SAND2012-8911C

Security Games for Controlling Contagion Under Asymmetric Information: The Power of Simple

Paper #104

ABSTRACT

Many adversarial domains such as viral marketing feature a contagious component beyond the original actions taken. Counterinsurgency, which is the effort to mitigate support for an opposing organization, is one such domain that has been studied recently and past work has modeled the problem as an influence blocking maximization that features an influencer and a mitigator. The two players act on a graph, where nodes represent assets (such as local leaders in a society) and edges represent the influence of one asset on another. Each player is then allowed to choose some subset of the nodes from which to begin propagating support for their cause. Past work has introduced scalable heuristic techniques for generating effective strategies using a double oracle algorithm.

In these domains, however, graph structure is often not known with certainty and one party can have an informational advantage over the other that can, in theory, cause unbounded loss. We model this asymmetric information situation as a two-player zero-sum Bayesian game where each Bayesian type represents one possible graph instantiation. The mitigator is uncertain which of these types nature has selected, but the influencer acts with full knowledge of the type. Contrary to what has been assumed in past work in security games, we show through extensive experimentation that many common forms of uncertainty can be addressed near-optimally by simple heuristics such as solving a subgame with just one or two types. This implies that optimal strategies that may not model the full range of uncertainty in such situations are often actually very robust to uncertainty. Finally, based on the simple heuristics, we introduce novel techniques to improve the efficiency of the optimal double oracle algorithm.

Categories and Subject Descriptors

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

General Terms

Algorithms, Security, Performance

Keywords

Game theory, Social contagion, Influence maximization

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

1. INTRODUCTION

Social contagion and the spread of information have long been areas of great interest due to the widespread impact of such phenomena as viral marketing, rumor spreading and the Arab Spring [12, 15, 19]. Counterinsurgency, the competition for the support of local leadership, has also been studied as a game with two strategic players [7, 6, 20]. The key computational question is to decide which local leaders to influence to achieve each player's desired end: maximize influence for one player, and mitigate the first player's influence for the other player.

These 'counter-contagion' games have received recent attention in the security games literature [20] and has been modeled as a graph where nodes represent leaders and edges between the nodes representing the probability of influence. However, this line of research has not yet examined the impact of asymmetric information. Informational challenges abound in counterinsurgency, where the insurgents are typically an indigenous group that has an informational advantage and the mitigators often have uncertainty about their knowledge of the social network [7].

In our work, the mitigator's uncertainty about the graph structure is modeled as a Bayesian game with each Bayesian type representing a separate instantiation of the graph. The mitigator's strategy must now reason over the distribution of types. The influencer's (insurgent's) perfect knowledge of the graph structure allows him to specify a behaviorial strategy which conditions the strategy used on the specific type. We show that given incorrect information about even a single edge, a mitigator can suffer unbounded loss. Furthermore, quantifying the impact of changing a single edge is #P-Hard, making it extremely challenging to efficiently evaluate one strategy's performance on a slightly different graph.

In the past few years, many researchers have addressed uncertainty in security games using a Bayesian model and introduced more efficient algorithms for handling them [11, 13, 22]. Contrary to what has been assumed in past work, however, our experiments show that simple heuristics actually produce near-optimal rewards in our domain under numerous models of uncertainty. Furthermore, evidence suggests that this may be true in the areas explored by prior research as well. This implies that such extremely challenging problems may often be amenable to very efficient, near-optimal heuristic techniques. Finally, this also suggests that strategies determined without modeling uncertainty may actually be very robust to many forms of uncertainty.

There are, however, some real-world domains that require guarantees on the quality of the solutions produced and heuristics cannot be used. For these problems, we introduce novel techniques for optimally solving Bayesian models of counter-contagion games with large numbers of types based on the high-performance heuristics mentioned. First, we develop the Verified Response which pro-

vides increased efficiency by re-using best-responses generated by other types. Second, we develop Centrality Similarity Sampling, which first clusters the types based on graph-theoretic measures and samples these to solve a smaller, representative subset of the original Bayesian game before adding the remaining types and seeding them with the actions found in the subgame. We show both techniques provide efficiency gains in practice while retaining optimality.

2. RELATED WORK

Recent work in game-theoretic security allocation have also dealt with domains that were modeled as graphs [1, 10, 4], however their actions were all deterministically defined and did not feature a probabilistic contagion component. The work in uncertainty in security games is also relevant [11, 13, 22], but once again do not feature the contagion component found in our domain.

This contagion process has been studied outside of the security games literature and is known as 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. This class of problems were first introduced as a discrete maximization problem by Kempe et al. (2003) who showed submodularity of the maximization problem, enabling a greedy approximation. This work has been followed-up by numerous proposed speed-up techniques [3, 14, 15].

Two-player variants of influence maximization have been studied as well, one of which is known as influence blocking maximization problems and are equivalent to the counter-contagion games we study. These models have been explored with both independent cascade and linear threshold models of propagation [2, 5], however, work in this area has generally focused only on the defender's best-response problem. The exception is Tsai et al. [20] which addresses the algorithmic challenge of finding equilibria strategies. Finally, Hung et al. (2011) and Howard (2010) also model counterinsurgency and attempt to optimize against a strategic adversary. However, none of these works model the uncertainty that is critical in domains such as counterinsurgency.

