Computational Economics (CS 396) Final Project Report

Brian Gauch
brian.gauch@vanderbilt.edu

ABSTRACT

Game theoretic approaches, particularly Stackelberg games, have been widely deployed in recent years to allocate security resources. Most existing security game models assume that a security resource assigned to a target can only protect that target. However, in many important real-world security scenarios, when a resource is assigned to a target it exhibits protection externalities; that is, it simultaneously provides protection for nearby targets. We build off of the formulation in Gan et al. [Gan] that incorporates protection externalities within a Stackelberg game. They made no assumptions about the topology of the space in which the targets and defense resources are located. However, this meant that resources were constrained so that they could only be located on a target. We note that in many cases security problems are fundamentally planar and develop a new model that incorporates this assumption. By constraining the problem to a planar space, we are able to consider a more flexible set of locations for resources, i.e., we remove the restriction that resources must be located at targets. We then evaluate the effectiveness of our model in comparison to Gan's model in terms of overall defender solution quality, and find that it is significantly more effective in many cases.

1. INTRODUCTION

1.1 Motivation and Previous Work

Game theory is currently being applied to model security for airports, ports, shipping and other vulnerable infrastructure. Much of the research has focused on Stackelberg games, games in which the attacker knows the defender's strategy when choosing their own through *a priori* surveillance [Korzhyk 11]. The goal of the game, from the defender's perspective, is to pick a strategy, i.e., a probability distribution over resource allocations that maximizes their utility. An assumption in most existing security game models is that a security resource assigned to a target protects only that target. However, in many real-world security scenarios, when a resource is assigned to a target, it exhibits protection externalities; that is, it also protects other "neighboring" targets.

One formulation that includes protection externalities is described in Gan et al. [Gan]. Their model makes no assumptions about the topology of the space in which the targets and resources are placed, however they do include the restriction that a resource must be allocated "at" exactly one target in some sense. This restriction is undesirable because there are many scenarios in which one might want to place resources at locations that are not co-incident with a target. Consider the situation in which there are more targets than resources; one might want to place the resources between the targets so that, if the targets are close together, a single resource could cover more than one target.

To do this, we need to specify the metric space in which the resources and targets lie. In many of the applications where externalities exist in the allocation of defense resources, one can imagine modeling the problem with targets on a plane and defense resources defending a disc on the plane. When a defense resource is a camera, radar, or mobile unit, the region that it defends is roughly a disc. Accordingly, we choose for our metric space a plane. Admittedly, this model is less suited to urban areas where line of sight is important. This model is also unsuited to cyber security or the defense of multi-story buildings, because the planar assumption is likely to be violated.

2. MODEL

Gan et al. describe a model for allocating resources to targets that incorporates protection externalities. We use this model as our inspiration and it is the model to which our approach will be compared. It is described more fully in Section 2.1. In Section 2.2, we describe our modifications to their model and discuss the pros and cons of our approach.

2.1 Gan's Model of Externalities

In [Gan], Gan et al. describe a Stackelberg game-based security model in which the defender allocates resources to a set of targets given that the resource has the externality of defending other neighboring targets.

The defender has d resources and a set of t targets to defend. The attacker's available actions (who can only attack one target) are exactly the set t. The set of actions available to the defender varies based on the experiment, but always corresponds to points on the plane so that the set of defended targets can be determined.

These actions are used to generate a game where the payoff for each pair of actions depends on the value of the target and whether or not the attacker got caught (the defender defended the target that the attacker attacked). More generally, the defender could have different values for targets that the attacker (the game could not be zero-sum).

Their algorithm assumes the availability of an adjacency matrix A indicating target adjacencies. A pure strategy for the defender corresponded to an allocation of integer resources/defenders to targets, such that each target was either defended or not defended (1 or 0). The authors acknowledged the limitation of only locating resources at targets and not between targets, but noted that a dummy target with no value to the attacker or defender could be placed at such locations. The set of actions available to the defender is the set of subsets of t of size d.

