Analysis of Heuristic Techniques for Controlling Contagion

Jason Tsai, Nicholas Weller, Milind Tambe

University of Southern California, Los Angeles, CA 90089
{jasontts@usc.edu, nweller@dornsife.usc.edu, 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, recent research has used novel heuristic oracles in a double oracle formulation to generate mixed strategies. However, these heuristic oracles were evaluated only on runtime and quality scaling with the graph size. Given the complexity of the problem, numerous other problem features and metrics must be considered to better inform practical application of such techniques. Thus, this work provides a thorough experimental analysis including variations of the contagion probability average and standard deviation. We extend the previous analysis to also examine the size of the action set constructed in the algorithms and the final mixed strategies themselves. Our results indicate that game instances featuring smaller graphs and low contagion probabilities converge slowly while games with larger graphs and medium contagion probabilities converge most quickly.

Introduction

Numerous competitive situations feature actions that produce a contagious effect. Word-of-mouth advertising / viral marketing has gained significant research interest as companies attempt to understand what makes some products or videos go viral (Trusov, Bucklin, and Pauwels 2009). Peacekeeping operations have also been shown to have a positive effect on neighboring areas (Beardsley 2011).

Counterinsurgency (COIN) is the military effort to win the support of a local population by activities such as providing security, medical supplies, and building infrastructure (U.S. Dept. of the Army and U.S. Marine Corps 2007). These efforts often reach the ears of neighboring populations that may then form a positive opinion about the organization/country as well. Given that counterinsurgency is, by definition, a competitive effort, deciding which areas to visit requires a principled, adversary-aware approach.

Our recent work proposes the use of game theory to develop these resource allocation policies (Tsai, Nguyen, and Tambe 2012), taking after the deployed applications now in use by numerous security agencies (Shieh et al. 2012; Pita et al. 2011). Specifically, we model the problem as a game with two players that takes place on a graph in which nodes represent local leaders and links between them represent the influence they have upon each other. Each player's task is then to each select a subset of the local leaders to visit and win over. These initial leaders will then probabilistically spread their opinion to neighboring leaders with the probabilities dictated by the edge weights. The goal of the insurgents is to maximize the support of the local populace while the counterinsurgents seek to mitigate the insurgents' support. We then solved the game for the Nash equilibrium strategies for each player. However, given the limitations of known optimal and approximate solution methods, we developed heuristic techniques that provide scalable solutions that showed minimal quality loss in our experiments.

Given the lack of theoretical guarantees for heuristic techniques, the purpose of this work is to thoroughly evaluate the performance of the proposed heuristic techniques and identify key trends that would inform their use in practical application. Our prior work evaluated runtime and solution quality as the graph size was increased. In this work, instead of only varying the graph size, we also evaluate the impact of changing the contagion probability on edges. In particular, our prior evaluation set all edge probabilities to 0.3, while here we draw the edge probabilities from a normal distribution and vary the average and standard deviation of the distribution. Furthermore, for all of the above variations, we examine not only the runtime and solution quality, but also the support set size of the strategies generated and the number of actions that are generated by the iterative double oracle approach.

Our investigation reveals that when the graph size increases, all of the algorithms converge in fewer iterations because the defender oracles are unable to find strong blocking actions. Furthermore, game instances with low contagion probabilities prove difficult because the attacker can continuously find new actions that avoid the entire set of current defender actions until a sufficient number of actions had been added to both players' action sets, thereby requiring many iterations to converge. Finally, our experiments varying the contagion probability distribution's standard deviation revealed little impact of this parameter on the performance of the algorithms tested.

Related Work

Related work generally falls into one of two categories: (1) game-theoretic resource allocation that does not feature networked domains with contagion and (2) influence maximization which has only focused on best-response generation and usually features only one 'player'. In game-theoretic resource allocation, the most relevant work provides algorithms for strategy generation in games that take place on a network (Basilico and Gatti 2011; Jain et al. 2011; Halvorson, Conitzer, and Parr 2009). However, these domains do not feature a diffusion effect which results in immense complexity as shown by work in influence maximization.