3. EXAMPLE DOMAIN

Counterinsurgency is, perhaps, the most important application of our model. In particular, we can view counterinsurgency as a game between two parties: the insurgents and the peacekeepers [8, 6, 7]. While this interaction is multi-faceted, we focus on one particular aspect of it: influence spread. In this context, the insurgents aim to spread their views, unrest, etc. among the local population, whereas the goal of the peacekeepers is to minimize this spread by engaging in their own information and influence campaign. A crucial feature of this setting is that influence does not just spread directly from the insurgents and peacekeepers: rather, people influence each other, and this *indirect* spread of influence is arguably just as important as direct influence, particularly since both parties have limited resources to engage in their respective campaigns. We model such spread of individual influence in a stochastic fashion, such that any two individuals who are socially connected can influence each other with some fixed, and independent, probability.

Clearly, the structure of the social influence graph is critically important in this setting. *Knowledge* of this structure, however, is asymmetric: the insurgents, who deeply understand local culture and social patterns, likely have considerably more information about the social network than the peacekeepers, who are often foreigners. We model this by assuming that the social network is known by the insurgents exactly, whereas the peacekeepers have a probability distribution over the structure of the graph (e.g., over which edges are present).

4. MODEL

4.1 Asymmetric Information Game

We model counterinsurgency as a two-player Bayesian zero-sum game situated on a graph in which two players, the influencer (denoted by I) and the mitigator (denoted by M) compete to maximize influence over the nodes. Formally, let $G = (V, E)$ be a graph with nodes V and edges E, and for each edge $(i, j) \in E$, let p_{ij} be the probability that node i's opinion will influence node j. We model propagation of influence in the graph as a synchronized independent cascade process [12] as follows. Suppose that the influencer initially attempts to influence a subset of nodes $S_I \subseteq V$ to his cause, and the mitigator's initial influence is aimed at a subset of nodes $S_M \subseteq V$. For nodes $v \in S_I \cap S_M$ which both players initially try to influence, initial "activation" (e.g., actual opinion adoption) happens in either player's favor with equal probability, while all the remaining nodes adopt the view of (are activated by) the player who directly targets them. Next, we activate all edges (i, j) in the graph with the corresponding probability of influence, p_{ij}. At that point, the influence process proceeds through a sequence of iterations. In each iteration, if a node j has not yet adopted an opinion but has active edges to neighbors who have, j either adopts the opinion of these neighbors when it is unanimous, or adopts each opinion with equal probability if j's active neighbors disagree. Viewing now the initial target nodes S_I and S_M as the strategies of the players I and M respectively, let $\sigma(S_I, S_M)$ be the expected number of nodes that adopt the influencer's opinion following the independent cascade process described above. We define the utility of the influencer to be $U_I(S_I, S_M) = \sigma(S_I, S_M)$.

We now depart from the model of Tsai et al. [20] by relaxing the complete/symmetric information assumption. Specifically, we assume that the influencer knows the actual influence graph G exactly, while the mitigator is uncertain about its true structure, and only knows the probability distribution over possible graphs. Let λ be an index identifying a particular graph G_λ, and let us make explicit the dependence of the expected influence on the graph, denoting it by $\sigma(S_I, S_M, \lambda)$. Finally, we denote by P the probability distribution over λ, with P_λ the probability that the true graph is the one identified by λ. From the mitigator's perspective, the influencer's decision will depend on his type, that is, on the true graph which the influencer observes. Thus, we view the influencer's strategy S_I as a function of λ, with S_I^λ denoting the influencer's strategy when his type is λ. The mitigator's utility is then $U_M(S_I, S_M) = -E_{\lambda \sim P}[\sigma(S_I^\lambda, S_M, \lambda)]$.

4.2 Generative Models for Graphs

Over the years, numerous stochastic generative models for graphs have been proposed to generate synthetic instances of graphs that resemble real social networks [16]; some of the best known examples of these are the preferential attachment process, which generates scale-free graphs, and the process of generating small-world networks pioneered by Watts and Strogatz [21]. Recently, a new generative model, BTER, has been developed, and the authors convincingly demonstrated that this model matches the important properties of real-world networks, such as the degree distribution and the distribution of clustering coefficients, far better than any previously proposed methods [18]. BTER graphs feature a scale-free collection of densely clustered community structures (dense Erdös-Rényi subgraphs), which are sparsely interconnected. Significantly, the parameter space of BTER models can generate graphs that ex-

hibit both the scale-free degree distribution, as well as those that have small-world network properties described by Watts and Strogatz. As such, we use these as the primary generative model for graphs in our evaluation, although in many instances we present results for the preferential attachment and small-world (Watts-Strogatz) models as well.

4.3 Models of Uncertainty

We consider several specific models of mitigator's uncertainty about the graph. Primarily, we explore the following three models. The first model, *Random Edge Uncertainty*, is the simplest: the mitigator has perfect information about the nodes in the graph, and is uncertain about a set of possible edges (i.e., about a set of pairwise relationships in the social network). The second model of uncertainty, *Inter-community Edge Uncertainty*, applies only to BTER graphs and models the mitigator's uncertainty about the inter-community edges. Indeed, it is very likely that the mitigator is relatively poorly informed about inter-community connections, since an accurate understanding of such relationships requires considerable information about the cultural patterns that govern interpersonal and inter-group communication. Note that in both of these classes of uncertainty, we may have a type λ for each possible subset of edges in the graph, and, thus, the number of types could be as large as $2^{|E|}$. The third class of uncertainty that we consider, *Influential Node Uncertainty*, models information asymmetry about who the most connected *nodes* are in the graph, motivated by the fact the most socially connected and influential nodes are not always readily identifiable, since their identity is very much the function of local culture which is much more familiar to the influencer than the mitigator. Specifically, we start with a baseline graph, then, for each, type, choose a set of j nodes and add k additional edges from each of these nodes to others. In this model, the number of types is at most ${}_vC_j$.

