Modeling Human Bounded Rationality to Improve Defender Strategies in Network Security Games

Rong Yang, Fei Fang, Albert Xin Jiang, Karthik Rajagopal,
Milind Tambe, and Rajiv Maheswaran

University of Southern California

Abstract. In a Network Security Game (NSG), security agencies must allocate limited resources to protect targets embedded in a network, such as important buildings in a city road network. A recent line of work relaxed the perfect-rationality assumption of human adversary and showed significant advantages of incorporating the bounded rationality adversary models in non-networked security domains. Given that real-world NSG are often extremely complex and hence very difficult for humans to solve, it is critical that we address human bounded rationality when designing defender strategies. To that end, the key contributions of this paper include: (i) comprehensive experiments with human subjects using a web-based game that we designed to simulate NSGs; (ii) new behavioral models of human adversary in NSGs, which we train with the data collected from human experiments; (iii) new algorithms for computing the defender optimal strategy against the new models.

Keywords: Bounded Rationality, Network Stackelberg Games, Decision-making, Quantal Response

1 Introduction

In the last few years there has been an increasing interest in using game theory to build decision-support tools for real-world security problems, in which security forces must allocate resources (such as patrol cars and canine units) to protect one or more potential targets that attackers (e.g. terrorists) would like to damage or destroy. Since in these domains the attacker can usually observe the defender's strategy before deciding on a plan of attack, it is natural to model this as a Stackelberg game where the leader (the defender) commits to a randomized strategy before the follower (the attacker) chooses a strategy. Such attacker-defender Stackelberg game models have been used as the basis of decision-support systems that have been successfully deployed in real-world domains, including ARMOR at the LAX airport [1] and IRIS at the Federal Air Marshals Service [2].

In this paper we focus on security games whose domains have structure that is naturally modeled as graphs. For example, in response to the devastating terrorist attacks in 2008 [3], Mumbai police deployed randomized checkpoints as one countermeasure to prevent future attacks [4]. This can be modeled as a Stackelberg game on a graph with intersections as nodes and roads as edges, where certain nodes are targets for attacks.

The attacker chooses a path on the graph ending at one of the targets. The defender can schedule checkpoints on edges to try to catch the attacker before a target is reached. Such *graph-based* Stackelberg security games have received recent study [5,6,7].

A common assumption of these previous studies is that the attacker is perfectly rational (i.e. chooses a strategy that maximizes their expected utility). This is a reasonable proxy for the worst case of a highly intelligent attacker, but it can lead to a defense strategy that is not robust against attackers using different decision procedures, and it fails to exploit known weaknesses in the decision-making of human attackers. Indeed, extensive experimental studies have shown that standard game-theoretic assumptions of perfect rationality are not ideal for predicting the behavior of humans in multi-agent decision problems, and various alternative models have been proposed [8,9,10,11]. Recently, Yang *et al* [12] studied human behavior models of attackers in the setting of (non-networked) Stackelberg security games, and showed that defender strategies based on a quantal response model (an adaptation of McKelvey and Palfrey's Quantal Response Equilibrium (QRE) concept [10] to the Stackelberg setting) achieved promising performance when tested against human subjects, outperforming previous methods for security games [13,14] as well as a behavior model based on Prospect Theory [8].

In this work, we initiate the study of human behavior models of adversaries in graph-based security games. Compared to the non-networked domains, the network structure of this domain further complicates the decision process of the human adversaries, further motivating the need to relax assumptions of perfect rationality. Specifically, the attacker must choose a path in the graph where each edge is covered by the defender with some observed probability, and thus must reason about sequences of random events. Our goal is to explore any bias and/or heuristic behavior exhibited by human adversaries when facing such decision problems, and to design defender strategies that exploit such behavior.[1] While it is generally accepted that humans tend to rely on heuristics when faced with complex problems (e.g., [15]), we are not aware of any existing studies that specifically addressed heuristic human behavior when faced with the kinds of problems in our domain. Burgess and Darken [16] proposed a fluid-simulation-based model for human path planning in continuous terrains. However, this model is less applicable to our domain in which the choices are discrete. Instead, we considered two behavior models for attackers in network security games. First, we adapted the quantal response model of [12] to network security games. For the second model (which we call quantal response with heuristics), the attacker's behavior now depends on the values of several easy-to-compute *features* of the attacker's decision problem. The rationale is that the attacker could be using some of these features as the basis of a heuristic decision procedure. In order to train our models and to evaluate their performances, we developed a web-based game that simulates the decision tasks faced by the attacker, posted the game on Amazon Mechanical Turk and collected data on how humans played the game. We