Specifically, influence maximization studies the problem of what subset of nodes to select in a graph to maximize the spread of influence (Chen, Wang, and Wang 2010; Leskovec et al. 2007; Kempe, Kleinberg, and Tardos 2003). The spreading occurs via known propagation dynamics, translating into a one player version of the problem we are interested in in this work. Influence blocking maximization is a very recent area that has looked at the two-player maximize/mitigate game we consider, but has thus far only focused on best-response generation instead of equilibrium strategies (Budak, Agrawal, and Abbadi 2011; He et al. 2011). Some research exists on competitive influence maximization where *all* players try to maximize their own influence instead of limiting others' (Goyal and Kearns 2012; Borodin, Filmus, and Oren 2010; Kostka, Oswald, and Wattenhofer 2008; Bharathi, Kempe, and Salek 2007). 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) 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

While many domains feature a competitive scenario on a network with diffusion effects, we will use counterinsurgency as an example domain to illustrate our methods. Specifically, the counterinsurgency domain includes one player that is attempting to win the support of the population to their cause while the second player attempts to thwart the first player's efforts (Hung, Kolitz, and Ozdaglar 2011; Howard 2011; Hung 2010). Each player is assumed to possess the ability to win local leaders over to their side via activities such as providing security, medical supply, or helping to build infrastructure. Furthermore, each leader has a known probability of influencing other leaders to support their affiliated player. Prior work has noted the immense size of the problem faced by military leaders engaging in counterinsurgency. Specifically, Hung (2010) notes that in Afghanistan, a single battalion with 5-30 teams is responsible for an area consisting of approximately 300-500 leaders.

The two-player interaction is modeled as a influence blocking maximization (IBM). 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 while the second player, the defender will attempt to mitigate the attacker's influence. Vertices in the graph represent local leaders that each player can affect and edges represent the influence of one local leader on another. Specifically, each edge $e = (n, m)$ has an associated probability p_e that leader n will influence leader m to side with n's chosen player. Only uninfluenced nodes are eligible to be influenced.

Each player must decide upon a strategy to choose an action, also referred to as a seed set, where an action is a subset of the nodes in the graph ($S_a, S_d \subseteq V$). The size of the subset, ($|S_a| = r_a$, $|S_d| = r_d$), is given and models the resource constraint of the player. Nodes in S_a support the attacker and nodes in S_d support the defender, except nodes in $S_a \cap S_d$ which have a 50% chance of supporting each player. The influence then propagates synchronously, 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' probabilistically. 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 in that time step. Propagation continues until no new nodes are influenced.

For a given pair of actions, the attacker's payoff in an IBM is equal to the *expected* number of nodes that will support the attacker and the defender's payoff is exactly the opposite of the attacker's. The natural solution concept for such a zero-sum game is the Nash equilibrium mixed strategy over their pure strategies (subsets of nodes of size r_a or r_d). The resulting mixed strategy can then be sampled each time the counterinsurgency team is deployed. Our propagation model implicitly assumes that the opinions of local leaders *resets* between deployments to reflect the difficulty of maintaining local support.

Double Oracle Approach

Traditionally, zero-sum games such as the one we have modeled are solved with a Maximin linear program. In our domain, however, the action spaces for each player are too large to store in memory for anything but trivial problem sizes, necessitating an alternate approach. The one elected in Tsai et. al (2012) involves an iterative approach that builds the action sets incrementally and is known as a double oracle algorithm.

The double oracle algorithm features a Maximin linear program at the core which solves for the equilibrium strategies for each player given the current action set. It also possesses two oracles, one for the attacker and one for the defender, that each produce a best-response action to the current opponent strategy. At each iteration, then, the Maximin linear program solves for the current equilibrium strategies and passes this as input to each of the oracles that each return with a new action to add. These new actions are added into the current action set for each player and the Maximin linear program is run again. This repeats until convergence, which occurs when neither oracle is able to find a best-response that is superior to the equilibrium strategy found by the Maximin linear program. The algorithm is outlined formally in Algorithm 1.