2.2 Planar Model

2.2.1 Model Definition

Previously, the set of actions available to the defender was the set of subsets of *t* of size *d*. Now, we would like to somehow allow resources to be placed not on targets. So, instead of taking in as input an adjacency matrix for targets, we need to define a metric space so that we can determine distances, and therefore adjacencies, ourselves. Although it would be natural to consider defending a three-dimensional space, for simplicity, we choose for our metric space a plane. In summary, we are adding a new feature to Gan's model, off-target resource placement, but at the cost of introducing a new restriction, i.e., requiring that all defenders and resources inhabit a two-dimensional world.

Specifically, we allow targets and resources to be placed anywhere within our "world", a rectangular region in $\mathbb{R}^2$. For convenience, we will consider the world to be bounded by [0,0] and [1,1]. Because there are an infinite number of points in this world, we need some sort of way to break it up into a finite number of regions such that all points within a region are equivalent for our purposes.

When a defense resource is a camera, radar, or mobile unit, the region that it defends is roughly a disc. Therefore, the most natural model is for a defense resource to defend a unit disk.

2.2.2 Unit Disks

Let each (uniform) resource have effective radius **r**. Let us notate the set of targets within distance **r** of a point p as T_p. Then we can say that two points p_1, p_2 are equivalent as potential locations for a defense resource if $T_{p1} = T_{p2}$.

Recall that, if the defender has **d** resources, defender actions correspond to subsets of size **d** of some set of potential resource locations **P**. In Gan's model, **P** is the set of target locations. It is natural for us to attempt to find the optimal **P** in the sense that it allows for best solution of the resulting game. Consider the set **S** of points created by selecting an arbitrary point from each region of equivalence. It is easy to see that for any strategy possible where $\mathbf{P} = \mathbb{R}^2$, there is an equivalent strategy (in the sense of having the same payoffs) when **P** = **S**.

This is actually a stronger relation than we need; we only need for the *best* strategy available when $\mathbf{P} = \mathbb{R}^2$ to be available when **P** = **S.** For this reason we introduce the notion of a non-dominated point/region. We say that a point p_1 is non-dominated if $\neg\exists\ p_2$ such that $T_{p1} \subset T_{p2}$. We similarly say that a region is non-dominated if this property holds for all points within the region. I claim that the best strategy available when $\mathbf{P} = \mathbb{R}^2$ is also available when **P** = **D** where **D** is the subset of **S** that is non-dominated.

Now the question is how to find **D** and how big it is. We start by observing that finding T_p for a point p can alternatively be thought of as finding the set of disks of radius **r**, each centered on a target, that intersect at point p. See figure 1 for a visualization of this representation. Our problem of finding equivalence regions on the plane now resembles the problem of finding cliques in a unit disk graph, where disks are centered on targets. Our problem of finding **D** would seem to correspond to the problem of finding all maximal cliques in the unit disk graph. Figure 2 is one natural example where this seems to be true.

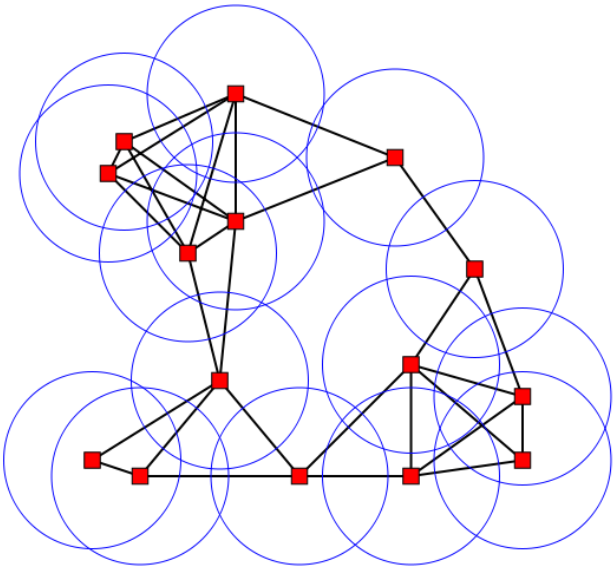

Figure 1: The intersection model of unit disks on a plane, where the center of each disk is a target, is a natural model for our application. This class of graphs is known as unit disk graphs.