We have also considered several other classes of uncertainty, such as uncertainty about the identity of the nodes who connect different communities, and uncertainty in which types involve both intra- and inter-community edges, but the experimental results for these are not markedly different than the results we obtain with the three primary classes described above. We therefore report only on the three primary uncertainty classes and refer the interested reader to our website: http://aamas2013.webs.com.

5. THE CHALLENGES OF UNCERTAINTY

Quantifying known model uncertainty is a crucial aspect of modeling and analysis [9], and the Bayesian game framework that we adopt here is one common way to do so in game theoretic settings. Naturally, we wish to ask how much do we gain by explicitly modeling uncertainty about the graph in the context of counter-contagion games, and at what cost? First, we address the question of added value by showing in the example that follows that, in general, ignoring uncertainty can yield a solution that is arbitrarily poor for the mitigator. Consider the graph shown in Figure 1 in which the edge from A to B is uncertain, $N > M$, and both players have a single resource. There are two ways in which the mitigator can ignore uncertainty: he can either assume that the edge does exist, or assume that it does not. Suppose that the influencer chooses to influence node A with probability 1. First, let the mitigator ignore the edge from A to B. In this case, his best response is to influence node C with probability 1. However, if the edge actually *does* exist, the mitigator's actual loss amounts to $\frac{N}{2}$, as compared to $\frac{M}{2}$ he would lose by influencing B. Since the difference between N and M is arbitrary, the resulting difference in utility can be made arbitrarily large.

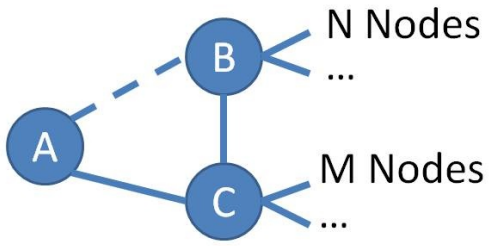


Figure 1: Example showing unbounded loss

Now, let the mitigator assume that there does exist an edge from A to B. In this case, his best response is to influence B with probability 1. However, if the edge in fact does not exist, the mitigator's loss would be $\frac{M}{2}$, whereas an optimal strategy of influencing C in this case would result in a loss of only a single node A. Since M is arbitrary, the amount of this loss can be made arbitrarily large.

Having shown that explicitly modeling uncertainty in our case can have very high value, we turn to the question of complexity. Here, the result is very clear, and very negative. At a high level, the challenge of efficiently reducing the runtime of computing equilibria in our setting lies in quantifying the impact of even small changes in the graph structure. If this could quickly and accurately be determined, then types could be efficiently clustered and bounds could be placed on the quality loss. The fact that computing the expected influence is #P-Hard [3] should already give us pause. Indeed, a simple corollary of this result reveals that such quantification is intractable in general.

PROPOSITION 1. *Computing the difference in expected influence for a given seed set even when a single edge is added to a graph is #P-Hard.*

PROOF. We prove this by contradiction. Call the difference function, $d(S, G, e)$, where S is the given seed set, $G = (N, E)$ is the base graph, and e is the edge to be added. Assume $d(\cdot)$ can be calculated in polynomial time. Define a graph $G' = (N, \emptyset)$. $\sigma_{G'}(S)$ can be calculated in polynomial time. Repeatedly add edges from E to G' until G is fully reconstructed, computing $d(S, G, e)$ in each iteration. Since the total influence of G is $\sum_{e \in E} d(S, G, e)$, this implies that we have computed influence in polynomial time, since only $|E|$ iterations were executed, which contradicts that fact that computing the expected influence of a graph is #P-Hard. □

6. DOUBLE ORACLE ALGORITHM

In the previous section, we have shown that the problem of uncertainty in the counter-contagion games we study here is both significant and extremely complex. Although the problem we face is in fact a zero-sum game, and any zero-sum game can be solved using a linear program (MaximinLP below), doing so in our case has three basic problems. The first is that payoff estimation requires determining the value of $\sigma(S_I^\lambda, S_M, \lambda)$, which has been shown to be #P-Hard [3]. Thus, constructing the payoff matrix for the MaximinLP is non-trivial. Second is that the strategy sets for both players are exponentially large, and thus even storing the entire payoff matrix is impractical. Compounding the issue of scale is the fact that in our Bayesian game, each type has its own exponentially large payoff matrix.

The first problem was addressed in prior research [20] and we briefly summarize the approach here. The standard technique for

estimating $\sigma(S_I^\lambda, S_M, \lambda)$ is to use a Monte Carlo simulation of the propagation process. However, since this is prohibitively slow in practice, Tsai et al. developed the LSMI heuristic payoff estimator to replace it and show that solutions generated with their heuristic payoff estimation produce strategies that perform well when used in the true game. Since the additional dimension of having multiple influencer types compounds the scalability challenge we face, we use the LSMI payoff estimator in the experiments shown here. We conducted limited testing using the Monte Carlo estimation as well but omit them as they exhibit the same trends as the LSMI results.