Algorithm 1 DOUBLE ORACLE ALGORITHM

```
1: Initialize **D** with random defender allocations.
2: Initialize **A** with random attacker allocations.
3: repeat
4:     $(\rho_d, \rho_a) = $ MaximinLP(**D**,**A**)
5:     $\mathbf{D} = \mathbf{D} \cup \{$DefenderOracle$(\rho_a)\}$
6:     $\mathbf{A} = \mathbf{A} \cup \{$AttackerOracle$(\rho_d)\}$
7: until convergence
8: return $(\rho_d, \rho_a)$
```

The double oracle algorithm, when using two optimal best-response oracles, has been shown to converge to the Maximin Nash equilibrium (McMahan, Gordon, and Blum 2003). However, with heuristic oracles, the algorithm becomes a best-response dynamics method that no longer provides runtime or quality guarantees. Thus, the purpose of this work is to evaluate the performance of the proposed heuristic techniques under a wide range of parameters and identify key trends that would inform their use in practical application.

LSMI Oracle

In studying contagion, a measure of payoff values must be specified. Here we adopt the commonly used metric in influence maximization which is the expected number of nodes that have been influenced by each player. In our particular zero-sum game, the expected number of nodes that will be influenced by the attacker is the relevant measure of reward. Under the independent cascade model of propagation that we have used in this work, it has been shown that even determining the expected influence of one player is #P-Hard (Chen, Wang, and Wang 2010), necessitating approximate or heuristic techniques for larger problem sizes. A commonly proposed method involves Monte Carlo simulations of the propagation process for, typically, 10,000-20,000 trials to reliably estimate the expected influence. Theoretically, the Monte Carlo process can guarantee an arbitrarily small error bound with sufficient trials (Kempe, Kleinberg, and Tardos 2003). To find a best-response, as each of the oracles must do, the Monte Carlo technique can be used to determine the payoff of all possible actions and the best can be chosen. However, this is prohibitively slow, so an approximate best-response technique was used by Budak et. al (2011) that continues to use the Monte Carlo estimation technique.

The LSMI oracle implements the best-response technique used by Budak et. al (2011), but uses a heuristic estimate of the actual payoff of a given pair of actions to replace the Monte Carlo estimation technique. The algorithm for the heuristic estimate is given in Algorithm 2 and is based on the idea of ignoring low probability influences. Note that because L-Eval$(\cdot)$ is a heuristic estimation of influence propagation instead of an ϵ-approximation as the Monte Carlo scheme is, the algorithm no longer guarantees an approximation bound on the best-response generated.

Algorithm 2 L-Eval(V, S_a, S_d)

```
1: $InfValue = 0$
2: for $v \in (V - S_a - S_d)$ do
3:     $\mathbf{N} = $ GetVerticesWithin$\theta(v) \cap (S_a \cup S_d)$
4:     /* Prioritize sources by *lowest* hop-distance to $v$*/
5:     **S** =makePriorityQueue(**N**)
6:     $p_a = 0, p_d = 0$
7:     while $\mathbf{S} \neq \emptyset$ do
8:         $s = \mathbf{S}$.poll()
9:         if $(s \in S_a)$ then
10:            $p_a = p_a + (1 - p_a - p_d)\cdot$ Prob$(s, v)$, $p_d = p_d$
11:        else /* $s$ must be in $S_d$ */
12:            $p_d = p_d + (1 - p_a - p_d)\cdot$ Prob$(s, v)$, $p_a = p_a$
13:        end if
14:    end while
15:    $InfValue = InfValue + p_a$
16: end for
17: return $InfValue$
```

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. For readers interested in how the heuristic is incorporated into the algorithm from Budak et. al (2011), more detail is given in Tsai et. al (2012).