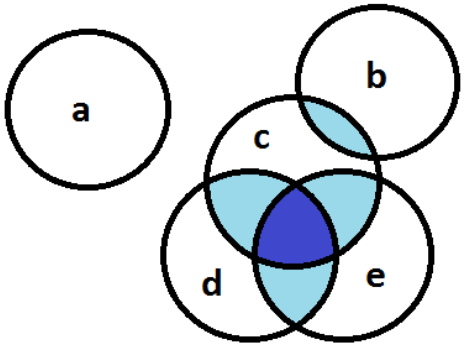

Figure 2: A natural example where **D** = the set of maximal cliques in the corresponding unit disk graph, {{a}, {b,c}, {c,d,e}}.

However, it turns out that the problems are not identical. A graph class defined by a geometric intersection model is said to have the Helly property if for every clique *C*, there is some single point *p* such that every vertex of *C* includes the point *p*. Clearly, to map from a maximal clique back to nonempty region, we need for our class of graphs, unit disk graphs, to have the Helly property; it turns out that unit disk graphs do not have the Helly property.

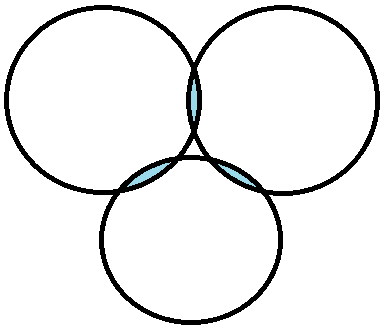

Figure 3: Counterexample of the Helly property for unit disk graphs. The three disks form a clique of size 3, but there is no single point that lies within all 3 disks.

2.2.3 Boxicity 2 Graphs

Not only does the Helly property not hold for unit disk graphs, as demonstrated in figure 3, but unit disk graphs can have an exponential number of maximal cliques [Gupta]. It is worth reconsidering our model of resources defending targets within a unit disk at this point. If we instead model a defense resource as defending all targets within some manhattan distance, finding **D** would correspond with finding all maximal cliques in a boxicity 2 graph. Figure 4 is an example of a boxicity 2 graph. Boxicity 2 graphs have the Helly property and only have $O(n^2)$ maximal cliques, where n in our case is the number of targets. Because there are a polynomial number of maximal cliques, they can all be found in polynomial time [Tsukiyama].

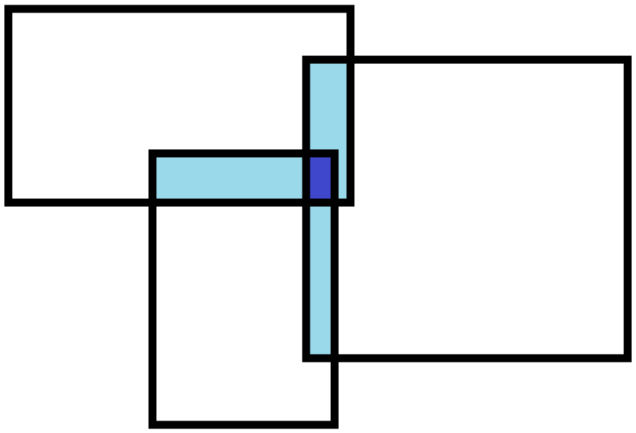

Figure 4: A boxicity 2 graph is the intersection graph of axis-aligned reactangles.

Boxicity 2 graphs can be considered to be a polynomial approximation of the more natural unit disk intersection model in the following sense. First construct the unit disk graph. Then construct a boxicity 2 graph such that each rectangle is inscribed in the corresponding disk. Two such rectangles intersect only if the disks they are inscribed in also intersect, as can be seen in figure 5. Using this fact and the Helly property, we can see that for such a boxicity 2 graph the set of its maximal cliques is a subset of **D**.