For the second problem of exponential strategy sets, the double-oracle algorithm introduced by Halvorson et al. [4], provides a solution. A double-oracle algorithm begins with a small subset of pure strategies for each player and computes equilibrium strategies for each player for the current subgame. In each iteration, best response strategies are computed for both players and added to the existing subgame. If the best responses already exist in the subgame, then the current equilibrium is an equilibrium of the true game and the algorithm converges. If not, then the strategies are added to the subgame and the process continues until convergence. In the context of Bayesian games, Halvorson et al. propose computing the best response for every player type, which in our case means that we compute the influencer's best response for each type (graph), and add all of these to the MaximinLP in each iteration.

1: Initialize **M** with random mitigator allocations.
2: Initialize each $\mathbf{I}_\lambda \in \mathcal{I}$ with a random influencer allocation.
3: **repeat**
4: $(\rho_M, \rho_\mathcal{I})$ = MaximinLP(**M**,$\mathcal{I}$)
5: **M** = **M** $\cup$ {MitigatorOracle($\rho_\mathcal{I}$)}
6: **for** $\{\lambda \in \Lambda\}$ **do**
7: r = {InfluencerOracle(ρ_M, λ)}
8: $\mathbf{I}_\lambda$ = $\mathbf{I}_\lambda \cup$ r
9: **until** convergence
10: **return** (ρ_d, ρ_a)

Algorithm 1: Double-Oracle Algorithm for Bayesian zero-sum games

Algorithm 1 shows the full double oracle algorithm for Bayesian zero-sum games introduced by Halvorson et al. The double oracle algorithm begins by initializing the mitigator and each influencer type with random actions. This subgame is solved via the call to MaximinLP with the corresponding mitigator and influencer equilibrium strategies stored in ρ_M and $\rho_\mathcal{I}$ (line 4). Then the mitigator's best-response oracle is called to determine the best action against the current influencer strategy. Then the algorithm iterates across all the influencer type best-response oracles and generates new actions to add to each subgame. The process then repeats until convergence.

When the number of types is small, the approach by Halvorson et al. works well. In our case, however, types correspond to instances of a graph, and for all models of uncertainty that we consider, the number of types grows exponentially in the size of the graph. Since computing a best response for a given type requires a non-negligible amount of computation, having to do this for every type will simply not scale. To tackle this problem we will present two sets of results in the following sections. First, we show empirically that simple heuristics actually produce near-optimal solutions. Second, based on the power of simple heuristics, we provide two additional enhancements to the double-oracle algorithm that maintain optimality while offering runtime improvements in practice.

7. THE POWER OF SIMPLE

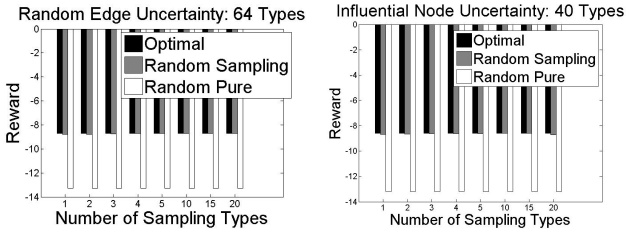
Our hardness results about the Bayesian counter-contagion problem that we study seem to suggest that scalability in the number of types and, thus, the size of the graph may be rather elusive. We now demonstrate through an extensive experimental study that in fact by sampling a tiny fraction of graph types from P_λ we can arrive at a nearly-optimal solution for a variety of generative models of social networks, as well as uncertainty about these. At face value, this is a shocking result, particularly since considering only a single type (i.e., no uncertainty) we already showed can be arbitrarily suboptimal. (However, it is sometimes still valuable to solve a given instance to guaranteed optimality as fast as possible. We deal with that problem separately below.)

7.1 Scale-up of Sampled Types

First, we examine the performance of the strategy obtained by optimally solving a subgame that includes only a randomly drawn subset of the Bayesian types, a technique we will refer to as *Random Sampling*. We increase the number of types the algorithm is allowed to include in the subgame and show that even with extremely few types the algorithm generates a near-optimal strategy. All results are an average of 20 trials and were run on machines with CPLEX 12.2, 2.8GHz CPU, and 4GB of RAM. Unless otherwise stated, all experiments were conducted with contagion probabilities on edges drawn from a $\mathcal{N}(0.4, 0.2)$ distribution, each player was allowed two nodes in their seed set, and the payoffs for each player are estimated using the LSMI heuristic introduced by Tsai et. al [20]. Experiments conducted using the Monte Carlo payoff estimation produced very similar results to the ones shown here, but could not be meaningfully scaled up so we omit these results. Since an optimal benchmark is necessary, the best-response oracles iteratively evaluate each available action to determine the best action instead of using the greedy hill-climbing approaches that are popular in influence maximization. Finally, graph sizes were kept at 40 nodes, the number of uncertain edges was kept at 6 ($2^6 = 64$ types) for Random Edge uncertainty results, and the number of types was kept at 40 for Influential Node uncertainty results to provide room for scaling up the degree of uncertainty modeled.