PAGERANK Oracle

PageRank is a popular algorithm to rank webpages (Brin and Page 1998), which we adapted 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, $\tau_u = \sum_v w_{u,v} \cdot \tau_v$

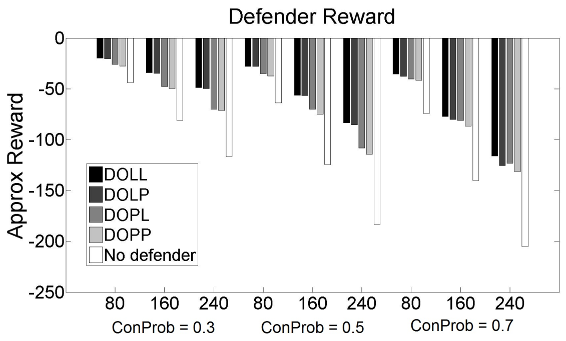


Figure 1: Preliminary test, r = 10, avg. = 0.3, s.d. = 0.1

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.

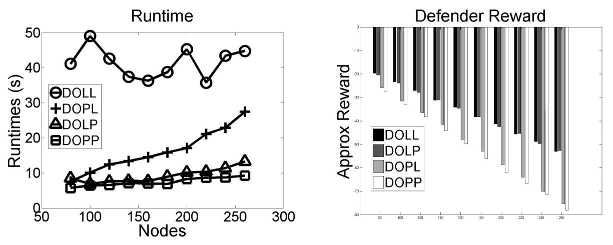
For the PAGERANK oracle, since the defender's goal is to minimize the attacker's influence, the defender oracle will focus on nodes incident to attacker sources $N_a = \{n | n \in V \wedge \exists e_{n,m}, m \in S_a\}$. Specifically, ordering the nodes of N_a by decreasing rank value, the top r_d nodes will be chosen as the best response. In the attacker's oracle phase, the attacker will simply choose the nodes with the highest ranks. Unlike the LSMI oracle, the PAGERANK oracle ignores payoff estimation entirely and generates a heuristic best-response to the current opponent strategy based purely on graph properties instead. While this results in much faster runtimes, the quality of solutions generated tend to be lower than those found by LSMI oracles.

Experiments

Three types of variations were explored in this work. First, we varied the size of the graph and kept all other parameters constant. Second, we varied the average contagion probability in the graphs at three separate graph sizes. Finally, we varied the standard deviation of the contagion probability in the graphs and again tested these at three separate graph sizes. All experiments featured a randomly generated scale-free graph, 10 resources per player ($S_d, S_a = 10$), and contagion probabilities on edges that were drawn from a normal distribution. Scale-free graphs were chosen due to their widespread use as proxies for general social networks and were generated according to the principle of 'preferential attachment' as introduced by Barabasi and Albert (1999). Our particular implementation adds edges between existing vertices and newly added vertices with a probability of $p = (\deg(v) + 1) / (|E| + |V|)$[1]. 100 trials were run for every data point shown.

Figure 1 shows a preliminary test that was conducted to provide a benchmark for the quality results. It shows the reward for the defender when each of the four algorithms is used as well as when no defender is present as well for graphs of size 80, 160, and 240 and with the average con-

[1]http://jung.sourceforge.net/doc/api/edu/uci/ics/jung/algorithms/generators/random/BarabasiAlbertGenerator.html



(a) Runtime (b) Quality

Figure 2: Scale-up results, r = 10, avg. = 0.3, s.d. = 0.1

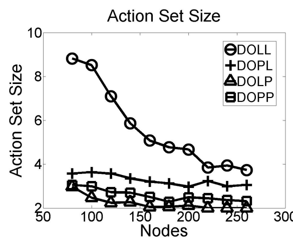
tagion probability set to 0.3, 0.5, and 0.7. As was done in previous work, the reward reported is the reward achievable by an adversary that best-responds to our algorithm's generated defender strategy by calculating the approximate best-response via the algorithm proposed by Budak et. al (2011). The solution is guaranteed to be within $\frac{1}{e}$ of optimal, making it a more consistent basis for comparison than any heuristic best-response. This reward was chosen because determining the optimal best-response for an attacker required far more than 20 minutes for each best-response. As mentioned, the graph sizes tested were limited to 260 nodes because for larger graphs even calculating the approximate best-response outlined above begins to take longer than 20 minutes as well.