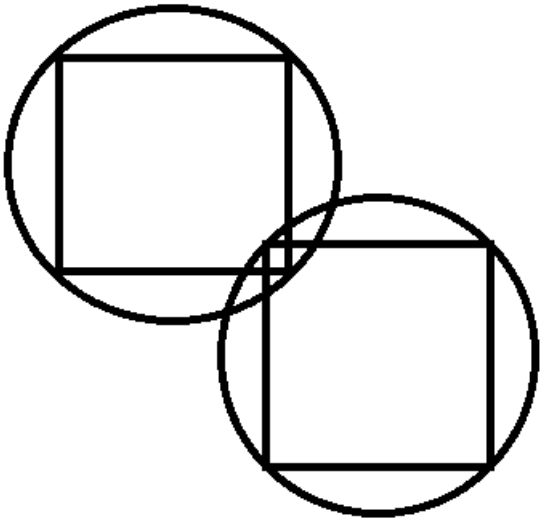

Figure 5: Two inscribed rectangles intersect only if the corresponding disks intersect.

Alternatively, if we consider the boxicity 2 graph not as an approximation of a true, disk-based model, but as the model itself (i.e., each defense resource can defend all targets within some manhattan distance), the maximal cliques of the boxicity 2 graph correspond exactly to **D**.

It is worth noting that although a unit disk graph has an exponential number of maximal cliques, because not all of these maximal cliques correspond to a region in **D**, **D** may still be of polynomial size even when a defense resource defends a unit disk. However the authors of this report are unaware of a method for enumerating the cliques that *do* obey the Helly property, or even counting them.

3. IMPLEMENTATION

Our model is implemented in C++ on a linux platform. It is comprised of four major modules, namely, the *world builder*, the *resource location selector*, the *game initializer*, and the *game solver*. Each of these is described in more detail below.

3.1 World Builder

The *world builder* is given the size of the world and the number of targets. It produces a mapping of targets to locations. Targets are placed uniformly at random locations.

3.2 Resource Location Selector

The *resource location selector* is given the target placements, the size of the world, the number of targets, and which model to use. It produces a set of resource locations to be considered during the game proper. It implements Gan's resource placement model and our planar, maximal clique algorithm. The resource location algorithm used is controlled by a flag for experimental purposes.

We implemented a simple maximal clique finding algorithm that runs in exponential time. It does not exploit any properties of boxicity 2 graphs except for the bounded number of maximal cliques. It simply keeps track of candidate cliques and keeps growing each, possibly creating several candidates of size k+1 from a candidate of size k. Initial candidates are the set of

vertices/boxes. A candidate is added to the maximal clique list when it can no longer be grown while still being a clique. Each candidate and its descendent candidates only need attempt to add each vertex once, because if it could not be included into the clique earlier in the process, it cannot be included later. Candidates are stored in a set so that duplicates of a maximal clique cannot be included by building them in a different order. To enforce the set property, it takes O(log(n)) time to insert a candidate. The algorithm is exponential because any number of candidates can "dead end" by attempting to insert themselves into the maximal clique set and finding that they are a duplicate.

3.3 Game Initializer

The *game initializer* is given as input the target placements, and the possible resource placements. It produces a normal form game utility matrix for use by the game solver. It has three main tasks:

1) It enumerates attacker actions that are simply the set of targets

2) It enumerates defender actions by selecting subsets of selected resource placements;

3) It then calculates the utility of each action pair by determining whether or not the attacker attacked a defended target, and using the value of the attacked target.

It would be simple enough to also incorporate target values, but we have no reason to believe that the distribution of target values would affect the relative effectiveness of our model vs. Gan's model. So, we simply treat all targets as having uniform value.

3.4 Game Solver

The *game solver* is given a game in normal form utility matrix form, and outputs the expected utility of the defender.

We used a linear program for solving zero sum games, which was implemented in homework 3 of Vanderbilt's spring 2015 offering of CS396: "Computational Economics".

3.5 Testing Framework

In addition to the modules there is a testing framework for their combination and analysis. A shell script that loops over all parameter values in the specified ranges, loops for the specified number of trials (100 in our case), calls the other modules in order with those parameters, and outputs the final resulting defender utility of each trial to a csv file for later analysis.