[1] We note that another source of complexity for human adversaries is the fact that the number of possible paths grow exponentially in the size of the graph. Thus for large graphs it becomes impractical for human adversaries to enumerate all paths and thus they must rely on other algorithms or heuristics. In this paper we instead focus on small graphs in which it is relatively easy to enumerate all paths, in order to isolate and study the effect of human reasoning of sequential random events. We plan to extend our setting to large graphs in future work.

trained our models using maximum-likelihood estimation given data from an initial set of experiments. We then computed defender strategies that optimize defender utility against each of these behavior models of the attackers, and compared the performance of these strategies in a subsequent set of experiments on Amazon Mechanical Turk. Overall, our models learned from data significantly outperformed the rational model as well as two other baseline models.

2 Problem Statement

We model a network security domain, similar to that introduced by Tsai *et al* [5]. We use the following notation to describe the game, which are also listed in Table 1. The game is played on a graph $G = (V, E)$. The attacker starts at one of the source nodes $s \in S \subset V$ and travels along a path chosen by him to get to one of the target nodes $t \in T \subset V$. The attacker's set of pure strategies $\mathcal{A}$ then consists of all the possible paths from some $s \in S$ to some $t \in T$, which we denote $A_1, \ldots, A_{|\mathcal{A}|} \subset E$. Meanwhile, the defender tries to catch the attacker by setting up check points on the passing edges before the attacker reaches the target. Let M be the total number of security resources, meaning the defender could then set up at most M simultaneous check points in the network. Thus the set of defender's pure strategies $\mathcal{D}$ consists of all subsets of E with at most M elements, which we denote $D_1, \ldots, D_{|\mathcal{D}|}$. If the attacker chooses a path which has at least one edge covered by the defender, then the attacker gets caught and receives a penalty, and the defender receives a reward for catching the attacker; otherwise, the attacker receives a reward for successfully attacking the target and the defender receives a penalty. Formally, assuming the defender plays an allocation D_i, and the attacker chooses a path A_j, the attacker succeeds if and only if $D_i \cap A_j = \emptyset$.

The game was assumed to be zero-sum in earlier work [5,6]. In this paper, we relax this assumption to consider a more general class of games. We use R_j^a to denote the rewards received by attacker for a successful attack through path A_j, and P_j^d to denote the penalty received by the defender. If the attacker gets caught on path A_j, we denote his penalty by P_j^a and the reward received by the defender by R_j^d. In this paper, we make no other assumptions on the payoff of the game except for the natural assumption that $R_i^a > P_i^a$ and $R_i^d > P_i^d, \forall i \in \{1, \ldots, |\mathcal{A}|\}$. In other words, adding more resources to protect a path would benefit the defender and harm the attacker. Taking everything together, we define a *network security game* Γ as the tuple:

$$(G, S, T, M, \{R_i^d\}, \{P_i^d\}, \{R_i^a\}, \{P_i^a\})$$

The attacker conducts surveillance to learn about the defender's strategy, so it is important for the defender to randomize her strategy to avoid exploitable patterns. In other words, the defender has to commit to a distribution over her pure strategies. We use x_e to denote the probability that an edge $e \in E$ will be covered by the defender and $\boldsymbol{x} = \langle x_e, \forall e \in E \rangle$ to denote the vector of marginal probabilities of covering each of the edges in the graph. In general, if the attacker chooses path A_i, the probability that he will be captured (denoted p_i) is the probability that at least one edge on the path A_i is covered by the defender. Tsai *et al* [5] showed that given $\boldsymbol{x}$, the sum of marginals on the edges of the path $\sum_{e \in A_i} x_e$ is an upper bound of p_i, and this upper bound