As can be seen, all of the algorithms provide at least a 30-40% improvement in reward obtained as opposed to having no defender present across all of the cases tested. Since this was intended as a preliminary justification for the algorithms, we will provide more in-depth analysis of the solution quality of the algorithms in the following subsections.

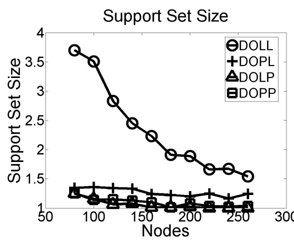
Graph size scale-up

The first set of experiments explored the impact of scaling up the size of the graph alone. Specifically, the four algorithms (all combinations of the LSMI and PAGERANK oracles) were run on randomly generated scale-free graphs with 80-260 nodes in increments of 20, with 10 resources and contagion probabilities drawn from a normal distribution $\mathcal{N}(0.3, 0.1)$. Graph sizes were limited to 260 nodes because the adversary best-response technique used to determine the defender's reward became too cumbersome for larger graphs.

Figure 2a shows the impact on runtime as the graph size is scaled up. As can be seen, the solution technique that features two LSMI oracles (DOLL) requires the longest run time at 40-50 seconds for all of the game sizes tested. Interestingly, there did not appear to be a consistent increase in runtime as was observed in the other 3 algorithms (each of which had at least one PAGERANK oracle). The other 3 algorithms were much faster, all requiring less than 30 seconds with a consistent trend as the graph size increases. DOPL requires more time than DOLP because of the fact that the defender PAGERANK oracle explicitly adapts to the attacker's strategy (only uses nodes adjacent to attacker nodes), while the attacker PAGERANK oracle does not. Previous work

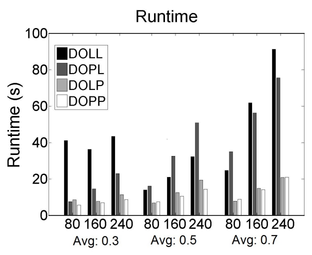


(a) Action set size

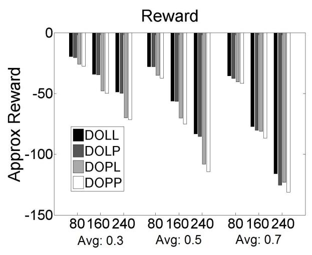


(b) Support set size

Figure 3: Scale-up results, r = 10, avg. = 0.3, s.d. = 0.1



(a) Runtime



(b) Quality

Figure 4: Contagion probability average results, s.d. = 0.1

explored scaling to larger graphs with more resources, but since this is not the focus of our work, we refer the interested reader to Tsai et. al (2012).

Figure 2b shows the impact on solution quality as the graph size is scaled up. Unsurprisingly, as the size of the graph increases, it becomes increasingly difficult for the defender to block the adversary's influence spread and the defender receives a correspondingly lower reward. As was seen in previous work, we also observe a large difference between algorithms that use a LSMI oracle for the defender as opposed to a PAGERANK oracle for the defender, with the latter providing much lower rewards. This is expected, due to the higher sophistication of the LSMI defender oracle as was noted in previous work.

Figure 3a shows the final number of actions in the defender's action set as the size of the graph is increased. The action set is defined as the number of actions available to the defender in the CoreLP phase of the double oracle algorithm and is exactly the number of new best-responses that have been found by the defender oracle. In the worst case, this would include all possible actions in the game, but as can be seen is generally far smaller, making the problem much more tractable. The attacker's action set size was always extremely similar if not identical to the defender's action set size.