4. RESULTS

4.1 Experimental Design

We will compare three models:

1) A simple security game with no protection externalities

2) Gan's model

3) Our planar, maximal-clique model

Model (1) can be viewed as a special case of (2) or (3) in which resource range = 0. Because we assign targets uniform value for the attacker and defender, the solution to (1) will always be to distribute defense resources with uniform probability among targets.

We will often refer to our model as the Planar model. We will sometimes refer to model (2) as the Non-Planar model, but recall that in (2) the defender benefits from the range of its resources and is merely naïve with respect to the underlying planar model.

Model (1) will be treated as a baseline for (2) and (3), so that we can see how much utility the defender gains from the existence of planar externalities (and awareness of externalities), and how much the defender gains from awareness of the planar space. If the expected utility of (3) – (1) is significantly higher than that of (2) – (1), then the space is worth modeling when it is planar and proximity-based externalities exist.

4.2 Results

We explored the parameter space up to #targets=10, #resources=5, and range=0.3 in 0.1 increments. For each combination of parameters, we played 100 games with both models, each time redistributing targets uniformly at random from [0,0] to [1,1]. There were 120 combinations of parameters tested in all. For all combinations with range>=0.1 and #targets >=4, there was a significant increase in defender utility when using the Planar model.

Below, we plot a representative subset of the results in figures 6 through 11. "Possible gain" is the maximum possible defender utility (associated with defending all targets) minus model (1)'s expected utility. "Utility gain" is the difference in expected utilities of (3) - (1) and (2) - (1) respectively.

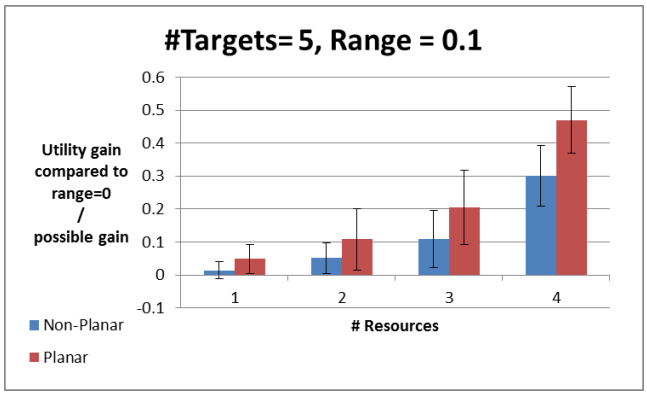

Figure 6: With 5 targets and range 0.1, we compared the mean utility gain across 100 trials for the non-planar and planar models for 1 through 4 resources. Using a two-tailed student t-test, we found that the planar model significantly outperformed the non-planar version for all test scenarios ($p < 0.05$).

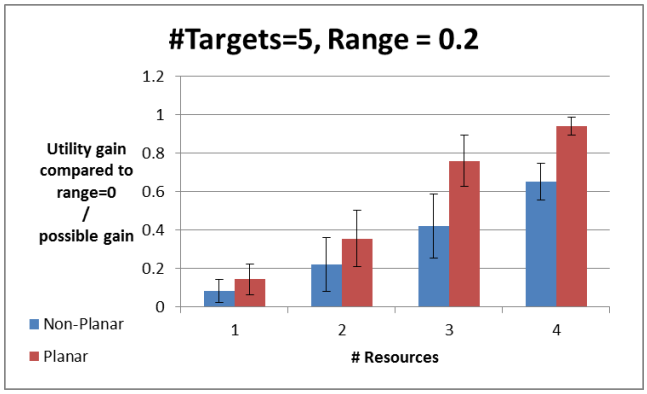

Figure 7: With 5 targets and range 0.2, we compared the mean utility gain across 100 trials for the non-planar and planar models with 1 through 4 resources. Using a two-tailed student t-test, we found that the planar model significantly outperformed the non-planar version for all test scenarios ($p < 0.05$).