Table 1. Notations used in this paper

$G = (V, E)$	Network game graph
M	Total number of defender resources
$\mathcal{A}$	Set of attacker paths, $\mathcal{A} = \{A_i\}$
A_i	i^{th} attacker path
R_i^a	Reward for attacker for a successful attack through path A_i
P_i^a	Penalty for attacker if he gets caught on path A_i
R_i^d	Reward for defender for catching attacker on path A_i
P_i^d	Penalty for defender for a successful attack through path A_i
$\mathcal{D}$	Set of defender allocations (strategies), $\mathcal{D} = \{D_j\}$
D_j	j^{th} defender allocation
Γ	A network security game
x_e	Probability that edge e will be covered by a resource

can be reached if the defender can ensure that in each pure strategy D_j played with positive probability, only one edge on the path A_i is covered. Tsai *et al* [5] proposed algorithms that sample defender pure strategies from $\boldsymbol{x}$, however such techniques are not guaranteed to reach this upper bound in all cases. In this paper, we make the simplifying assumption that the total amount of defender resources M is equal to 1 for the following two reasons: 1) with only one resource, the upper bound on covering path A_j is reached; 2) we are focusing on small graphs. In the future, we plan to relax the restriction of only one resource. Then since at most one edge of the graph will be covered in any pure strategy D_j, we have $p_i \equiv p_i(\boldsymbol{x}) = \sum_{e \in A_i} x_e$ for all i. Then we can write the expected utility of the defender if attacker chooses path A_i as

$$U_i^d(\boldsymbol{x}; \Gamma) = p_i(\boldsymbol{x}) R_i^d + (1 - p_i(\boldsymbol{x})) P_i^d \tag{1}$$

and the expected utility for the attacker if he chooses A_i as

$$U_i^a(\boldsymbol{x}; \Gamma) = (1 - p_i(\boldsymbol{x})) R_i^a + p_i(\boldsymbol{x}) P_i^a \tag{2}$$

Let $q_i(\boldsymbol{x}; \Gamma)$ denote the probability that attacker chooses path A_i, given the defender's marginal coverage on all the edges $\boldsymbol{x}$. The optimal strategy for the defender is to maximize the average expected utility:

$$\max_{\boldsymbol{x}} \sum_{A_i \in \mathcal{A}} q_i(\boldsymbol{x}; \Gamma) U_i^d(\boldsymbol{x}; \Gamma) \tag{3}$$

It is thus important for the defender to accurately model the attacker's response to her strategy, i.e., $q_i(\boldsymbol{x}; \Gamma)$ for all i.

We assume that the attacker can observe M (which is equal to 1), as well as the the defender's marginal coverage on all the edges $\boldsymbol{x}$. A fully rational attacker would be able to deduce that $p_i = \sum_{e \in A_i} x_e$ for all i and choose a path that maximizes his expected utility: $i^* = \arg\max_i U_i^a(\boldsymbol{x}; \Gamma)$. However in real-world security problems, we are facing human attackers who may not respond optimally. The goal of this paper is to explore models that can better predict the behavior of human attackers.

3 Adversary Models

In this section, we propose several models of how a human attacker responds to the defender's strategy.

3.1 Basic Quantal Response Model

In our first model the attacker's mixed strategy is a quantal response (QR) to the defender's strategy. Under this QR model, given a graph game Γ and a defender's strategy $\boldsymbol{x}$, the probability that the adversary is going to choose path A_i is

$$\texttt{QR}: \quad q_i(\lambda \mid \boldsymbol{x}; \Gamma) = \frac{e^{\lambda U_i^a(\boldsymbol{x};\Gamma)}}{\sum_{A_k \in \mathcal{A}} e^{\lambda U_k^a(\boldsymbol{x};\Gamma)}} \tag{4}$$

where $\lambda > 0$ is the parameter of the quantal response model [10], which represents the error level of adversary's quantal response. When $\lambda = 0$, the adversary chooses each path with equal probability; when $\lambda = \infty$, the adversary becomes fully rational and only selects the paths which give him the maximum expected utility. It is shown in many empirical studies that λ usually takes a positive finite value. A version of the QR model for non-networked security domains was studied in [12].