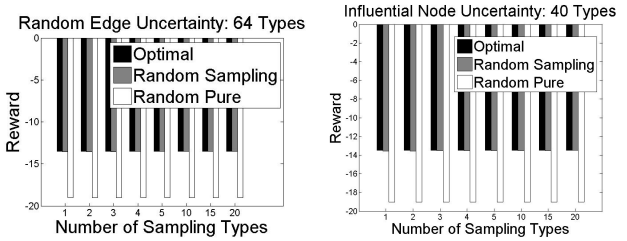
In each of Figures 2, 3, and 4, we show the reward achieved by *Random Sampling* compared to optimal as the number of sampled types increases. As an additional benchmark, we also show the reward obtained by a mitigator that randomly chooses a pure strategy to play (Random Pure). The experiments show results across three different generative graphs commonly used to model social networks: scale-free, small-world, and BTER graphs. For each graph type, we show results using two forms of uncertainty, Random Edge uncertainty and Influential Node uncertainty. For BTER graphs, we also examined the same scale-up with respect to Inter-community Edge uncertainty, but we omit the figure as it reveals the same trend as the results shown here. Normally, we would expect both Random Pure and *Random Sampling* to do very poorly relative to the optimal solution, but this was not the case at all.

For all three figures, the x-axis shows the number of types the algorithm is allowed to sample and the y-axis shows the mitigator's expected reward. Note that the x-axis results are not linear and actually show results for 1-5 samples then increases in increments of 5. The phenomenal performance of even just a single type is highly unexpected. As Figure 2 shows, under both forms of uncertainty for scale-free graphs, even a single sampled type is already quite close to optimal in these games with 64 and 40 types total. Similar results can be seen in Figure 3, which shows the results for small-world graphs. For BTER graphs, under Influential Node uncertainty, all algorithms perform very poorly (yet nearly equally



(a) Random edge uncertainty (b) Influential node uncertainty

Figure 2: Reward comparison, Scale-Free graphs



(a) Random edge uncertainty (b) Influential node uncertainty

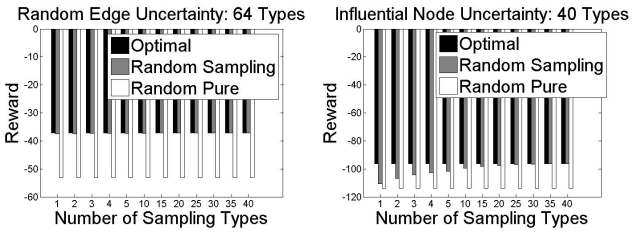
Figure 3: Reward comparison, Small-world graphs

poorly) because it is so difficult to mitigate the influencer. The optimal solution loses around 100 points of reward (as opposed to 8-15 in scale-free and small-world graphs) and only beats Random Pure by approximately 10%.

7.2 Scale-up of Uncertainty

In light of our findings, we added two additional algorithms to the mix to evaluate performance as the degree of uncertainty is scaled up in a fixed game size. First, we include *Max Prob*, which simply chooses the highest probability type and solves the subgame with that single type as an even simpler alternative to *Random Sampling*. Second, we attempt to improve upon *Random Sampling*in the rare cases when the performance is slightly suboptimal with a novel technique that builds upon the sampling idea, Centrality Similarity Sampling (CSS). CSS attempts to improve upon *Random Sampling* by intelligently choosing the N types that are used to create the subgame that we solve. CSS clusters the types via k-medians clustering and uses only a single type per cluster as a representative in a smaller Bayesian game where each type inherits the cumulative probability mass associated with its cluster. Ideally we would like to use payoff matrix similarity as the basis for clustering, however, we do not ever expand the entire payoff matrix in a double oracle formulation and even if we did it would be prohibitively large to use. Instead, we measure similarity by evaluating the difference between the centrality of each node of both graphs as measured by PageRank. The resulting strategies generated by these heuristics are evaluated against the actual type distribution of influencers.

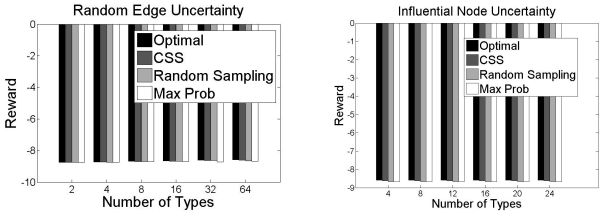
As before, we show results for scale-free, small-world, and BTER graphs as the degree of uncertainty is scaled up but the game size is kept fixed. For all three graph types, we show results under Random Edge and Influential Node uncertainties. For BTER graphs, we also vary the density of connections within the communities ($\rho = 0.5$ or $\rho = 0.9$) and show results with Inter-Community Edge uncertainty. For Random Edge uncertainty we varied the number of uncertain edges from 1 to 6 (2 to 64 types), for Influential Node uncertainty we varied the number of types from 4 to 40 in increments of 4, and for Inter-Community Edge uncertainty we once again varied the number of uncertain edges from 1 to 6. For Influential Node uncertainty, each type randomly selected 3 nodes and randomly added 4 edges to each of them. The number of types was kept low because of the poor scalability of the optimal algorithm.