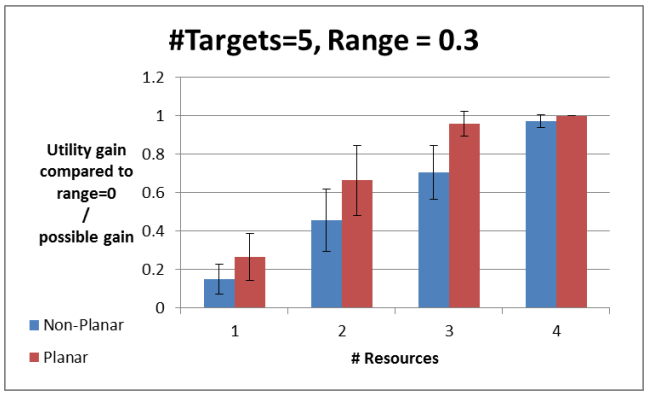

Figure 8: With 5 targets and range 0.3, we compared the mean utility gain across 100 trials for the non-planar and planar models with 1 through 4 resources. Using a two-tailed student t-test, we found that the planar model significantly outperformed the non-planar version for all test scenarios ($p < 0.05$).

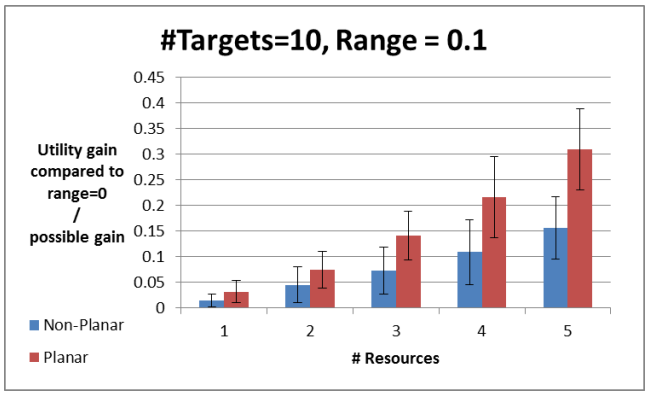

Figure 9: With 10 targets and range 0.1, we compared the mean utility gain across 100 trials for the non-planar and planar models with 1 through 5 resources. Using a two-tailed student t-test, we found that the planar model significantly outperformed the non-planar version for all test scenarios ($p < 0.05$).

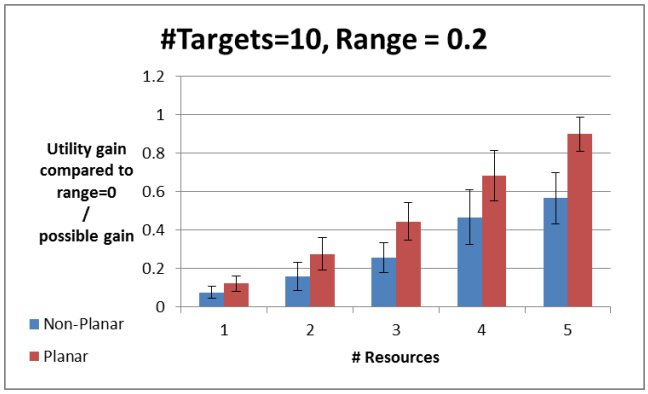

Figure 10: For 5 targets and range 0.1, we compared the mean utility gain for the non-planar and planar models with 1 through 5 resources. Using a two-tailed student t-test, we found that the planar model significantly outperformed the non-planar version for all test scenarios ($p < 0.05$).

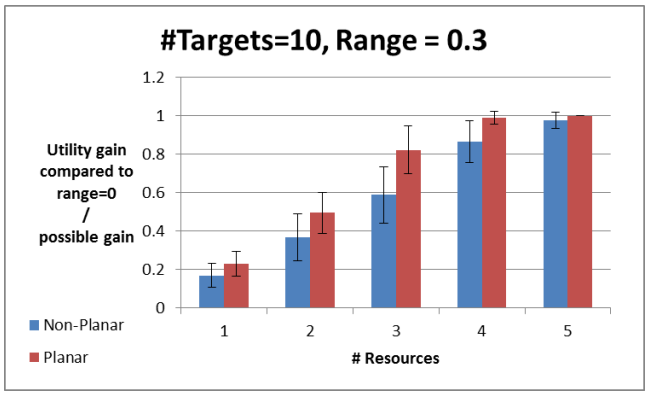

Figure 11: With 10 targets and range 0.3, we compared the mean utility gain across 100 trials for the non-planar and planar models with 1 through 5 resources. Using a two-tailed student t-test, we found that the planar model significantly outperformed the non-planar version for all test scenarios ($p < 0.05$).