Figure 3b shows a similar metric and features the number of actions in the support set of the final defender strategy. The support set is the set of actions that have non-zero probability in the final mixed strategy. Again, the final attacker support set size was always extremely similar if not identical to the defender's.

As can be seen, both the action set and the support set sizes are much larger with the DOLL algorithm than for any of the other algorithms. This is due to the sophistication of the LSMI oracles as opposed to the PAGERANK oracle. The PAGERANK oracles converge extremely quickly to a small set of actions and often do not generate new actions in response to new adversary strategies. This is especially true for the PAGERANK attacker oracle, since the defender oracle actually chooses nodes directly adjacent to the attacker. Thus, even when only one PAGERANK oracle is used, the algorithm overall converges quickly. The DOLL algorithm is iterating many more times than algorithms featuring the PAGERANK oracle, leading to the previous runtime result with DOLL being far slower than the other algorithms.

Furthermore, the trends seen in both Figure 3a and b show the size of the final action set and support set decreasing as the graph size is increased. This is due to the fact that as the graph grows larger, very few actions are useful for the defender to use to defend against the spread of the attacker's influence. For the attacker, randomization becomes less essential for the same reason. Thus, both players converge to a very small set of actions for the final mixed strategy.

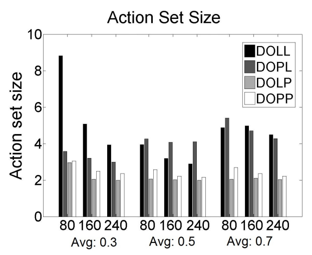
Contagion probability: Average

To explore the impact of changing the contagion probabilities on the four algorithms, we tested three different contagion probability averages for three separate graph sizes. Specifically, we ran all four algorithms with the contagion probabilities drawn from normal distributions $\mathcal{N}(0.3, 0.1)$, $\mathcal{N}(0.5, 0.1)$, and $\mathcal{N}(0.7, 0.1)$. The graph sizes tested were 80, 160, and 240 node random scale-free graphs with 10 resources allowed per player. We measured the same 4 metrics as in the previous section: runtime, solution quality, action set size, and support set size.

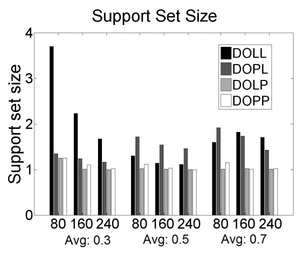
Figure 4a shows the results pertaining to runtime. The x-axis is divided into three sets of three bars each. Each set represents one setting for the contagion probability average (0.3, 0.5, 0.7) while each bar represents the runtime result for one algorithm. At averages of 0.5 and 0.7, consistent trends can be seen, with larger graphs taking longer and higher probabilities leading to longer runtimes for algorithms with LSMI oracles. This is because LSMI oracles speed up heuristic estimation by calculating only high probability influences, but when contagion probabilities are higher, this leads to many more nodes that must be processed by the algorithm.

For the case of 0.3, however, the trend is not consistent for the DOLL algorithm. Experiments suggest that with low contagion probabilities, two LSMI oracles continually find new best-responses to each other's strategies. This occurs because at low contagion probabilities, different parts of the graph interact minimally and the attacker is able to move to 'new' nodes and entirely avoid the defender, resulting in a cat-and-mouse game that requires many more iterations to converge than when a PAGERANK oracle is used.

Figure 4b shows the reward for the defender using the same approximate best-response technique described previously. Unsurprisingly, larger graphs lead to lower reward for the defender because it is harder to defend. Higher con-



(a) Action set size



(b) Support set size

Figure 5: Contagion probability average results, s.d. = 0.1

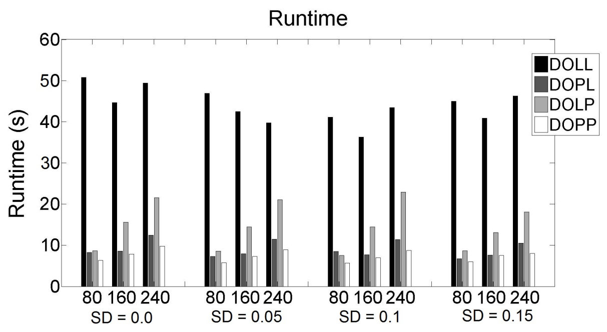


Figure 6: Contagion probability s.d. results, avg. = 0.3

tagion probabilities also result in lower defender rewards for the same reason.