(a) Random edge uncertainty (b) Influential node uncertainty

Figure 4: Reward comparison, BTER graphs

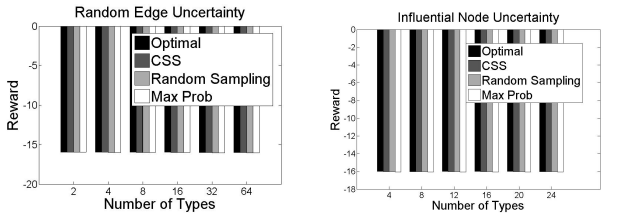
Figure 5 shows the results for scale-free graphs under each model of uncertainty. As can be seen, as we scale up the uncertainty the sampling techniques continue to perform extremely well and stay neck and neck with the optimal solution. Even with a total of 64 types, a *random* sample of *two* types and performs nearly optimally. Even using the strategy generated by solving against a single type (*Max Prob*) performs nearly optimally. The same is true of the small-world graph tests shown in Figure 6, as evidenced by all four bars staying extremely close through both scale-up experiments. BTER graphs under Random Edge and Inter-community Edge uncertainty (Figures 7 and 8) show very similar results with minimal performance degradation. As can be seen in Figure **??**, under Influential Node uncertainty the different techniques seem to perform slightly differently, but the story is once again the same as noted previously. All of the algorithms perform extremely poorly because it is simply too difficult to mitigate a strategic influencer in this situation. Even in the worst case, the sampling algorithms perform within 10% of the optimal solution, with CSS generally staying within 5%. Finally, as can be seen by the side-by-side comparison for the BTER graphs, the community density affects the overall difficulty of the problem (lowers reward for the mitigator across the board) but does not seem to create additional difficulties for the sampling techniques relative to the optimal soluation.



(a) Random edge uncertainty (b) Influential node uncertainty

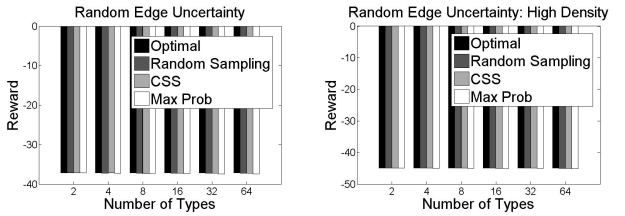
Figure 5: Reward comparison, Scale-Free graphs

Although not shown here, additional experiments were conducted that explored more forms of uncertainty and other parameter variations. These included uncertainty over a mixture of intra- and inter-community edges and a form of uncertainty wherein each type possessed a random set of N inter-community edges. We also created resource imbalances by testing games in which the mitigator was allowed 3 or 4 nodes against the influencer's 2. Similarly we tried the opposite, with the influencer being given 3 or 4 nodes against the mitigator's 2. We also varied the distribution of con-



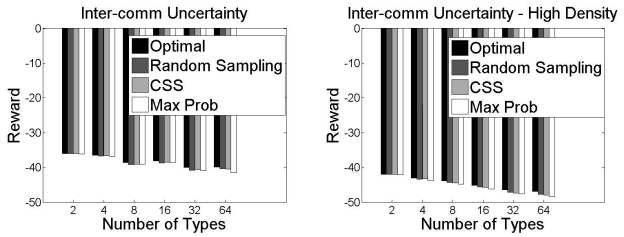
(a) Random edge uncertainty (b) Influential node uncertainty

Figure 6: Reward comparison, Small-World graphs



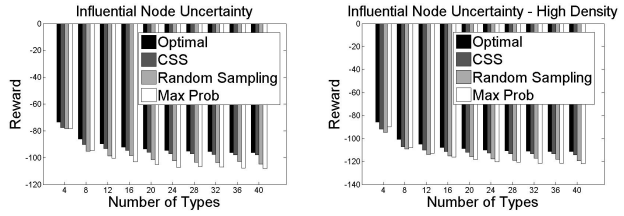
(a) Low community density (b) High community density

Figure 7: Uncertainty about a set of random edges



(a) Low community density (b) High community density

Figure 8: Uncertainty about a set of inter-community edges



(a) Low community density (b) High community density

Figure 9: Uncertainty about which nodes are highly connected

tagion probabilities on edges from the 0.4 we set it to for the experiments shown to 0.7 to examine the impact of changes in contagion probability. For BTER graphs we also examined multiple degree distributions in addition to the variation of community density that we show here. All of these variations, however, resulted in a multitude of graphs that looked extremely similar to the edge uncertainty graphs shown here, with simple heuristics performing extremely close to optimally. The results for the full set of experiments conducted, including those not shown, are available online at our website: http://aamas2013.webs.com.

7.3 Analysis

The results shown are surprising in their extremity, especially in light of the worst case performance result presented in Section 5. We now delve deeper into why this is occurring in these games. A plausible hypothesis is that the problems are simple to begin with and that most actions provide very high reward to the mitigator. To examine this, we plot the distribution of performance of *pure* strategies available to the mitigator by evaluating the expected reward obtained by each of the mitigator's pure strategies against the best response of the entire range of influencer types. To ease analysis, we bucket the rewards obtained into integer values. We show the results for two prototypical game instances to illustrate our findings. As can be seen in Figure 10a, the majority of actions lie in two clusters near -67 and -57. Figure 10b shows a distribution resembling a normal distribution. Neither of these shows a high percentage of high-reward actions, indicating that the initial hypothesis is incorrect and that substantial value is gained by optimally solving a subgame composed of a subset of types.

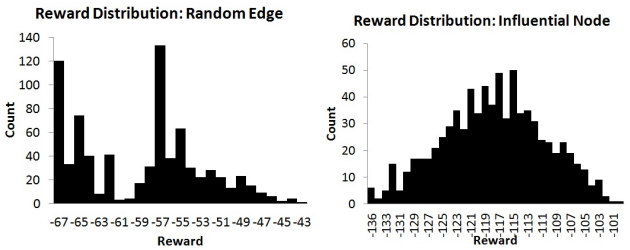
To better understand the high performance of the simple sampling approach, we examine the type-by-type performance of a given strategy. First, we solve a subgame against a single type optimally and evaluate that strategy's performance on each of the remaining Bayesian types. This is compared against the type-by-type performance of the optimal strategy over the full Bayesian game. These experiments were run on a single instance each to illustrate the trends found. These results are shown in Figures 11 and 12, which show the mitigator's reward on the y-axis and the type ID on the x-axis. Figure 11 shows the results under Random Edge uncertainty and Figure 12 shows the results under Influential Node uncertainty. In the 'Single Type Strategy' graphs, the leftmost type shown is the type that was solved optimally.