Overall, our planar model reaps significantly more benefits from proximity-based externalities than Gan's model, assuming that the space is in fact planar.

In trivial scenarios, where the number of defense resources is no less than the number of targets, all three models naturally perform the same. Unsuprisingly, in relatively easy scenarios, where the number of resources is high and resources have long range, Gan's model perfoms almost as well as ours; Clearly, as range approaches world size, Gan's model becomes equivalent to ours.

It seems that in harder defense scenarios, where the number of resources is small compared to the number of targers, and where the range of each resource is small compared to the world size, our model gets ~1.5-2 times the benefit from externalites as Gan's model.

5. RELATED WORK

Game theory in its present form was originally presented by von Neumann in his 1928 paper. Although most closely tied to economics, almost since its inception game theory has been applied to security problems, most famously during the 1950s to model global nuclear strategy. Also during the 1950s, Nash famously developed a criterion for mutual consistency of players' strategies in non-cooperative games, known as the Nash equilibrium [Wiki]. This sparked great interest in the field as many research groups extended the models, defining for example repeated and extensive form games, and applied them to new areas.

More recently, thanks to increased computational power and new, efficient algorithms, it has become practical to use Stackelberg games to model real-world security problems. Because it seems natural to assume that the attacker conducts surveillance, Stackelberg games became more popular for such problems than models in which both players act simultaneously. One of the key revitalizing papers was Brown et al., 2006 [Brown 06] in which they argue that critical infrastructure defense must become more sophisticated to be adequately protect against terrorist attacks. They advocate the use of optimization models, in particular, the Stackelberg model, as a way to model attackers and defenders to develop a robust protection system. Real world applications include the ARMOR security system deployed at the Los Angeles International Airport [Pita, 2008] and the IRIS system deployed by the Federal Air Marshals Service to allocate marshals to tours of duty to protect commercial flights [Jain, 2010].

Gan et al. [Gan] term games in which one defense resource can defend multiple nearby targets Security Games with Protection Externalities (SPEs). They use the term externality because each resource is assigned to one primary target, and may defend other nearby targets, as a bonus. They show that finding Stackelberg equilibria in such games is NP-hard. Nevertheless, they show that polynomial approximations perform better than ignorant solutions when proximity-based protection externalities exist.

Fang et al. [Fang 2013] and Xu et al. [Xu 2014] did not call the effect an externality, but studied patrol paths on the real number line with moving targets, where a defense resource had an effective range. Because a defense resource was not bound to a particular target, but was instead constrained to a particular metric space, this work is in some senses more similar to our own than Gan's.

6. CONCLUSIONS

6.1 Summary

We are interested in game theoretical approaches to the allocation of resources to the defense targets – so-called security games. We were inspired by Gan's model of security games, which incorporates protection externalities. We introduce our planar, maximal clique model for allocating defender resources to protect targets. Our version extends the protection externalities described by Gan to allow for resource locations other than at target locations. To solve this problem we have defined a metric space for the game world, in our case, a plane, so that we can say that a resource defends a target if it is sufficiently close. Although the most natural model is for a resource to defend a disk around itself, we compromised by having a resource defend an axis-aligned rectangle around itself. This allows us to find non-dominated resource locations quickly, since these correspond to maximal cliques in a boxicity 2 graph where each target is an axis-aligned rectangle.

We implemented Gan's version and ours and evaluated the two approaches using the expected utility gains over the externality-free model. We tested both models over a range of scenarios, varying the numbers of targets and resources from 1 to 10 and from 1 to 5, respectively. Effective resource range was varied from 0.1 to 0.3 in increments of 0.1, with all targets placed uniformly at random between [0,0] and [1,1]. For each of the 120 resulting scenarios, utility was calculated using a zero sum Nash equilibrium solver.