3.2 Quantal Response with Heuristics

In a network security game Γ, in order to evaluate the expected utility of a path A_i, $U_i^a(\boldsymbol{x}; \Gamma)$, the attacker has to compute p_i, which requires reasoning about a sequence of random events, i.e., whether or not each edge on the path will be covered by the defender. Even in our simplified games in which $M = 1$ and thus a perfectly-rational attacker can compute p_i as the sum $\sum_{e \in A_i} x_e$, computing this probability can be more difficult for bounded-rational human attackers who might not know this formula. Instead, the adversary might use simple heuristics to evaluate the "utility" of each path.

We propose the following model of the attacker's behavior which we call Quantal Response with Heuristics (QRH):

$$\texttt{QRH}: \quad h_i(\mu \mid \boldsymbol{x}; \Gamma) = \frac{e^{\mu \cdot f_i(\boldsymbol{x})}}{\sum_{A_k \in \mathcal{A}} e^{\mu \cdot f_k(\boldsymbol{x})}} \tag{5}$$

where $\mu = \langle \mu_1, ..., \mu_m \rangle$ is a vector of coefficients of the model and given $\boldsymbol{x}$,

$$f_i(\boldsymbol{x}) = \langle f_{i1}(\boldsymbol{x}), .., f_{im}(\boldsymbol{x}) \rangle$$

is a vector of m *features* for path A_i that influences the attacker's decision making.

We observe that under both QR and QRH models the attacker's mixed strategy belongs to the *exponential family* of distributions widely used in statistical learning. The form of the QRH model is more general than QR: it allows linear combinations of multiple features, and furthermore $f_{ij}(\boldsymbol{x})$ can be any function, including the attacker's expected utility $U_i^a(\boldsymbol{x}; \Gamma)$ used in the QR model. On the other hand, since our focus for the QRH model is on simple heuristics, we use a set of five features that are easy to compute for humans and thus could be used as basis for heuristics. These features are listed in Table 2.

Table 2. Lists of Path Features

$f_{i1}(\boldsymbol{x}) := \sum_{e \in E} A_{ie}$	Number of edges
$f_{i2}(\boldsymbol{x}) := \max_{e \in A_i} x_e$	Minimum edge coverage
$f_{i3}(\boldsymbol{x}) := \min_{e \in A_i} x_e$	Maximum edge coverage
$f_{i4}(\boldsymbol{x}) := \sum_{e \in A_i} x_e$	Summation of edge coverage
$f_{i5}(\boldsymbol{x}) := f_{i4}(\boldsymbol{x}) / f_{i1}((\boldsymbol{x}))$	Average edge coverage

4 Learning

4.1 Data Collection

In order to estimate the values of the parameters of our models, we first need data on how humans behave when faced with the kind of decision tasks the attacker faces. We developed a web-based game which simulates the decision tasks faced by the attacker in network security games, and collected data on how human subjects play the game by posting the game as a Human Intelligent Task (HIT) on Amazon Mechanical Turk (AMT).[2]

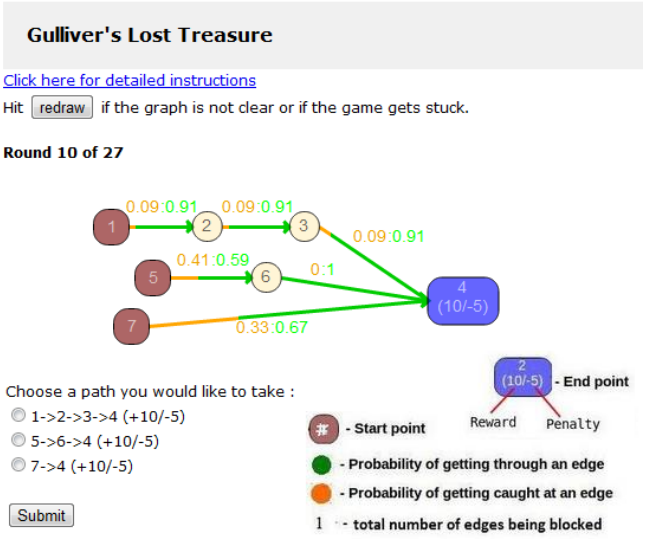
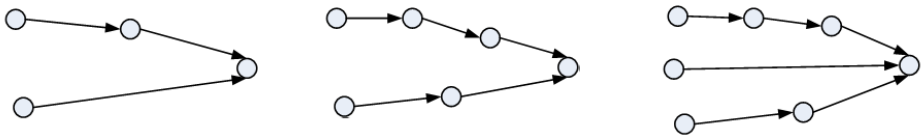