As we noticed in the scale-up experiments, larger graphs lead to fewer actions in the action set as well as the final support set, as shown in Figures 5a and b. As mentioned, at the lowest contagion probability tested (0.3), the action and support set sizes are very large for DOLL, causing very high runtimes due to the many iterations required to generate the observed action sets.

Contagion probability: Standard deviation

Next we tested variations of the standard deviation of the normal distribution that the contagion probabilities on edges are drawn from. Specifically, we ran all four algorithms with the contagion probabilities drawn from normal distributions $\mathcal{N}(0.3, 0.0)$, $\mathcal{N}(0.3, 0.05)$, $\mathcal{N}(0.3, 0.1)$, and $\mathcal{N}(0.3, 0.15)$. These results, however, did not show statistically significant differences in the results when the standard deviation was changed under the particular parameter settings we tested. We only show the runtime results in Figure 6 to support this claim, but the quality, action set size, and support set size results all looked similarly homogenous across the different standard deviations tested.

Conclusion

Previous research in game-theoretic contagion blocking proposed heuristic algorithms to scale to realistic domain sizes. In this work, we perform a thorough investigation of the attributes of these algorithms and the solutions they generate by systematically varying the graph size, contagion probability distribution's average, and contagion probability distribution's standard deviation. Our experiments reveal that when the graph size increases, all of the algorithms converge in fewer iterations because the defender becomes unable to find strong blocking actions. Interestingly, low contagion probabilities provided difficult problem instances because the attacker could continuously find new actions that avoided the entire set of current defender actions until a sufficient number of actions had been added to both players' action sets, thereby requiring many iterations to converge. For the LSMI oracle, which increases in complexity as contagion probabilities are increased, this results in low contagion probability problems requiring high runtimes (due to more iterations), high contagion probability problems also requiring high runtimes (due to more time required per iteration), and a relatively easy middle region.

While these findings serve as a much more thorough investigation of the behavior of these heuristic algorithms, additional avenues for future analysis clearly exist. For example, only scale-free graphs were used in this work. While this is accepted as a good approximation for general-purpose social networks, many other social network structures have been observed in the literature. Also, sociologists have noted the importance of structural factors in the social networks such as the clustering coefficient, skew of degree distribution, degree correlation, and path lengths on the contagion behavior. In future work we will analyze the algorithms on graphs across the spectrum of these metrics and design new heuristics tailored for specific parts of the parameter space that remain challenging for existing methods.

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

[1999] Barabási, A.-L., and Albert, R. 1999. Emergence of Scaling in Random Networks. *Science* 286(5439):509–512.

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

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

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

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

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

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

[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.

[2012] Goyal, S., and Kearns, M. 2012. Competitive contagion in networks. In *STOC*.

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

[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.

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

[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.

[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.

[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.

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

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

[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.

[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.

[2011] Pita, J.; Tambe, M.; Kiekintveld, C.; Cullen, S.; and Steigerwald, E. 2011. Guards - game theoretic security allocation on a national scale. In *AAMAS*.

[2012] Shieh, E.; An, B.; Yang, R.; Tambe, M.; Baldwin, C.; DiRenzo, J.; Maule, B.; and Meyer, G. 2012. Protect: A deployed game theoretic system to protect the ports of the united states. In *AAMAS*.

[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.

[2012] Tsai, J.; Nguyen, T. H.; and Tambe, M. 2012. Security games for controlling contagion. In *AAAI*.

[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.