We found that a knowledge of the planar space results in significant improvements to expected defender solution quality in all non-trivial scenarios where the number of targets >=4 and range >= 0.1.

6.2 Future Work

There is a fundamental tradeoff between the number of actions considered and the time to solve the resulting game. Because the solution to the SPE corresponds to a set covering, each target should be defended by at least one action to allow for a good solution. In this project, we simply considered all $O(n^2)$ maximal cliques, but more practically a O(n) subset should be considered so that the resulting game takes no longer to solve.

What makes a selection of maximal cliques good? That is, what actions should be added first when attempting to allow for good solutions to the game? There are a number of natural heuristics one might consider. One could greedily add the largest maximal cliques to the set of actions. One could greedily add the action that protects the most as-yet-unprotectable targets. One could imagine more sophisticated algorithms searching for a set of actions with backtracking. This may merit significant further work.

Although the exponential running time of our boxicity 2 maximal clique finding algorithm itself did not matter for the purposes of our tests, since the game was of exponential size, it may matter when used with a polynomial, approximate game solving algorithm such as CLASPE. For this reason, it may be worthwhile to develop faster boxicity 2 maximal clique finding algorithms. Using the general result from [Tsukiyama], we see that there exists a $O(n^3m)$ algorithm. We suspect that a $O(n^2\log(n))$ algorithm exists that uses a sweep line. Since a boxicity 2 graph may have $O(n^2)$ maximal cliques, $O(n^2)$ is the fastest we could hope for.

Polynomial algorithms may even exist to find all maximal cliques in a unit disk graph that obey the Helly property; that is, it may be possible to find not just a subset of **D** but all of **D** - this would allow for truly optimal solutions in the sense that the expected utility of the defender in the solved, resultant game would be maximized. If such an algorithm existed, the authors of this report suspect that it would fall in the area of computational geometry, not graph theory.

Finally, metric spaces other than a plane could be considered.

7. REFERENCES

[1] Jiarui Gan, Bo An, and Yevgeniy Vorobeychik. Security games with protection externalities. (AAAI, 2015, to appear).

[2] Gupta, Rajarshi, Jean Walrand, and Olivier Goldschmidt. "Maximal cliques in unit disk graphs: Polynomial approximation." Proceedings INOC. Vol. 2005. 2005.

[3] Tsukiyama, Shuji, et al. "A new algorithm for generating all the maximal independent sets." SIAM Journal on Computing 6.3 (1977): 505-517.

[4] Fang, F.; Jiang, A. X.; and Tambe, M. 2013. Optimal patrol strategy for protecting moving targets with multiple mobile resources. In Proceedings of the 12th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'13), 957–964.

[5] Brown, Gerald, et al. "Defending critical infrastructure." Interfaces 36.6 (2006): 530-544.

[6] Xu, H.; Fang, F.; Jiang, A. X.; Conitzer, V.; Dughmi, S.; and Tambe, M. 2014. Solving zero-sum security games in discretized spatio-temporal domains. In Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14).

[7] Pita, James, et al. "Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport." Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[8] Jain, Manish, et al. "Security Games with Arbitrary Schedules: A Branch and Price Approach." AAAI. 2010.

[9] Korzhyk, Dmytro, Vincent Conitzer, and Ronald Parr. "Complexity of Computing Optimal Stackelberg Strategies in Security Resource Allocation Games." AAAI. 2010.

[10] Spinrad, Jeremy P. Efficient graph representations. American mathematical society, 2003.

[11] Wikipedia contributors. "Game theory." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 1 Mar. 2015. Web. 4 Mar. 2015.

[12] Wikipedia contributors. "Clique (graph theory)." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 6 Apr. 2015. Web. 6 Apr. 2015.

[13] Wikipedia contributors. "Unit disk graph." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 24 Aug. 2014. Web. 6 Apr. 2015.

[14] Wikipedia contributors. "Boxicity." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 22 Mar. 2015. Web. 6 Apr. 2015.