Fig. 1. Game Interface (colored)

Figure 1 displays the interface of the game. Players were introduced to the game through a series of explanatory screens describing how the game is played. In the game, the web interface presents a graph to the subjects and specifies the source(starting) nodes and the target nodes in the graph. The subjects are asked to select a path from

[2] https://www.mturk.com

one of the source nodes to one of the target nodes. They are also told that the defender is trying to catch them by setting up checkpoints on the edges. The probability that there will be a check point on each edge is given to the subjects, as well as the reward for successfully getting through the path and the penalty for being caught by the defender. Thus each instance of this game can be specified by a network security game and a defender strategy. Formally, we define a *game sample* as $g = (\Gamma, \boldsymbol{x})$, where Γ is a network security game and $\boldsymbol{x}$ is a defender strategy. Each human subject plays multiple rounds in sequence, each corresponding to a different game sample. In each game round, after a subject selects a path in the network, the edges that will be covered by the defender is sampled according to the probability shown in the figure. Subjects get a positive score if they successfully get through the path and a negative score if they select a path which has edges covered by the defender. In order to mitigate learning effects, subjects were not told of the result of each game round until they finish all game rounds. Each subject receives $\$0.5$ for participating in the experiments, and is paid $\$0.01$ bonus for each point they earn. In our experiments subjects earned $\$1.1$ bonus on average.



Graph 1 Graph 2 Graph 3

Fig. 2. Graphs Tested in Data Collection

We conducted a first set of experiments on three simple graphs, shown in Figure 2. Since the purpose of this set of experiments is to collect data to train our models, we want to use a wide variety of defender strategies. We first randomly generated 1000 different defender strategies for each graph. We then used k-means clustering to classify these random strategies into K clusters. The centers of the clusters are selected as the representative strategies and used in the experiments. We selected 10 strategies for Graph 1, 10 strategies for Graph 2 and 20 strategies for Graph 3; details on the strategies can be found at an online appendix.[3] In total, we tested 40 different game samples, each of which are played by 40 different subjects.

4.2 Training the QR Model

We first train the basic quantal response model, `QR`, using the data collected in the experiment described in Section 4.1. We use Maximum Likelihood Estimation (MLE) to tune the parameter λ. Our derivation here is similar to that of [12].

[3] http://anon-aamas2012-paper826.webs.com/

Given the choices of N subjects, with $\tau(n)$ denoting the path chosen by player n, the likelihood of λ on game sample g is

$$L_{QR}(\lambda \mid g) = \prod_{n=1}^{N} q_{\tau(n)}(\lambda \mid \boldsymbol{x}; \Gamma)$$

Then the log-likelihood of λ is

$$\log L_{QR}(\lambda \mid g) = \sum_{n=1}^{N} \log q_{\tau(n)}(\lambda \mid \boldsymbol{x}; \Gamma)$$

Let N_i be the number of subjects attacking target i. Then, we have

$$\log L_{QR}(\lambda \mid g) = \sum_{A_i \in \mathcal{A}} N_i \log q_i(\lambda \mid \boldsymbol{x}; \Gamma)$$

Combining with Equation (4),

$$\log L_{QR}(\lambda \mid g) = \lambda \sum_{A_i \in \mathcal{A}} N_i U_i^a(\boldsymbol{x}) - N \log(\sum_{A_i \in \mathcal{A}} e^{\lambda U_i^a(\boldsymbol{x})}) \tag{6}$$

We train the model by maximizing the total log-likelihood of all the 40 game samples

$$\max_{\lambda} \sum_{g \in \mathcal{S}} \log L_{QR}(\lambda \mid g) \tag{7}$$

where $\mathcal{S}$ denotes the set of all 40 game samples. It is relatively straightforward to verify that the second order derivative of $\log L_{QR}(\lambda \mid g)$ is always nonpositive. Thus $\log L_{QR}(\lambda \mid g)$ is a concave function in λ for all g. Therefore, the total log-likelihood of Equation 7 is concave and we can apply any local optimization solver (we used Matlab's `fmincon` solver). The maximum-likelihood estimate of λ based on the data is 0.34.