In Figure 11a, notice that the optimal strategy for the single solved type actually performs nearly identically against all other influencer types. The same is also true of the optimal strategy shown in Figure 11b, with both strategies generating approximately equal rewards. In Figure 12a, on the other hand, the difference is quite substantial across the types, with many actually performing much more poorly. Notice, however, that even the optimal strategy, shown in Figure 12b, shows a similar result with the reward against most types being very low. As noted before, this is due to the fact that this problem is much harder. The differences between the types were much more substantial for this form of uncertainty with 3 randomly selected nodes having 4 edges added to them each as compared to Random Edge or Key Edge uncertainty, where the maximum difference was 6 edges in total and the average differences was far smaller. This is only evident in the BTER graphs because the added edges sometimes connect multiple communities that were previously not connected, causing sharp changes in payoffs. Thus, as shown previously, since optimizing against all types is also extremely challenging, the overall expected rewards for the heuristics still perform very well relative to optimal.

Our analysis suggests that under the naturalistic models of uncertainty explored here, most problems are very well represented by a single type. This was also true for the two additional types of uncertainty that were tested but not shown. In the case when this is not true and types differ substantially, they are so difficult that all algorithms perform poorly.

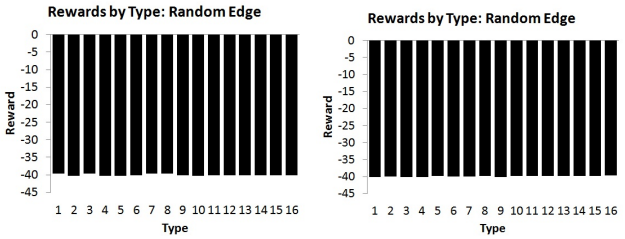
8. EXACT ALGORITHMS REVISITED

We presented extensive evidence that, in fact, sampling only a few types achieves nearly optimal utility for the mitigator. Since some domains call for guarantees on the solution quality, we now suggest two enhancements to the optimal double-oracle algorithm discussed in Section 6 that leverage this insight. First, we propose



(a) Random Edge uncertainty (b) Influential Node uncertainty

Figure 10: Distribution of pure strategy performance



(a) Single Type Strategy (b) Optimal Strategy

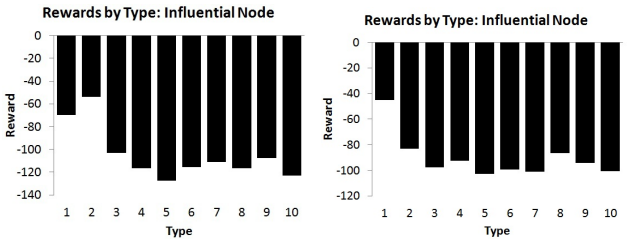
Figure 11: Performance distribution, Random Edge uncertainty

the *Verified Response* algorithm, which aims to take advantage of intuition that many graph types are sufficiently similar in the resulting payoff structure and, therefore, yield similar best responses for the influencer. Indeed, even if the best response is not the same, the best response for one type may often be at least a *better* response for the other type, which is sufficient for the double oracle algorithm to make progress. The *Verified Response* algorithm combines this with the fact that verifying whether or not a given action is a better response for a type is much faster than determining a best response directly.

1: Initialize **M** with random mitigator allocations.
2: Initialize each $\mathbf{I}_\lambda \in \mathcal{I}$ with a random influencer allocation.
3: Initialize $\mathbf{A} = \emptyset$ // stores all influencer best-responses
4: **repeat**
5: $(\rho_M, \rho_\mathcal{I}) =$ MaximinLP(**M**,$\mathcal{I}$)
6: **M** = **M** $\cup$ {MitigatorOracle($\rho_\mathcal{I}$)}
7: **for** $\{\lambda \in \Lambda\}$ **do**
8: r $= \emptyset$
9: **for** {a $\in$ **A**} **do**
10: **if** {isBetterResponse(ρ_M,a,λ) **then**
11: r = a
12: break
13: **if** {r $= \emptyset$} **then**
14: r $=$ {InfluencerOracle(ρ_M, λ)}
15: **A** = **A** $\cup$ r
16: $\mathbf{I}_\lambda$ = $\mathbf{I}_\lambda$ $\cup$ r
17: **until** convergence
18: **return** (ρ_d, ρ_a)

Algorithm 2: Verified Response Modification

Algorithm 2 shows the full double oracle algorithm for Bayesian zero-sum games that implements the *Verified Response* modification in lines 9-12. Normally, the standard double oracle algorithm then iterates across all the influencer type best-response oracles and generates new actions to add to each subgame. However, since



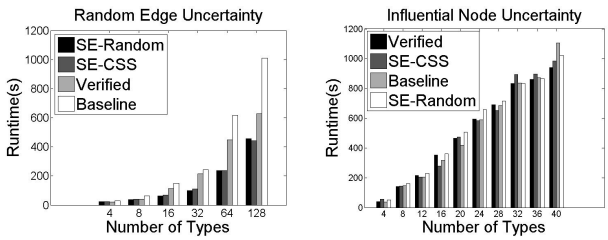
(a) Single Type Strategy (b) Optimal Strategy

Figure 12: Performance distribution, Influential Node uncertainty

these calls are computationally expensive, the *Verified Response* modification adds an intermediate step. Instead of directly calling a type's best-response oracle, the *Verified Response* heuristic first checks whether any previously generated best-response (by any type) is a *better* response for the current type. This can be done efficiently by comparing against the reward achieved by this type in the last call to MaximinLP.

