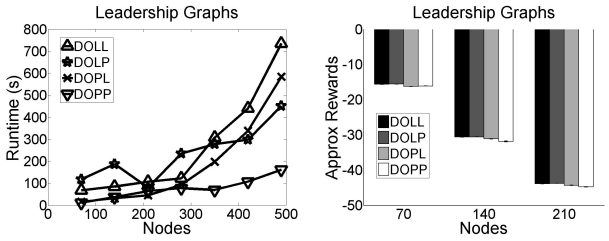
As noted previously, a battalion is responsible for 4-7 districts, so we create synthetic graphs with multiple copies of a village structure (70 nodes each) and link all district leaders together to create multi-district graphs. In our experiments, for every district, each player is given 3 resources. Figure 3 shows runtime and solution quality against an APPROX attacker best-response. Since we cre-



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

Figure 2: Afghanistan leadership network results

ate 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.



(a) Runtime (b) Quality

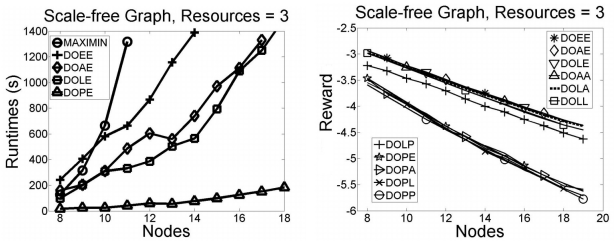
Figure 3: Synthetic leadership network results

6.2 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 [10]. We conduct experiments on randomly generated scale-free graphs of various sizes to illustrate both the runtime scalability and quality of each algorithm in graphs resembling social networks as opposed to leadership networks.

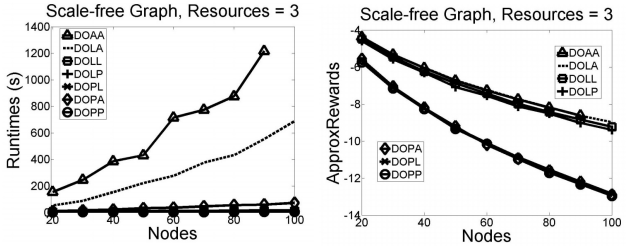
Figure 4 shows the results for small scale-free graphs of 8-20 nodes with 3 resources for each player. The runtime graph, Figure 4a shows only the algorithms that exceed 20 minutes for clarity. The remaining heuristic algorithms' results all hug the x-axis because they take minimal time for these graphs. As would be expected, Maximin scales the most poorly and is only able to handle graphs of up to 11-12 nodes. The approximate algorithm, DOAE improves upon DOEE and can handle up to 16-17 nodes, but swapping out the APPROX oracle for the very fast LSMI oracle does not improve runtime scalability very noticeably. This is because although the LSMI oracle is orders of magnitude faster than the APPROX oracle, the EXACT attacker oracle's runtime eclipses both of them, making the improvement irrelevant.

In Figure 4b, we show the reward obtained by the defender when using the strategies generated against an EXACT attacker best-response as described earlier. The key point is that the majority of rewards are indistinguishable from the optimal algorithms. The DOLL algorithm begins to diverge slightly when the graph nears 100 nodes, but the major exceptions are the algorithms featuring PAGERANK defender oracles. Interestingly, DOLP, which uses LSMI for the



(a) Runtime (b) Quality

Figure 4: Scale-free, 8-20 nodes, 3 resources



(a) Runtime (b) Quality

Figure 5: Scale-free, 20-100 nodes, 3 resources

defender and PAGERANK for the attacker still generates high rewards.

Figure 5 shows runtime and quality for larger scale-free graphs of 20-100 nodes with 3 resources for each player. As can be seen, the algorithms featuring the APPROX oracle (DOAA, DOLA) begin to exceed our 20-minute cutoff near 100 nodes while the remaining heuristic algorithms continue to hug the x-axis because even these games are completed in minimal time. As discussed previously, due to the inefficiency of the EXACT oracle, we use an APPROX best-response to calculate a more conservative reward value. Figure 5b again shows algorithms with PAGERANK defender oracles performing noticeably more poorly than the other algorithms. DOLP is again very close to the top performers. Note that while this may be due to the APPROX best-response being used instead of an EXACT best-response, it is very unlikely than an attacker could perform any better given the hardness of the best-response problem.

Finally, we show very large graph scalability with 100-500 nodes and 20 resources per player in Figure 6. These games can only be handled by algorithms using two heuristic oracles, so we try all combinations of LSMI and PAGERANK oracles. When two LSMI oracles are used, the algorithm begins to exceed 20 minutes around 450-475 nodes. However, when even one of the oracles is replaced with a PAGERANK oracle, the algorithm scales much better. As we noted earlier, DOLP performs very close to DOLL's quality and here we see that it scales much better, suggesting that this combination of oracles provides the best blend of runtime scalability and quality.

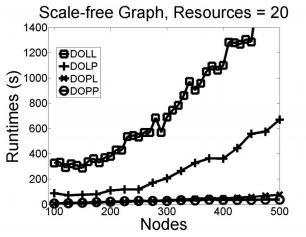


Figure 6: Scale-free, 100-500 nodes, 20 resources

7. 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 scale-free graphs 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 wide-ranging applications.

8. FUTURE DIRECTIONS

This type of maximize/mitigate scenario can be used to model a number of other domains that we hope to apply them to. For example, anti-vaccination groups have become a serious issue for health organizations to address [25]. By modeling the interaction as an adversarial information diffusion problem, the techniques here can help health organizations mitigate the impact of anti-vaccination propaganda. In political campaigns, candidates often attempt to disseminate negative information about their opponents to sway votes against them. Again, we can model this scenario with one party attempting to maximize the spread of this information while another party attempts to block the spread by disseminating its own news (e.g., their own negative propaganda, positive spin on the negative news, bigger news). These new domains will introduce novel challenges as we improve the fidelity of our models to fit these problems.

9. 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.

10. REFERENCES

[1] J. Aspnes, K. Chang, and A. Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '05, pages 43–52, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[2] J. Aspnes, N. Rustagi, and J. Saia. Worm versus alert: who wins in a battle for control of a large-scale network? In *Proceedings of the 11th international conference on Principles of distributed systems*, OPODIS'07, pages 443–456, Berlin, Heidelberg, 2007. Springer-Verlag.

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

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

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

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

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

[8] P.-A. Chen, M. David, and D. Kempe. Better vaccination strategies for better people. In *Proceedings of the 11th ACM conference on Electronic commerce*, EC '10, pages 179–188, New York, NY, USA, 2010. ACM.

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

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

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

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

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

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

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

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

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

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

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

[20] V. S. A. Kumar, R. Rajaraman, Z. Sun, and R. Sundaram. Existence theorems and approximation algorithms for generalized network security games. In *Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems*, ICDCS '10, pages 348–357,

Washington, DC, USA, 2010. IEEE Computer Society.

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

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

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

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

[25] P. Shetty. Experts concerned about vaccination backlash. *The Lancet*, 375(9719):970–971, 2010.

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

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