4.3 Training the QRH Model

In training the `QRH` model, we need to first decide which subset of the 5 features from Table 2 to use in the model, and then train the model for the selected features. Although in general the more features we select the better the fit will be, taking the set of all features can result in over-fitting. This *feature selection* problem is well-studied in statistics and machine learning, and techniques such as L1-regularized regression methods were proposed to introduce bias towards smaller sets of features. In this paper we apply a simple form of bias: we consider only subsets of features of sizes 1 and 2. We then select the top-performing subsets of size 1 and the top-performing subsets of size 2. Specifically, for each $L \in \{1, 2\}$, we do the following:

1. For each of the $\binom{5}{L}$ possible subsets of size L, we train a QRH model using this subset of features using MLE;
2. We compare the models using 2-fold cross validation, and pick the top two feature combinations.

Since we are only selecting from 5 features, we only have to evaluate a small number of models. In future work we plan to explore more sophisticated feature-selection techniques, which would allow us to select from a large set of possible features.

In order to apply 2-fold cross validation, we first randomly divided all the 40 game samples into two equal-sized sets, $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$. We conducted two rounds of training, one using $\boldsymbol{S}_1$ and the other using $\boldsymbol{S}_2$. In each round of training, the model is trained by maximizing the total log-likelihood of the game samples in the training set:

$$\max_{\mu} \sum_{g \in \mathcal{S}_{train}} \log L_{QRH}(\mu \mid g), \tag{8}$$

where $\mathcal{S}_{train} \in \{\mathcal{S}_1, \mathcal{S}_2\}$ is the training set, and $\log L_{QRH}(\mu \mid g)$ is the log-likelihood of QRH model of game sample g, derived similarly as $\log L_{QR}(\lambda \mid g)$:

$$\log L_{QRH}(\mu \mid g) = \mu \cdot (\sum_{A_i \in \mathcal{A}} N_i f_i(\boldsymbol{x})) - N \log(\sum_{A_i \in \mathcal{A}} e^{\mu \cdot f_i(\boldsymbol{x})}). \tag{9}$$

We can show that $\log L_{QRH}(\mu \mid g)$ is a concave function in μ, since the Hessian matrix is negative definite. Therefore, it can be solved use any local optimization solver.

Given a combination of the features f_i, let μ^1 and μ^2 be the training results on $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, respectively. We measure the model fit of f_i as the sum of the log-likelihoods of $\boldsymbol{S}_2$ under the model for μ^1 and $\boldsymbol{S}_1$ under the model for μ^2:

$$\texttt{Fit}(f_i) = \sum_{g \in \mathcal{S}_2} \log L_{QRH}(\mu^1 \mid g) + \sum_{g \in \mathcal{S}_2} \log L_{QRH}(\mu^2 \mid g) \tag{10}$$

Table 3. Fit ($\log L$) of model `QRH` using single feature

features	Train on $\boldsymbol{S}_1$		Train on $\boldsymbol{S}_2$		Total
	training	testing	training	testing	testing
1	-707.2	-672	-636.8	-744.4	-1416.4
2	-693.6	-666	-658	-702	-1368
3	-636	-580.4	-573.6	-642.8	***-1223.2***
4	-677.2	-723.6	-710	-690.8	-1414.4
5	-667.6	-618.4	-606.4	-680.4	***-1298.8***
U_i^a	-645.6	-689.6	-682.8	-652.4	-1342

Table 3 displays the fit results for single features. For comparison, we also conduct the MLE training with 2-fold cross validation for the QR model and list the fitting result on the last row in Table 3. Over all, feature 3 (maximum edge coverage) achieves the best fitting performance, which is also better than the QR model. Additionally, feature 5 (average edge coverage) also achieves better fitting performance than the QR model. Table 4 displays the fit result with two features. The best two feature combinations are $(1, 4)$ and $(3, 4)$.