The second technique we introduce is based on the fact that using only a few types already yields a good solution. Thus, using the resulting solution as, essentially, a warm start for the double-oracle algorithm should reduce the number of required iterations and, therefore, reduce total computation time. Specifically, we propose an algorithm, *Subgame Expansion* , which first solves the subgame that is induced by sampling several influencer types. Then, once this approximate game is solved optimally, we add all the remaining types, each seeded with the actions already generated in the startup phase, and proceed using Algorithm 2 from that point on.

Now we evaluate the runtime efficiencies gained by *Verified Response* and *Subgame Expansion*. Specifically, we evaluate their performance on BTER graphs on two different forms of uncertainty to demonstrate their performance. The results are shown in Figure 13 and show that for Random Edge uncertainty the runtime is improved dramatically by our techniques while for Influential Node uncertainty the performance is inconsistent. This is in line with expectations from our experiments in Section 7.3 that showed Influential Node uncertainty produces types that are quite different. This violates the fundamental assumption of both *Verified Response* and *Subgame Expansion* that best-responses for one type will at least be good responses for another type. While this did not meaningfully impact the solution quality of the heuristics, optimal algorithms based on this principle will sometimes suffer longer runtimes.



(a) Random Edge uncertainty (b) Influential Node uncertainty

Figure 13: Runtime comparison

9. DISCUSSION

The CSS algorithm developed here could have been presented in the same form as many past researchers have showcased their work and compared against only the optimal algorithm. Removing the *Random Sampling* and *Max Prob* results, we could have shown phenomenal runtime improvements at virtually no cost to solution quality and ended with a very impressive result. However, the additional experiments that have been conducted in this work have revealed an unexpected reality - that simple heuristics work extremely well in this domain across a variety of models of real-life uncertainties.

Indeed, a closer examination of previous literature in security games that addresses uncertainty reveals that similar phenomena may have been true elsewhere. In Jain et. al [11], the authors provided a novel algorithm to handle large Bayesian Stackelberg games (HBGS), but do not compare against other simple solution techniques at all. Interestingly, they noted that allowing the algorithm to solve for approximate solutions barely impacts solution quality but improves scalability tremendously. In Yin and Tambe [22], the authors provide a novel algorithm (HUNTER) for optimally handling Bayesian Stackelberg games with many types. While the algorithm is orders of magnitude faster than previously proposed optimal algorithms, the authors report that BRASS, a far less complex solution method [17], achieves an average loss of 0.7 in a game where the range of rewards for optimal solutions ranged from -26 to 17 compared against their algorithm. Again, the authors do not provide a benchmark 'simple' approach to demonstrate the practical gain achieved by an optimal solution. While the guarantees provided by optimal and approximate algorithms are necessary in some domains, one is left wondering if very simple heuristics may actually provide near-optimal results in many of these domains as well.

By no means does our work dispute the fact that extremely large Bayesian zero-sum games remain very challenging to solve well in general and there are certainly many problem classes that are not amenable to simple heuristics. In Kiekintveld et al. [13], for example, the authors introduce several techniques for handling large numbers of Bayesian types to address payoff uncertainty and they show that simple techniques do not perform near-optimally. Our work stresses the need to perform this type of verification of whether or not simple techniques work before embarking on extensive algorithmic gymnastics to achieve runtime improvements on minimal gains in solution quality. Although we have provided some analysis of why this occurs in our domain, this is only the beginning of research in this direction and clearly more work is needed. Finally, our findings give hope that many very challenging problems in computational game theory may actually be very effectively addressed by simple techniques: the power of simple.

10. REFERENCES

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

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

[3] 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.

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

[5] X. He, G. Song, W. Chen, and Q. Jiang. Influence blocking maximization in social networks under the competitive linear threshold model. In *SDM*, 2012.

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

[7] 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.

[8] 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.

[9] R. L. Iman and J. C. Helton. An investigation of uncertainty and sensitivity analysis techniques for computer models. *Risk Analysis*, 8(1):71–90, 1988.

[10] 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.

[11] M. Jain, M. Tambe, and C. Kiekintveld. Quality-bounded solutions for finite bayesian stackelberg games: Scaling up. In *International Conference on Autonomous Agents and Multiagent Systems*, 2011.

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

[13] C. Kiekintveld, J. Marecki, and M. Tambe. Approximation methods for infinite bayesian stackelberg games: Modeling distributional payoff uncertainty. In *International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[14] 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.

[15] 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.

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

[17] J. Pita, M. Jain, F. OrdÃşÃśez, M. Tambe, S. Kraus, and R. Magori-Cohen. Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *The Eighth International Conference on Autonomous Agents and Multiagent Systems*, 2009.

[18] C. Seshadhri, T. G. Kolda, and A. Pinar. Community structure and scale-free collections of erdös-rényi graphs. *Phys. Rev. E*, 85:056109, May 2012.

[19] 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.

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

[21] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, June 1998.

[22] Z. Yin and M. Tambe. A unified method for handling discrete and continuous uncertainty in bayesian stackelberg games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.