Based on the 2-fold cross validation results, we selected four candidate feature combinations for the QRH model: feature 3, feature 5, feature 1 + feature 4, feature 3 +

Table 4. Fit of model QRH using two features

features	Train on $\boldsymbol{S}_1$		Train on $\boldsymbol{S}_2$		Total
	training	testing	training	testing	testing
(1,2)	-693.2	-672	-635.2	-734.4	-1406.4
(1,3)	-630.4	-594.4	-570	-657.2	-1251.6
(1,4)	-603.6	-602	-573.6	-636	***-1238***
(1,5)	-648.8	-638.4	-606	-684.4	-1322.8
(2,3)	-636	-582.8	-572.4	-656.8	-1239.6
(2,4)	-631.6	-655.2	-638	-649.6	-1304.8
(2,5)	-643.6	-581.2	-566.4	-660.8	-1242
(3,4)	-616	-601.6	-573.6	-644	-1245.6
(3,5)	-636	-581.2	-571.2	-646.8	***-1228***
(4,5)	-610.4	-615.6	-592.4	-635.6	-1251.2

Table 5. Parameters for selected models

features	3	5	(1,4)	(3,5)
parameter value	-9.95	-6.26	(1.04, -10.60)	(-9.67, -1.95)

feature 5. We then tuned the model parameters for these candidates by training on the whole data set $\boldsymbol{S}$. The final values for the parameters are listed in Table 5.

5 Computing Defender Strategy

In this section, we describe how we compute optimal defender strategies against different models of attackers.

5.1 Best Response to QR model

Given a QR model of the adversary, the defender's expected utility by playing strategy $\boldsymbol{x}$ in a network security game Γ is :

$$\sum_{A_i \in \mathcal{A}} q_i(\lambda \mid \boldsymbol{x}; \Gamma) U_i^d(\boldsymbol{x}; \Gamma). \tag{11}$$

Combining with Equation (1) we have the following optimization problem to compute the defender's optimal strategy against a QR model of the adversary:

$$\max_{\boldsymbol{x},p} \frac{\sum_{A_i \in \mathcal{A}} e^{\lambda R_i^a} e^{-\lambda(R_i^a - P_i^a)p_i}((R_i^d - P_i^d)p_i + P_i^d)}{\sum_{A_i \in \mathcal{A}} e^{\lambda R_i^a} e^{-\lambda(R_i^a - P_i^a)p_i}} \tag{12}$$

$$\text{s.t.} \sum_{e \in E} x_e \leq M, \quad 0 \leq x_e \leq 1, \quad \forall e \in E \tag{13}$$

$$p_i = \sum_{e \in A_i} x_e, \quad \forall A_i \in \mathcal{A} \tag{14}$$

where $\lambda = 0.34$ as learned from the data. With only one resource, Constraint (14) ensures that p_i is the probability that path A_i will be covered by the defender. With more than one resources, it gives an upper bound on the probability that A_i will be covered. The objective function, Equation (12), is a nonlinear fractional function, thus is not guaranteed to be concave. We use a heuristic algorithm based on local optimization with random restarts, described in Algorithm 1. The algorithm generates a new starting point in each iteration and (at Line 5) calls `FindLocalMaximum` to find a locally optimal solution of (12). The best local optimal solution is returned in the end. We used Matlab's `fmincon` as the local optimizer.

Algorithm 1: Local Search with Random Multi-Restart

```
1 Input: IterN;
2 opt_g ← −∞;
3 for i ← 1, ..., IterN do
4 |   x_0 ← randomly generated feasible starting point;
5 |   (opt_l, x*) ← FindLocalMaximum(x_0);
6 |   if opt_l > opt_g then
7 |   |   opt_g ← opt_l, x_opt ← x*
8 return opt_g, x_opt;
```

5.2 Best Response to QRH model

In this section, we explain our approach for computing an optimal defender strategy against a QRH model given any combination of features $f_i(\boldsymbol{x})$ and the corresponding feature coefficients μ.

Given a network security game Γ and the defender's strategy $\boldsymbol{x}$, the probability that the attacker will select path A_i is $h_i(\mu \mid \boldsymbol{x}; \Gamma)$ as: defined by Equation 5. Then the defender's expected utility can be written as

$$\sum_{A_i 1 \mathcal{A}} h_i(\mu \mid \boldsymbol{x}; \Gamma) U_i^d(\boldsymbol{x}; \Gamma).$$

Therefore we can formulate the defender's optimal strategy as the solution of the following optimization problem:

$$\max_{\boldsymbol{x}, p} \frac{\sum_{A_i \in \mathcal{A}} e^{\mu \cdot f_i(\boldsymbol{x})} ((R_i^d - P_i^d) p_i + P_i^d)}{\sum_{A_i \in \mathcal{A}} e^{\mu \cdot f_i(\boldsymbol{x})}} \tag{15}$$

$$\text{s.t.} \sum_{e \in E} x_e \leq M, \quad 0 \leq x_e \leq 1, \quad \forall e \in E \tag{16}$$

$$p_i = \sum_{e \in A_i} x_e, \quad \forall A_i \in \mathcal{A} \tag{17}$$

Table 6. Attacker Models Tested Evaluated

Uniform	Defender covers each edge with equal probability
Maximin	Attacker always chooses the worst path for the defender
Rational	Attacker maximizes his expected utility
QR	quantal response ($\lambda = 0.34$)
QRH-1	QRH with maximum edge coverage ($\mu = -9.95$)
QRH-2	QRH with average edge coverage ($\mu = -6.26$)
QRH-3	QRH with number of edges and sum of edge coverage ($\mu = \langle 1.04, -10.60 \rangle$)
QRH-4	QRH with maximum edge coverage and average of edge coverage ($\mu = \langle -9.67, -1.95 \rangle$)

where $f_i(\boldsymbol{x})$ is a subset of the features described in Table 2. Again, the objective function (15) is a nonlinear fractional function, so is not guaranteed to be concave. Nevertheless we can apply Algorithm 1, with `FindLocalMaximum` to find a locally optimal solution of (15).

6 Evaluation

In this section, we evaluate the performance of different models in network security games. We use the same web-based game that we introduced in Section 4.1 to set up the experiments with human subjects. Different from the first set of experiments, where we intended to collect data on how humans play the game in order to train the model, the goal of this new set of experiments is to use the defender strategies computed from the different models to play against human subjects in order to compare the performance of these models.

6.1 Experiment Settings

Fig. 1 shows the interface of the web-based game we developed. We have provided details on the game rules in Section 4.1. We now focus on describing the game instances that are included in these experiments.

We tested eight different graph types, including the three graphs used in data collection that are displayed in Figure 2. The other five graphs are displayed in Figure 3. Among the eight graphs, we have four graphs with a single target (graph 1-4) and four graphs with multiple targets (graph 5-8). The models are trained using the data from single-target graphs, we are interested to see how they perform in multi-target graphs. For each graph type, we designed two different sets of payoffs (i.e. the reward/penalty for the attacker and the defender on each path)[4]. Therefore, we have a total of $8*2 = 16$ security games in the experiments. For each of these games, we computed the defender strategies from eight different models. Table 6 lists the eight models. Therefore, for each game instance, we have eight different defender strategies. In total, we have $8*16 = 128$ different game samples (i.e., combinations of security games and defender strategies). Each of the game samples is played by 40 different subjects.

[4] The details of the payoffs can be found on the online appendix: http://network-security-aamas2012.webs.com/

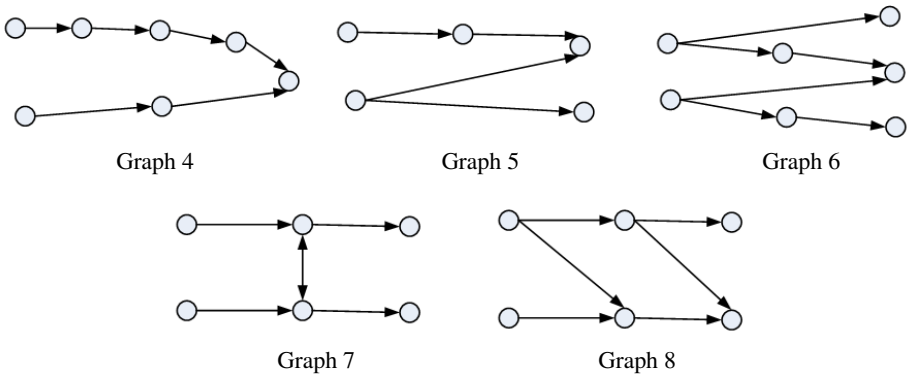


Fig. 3. Graphs Tested in Evaluation Experiments

6.2 Experiment Results

We evaluate the performance of different defender strategies using the defender's expected utility. Given that a subject selects path A_i, the defender's expected utility is computed with Equation (1).

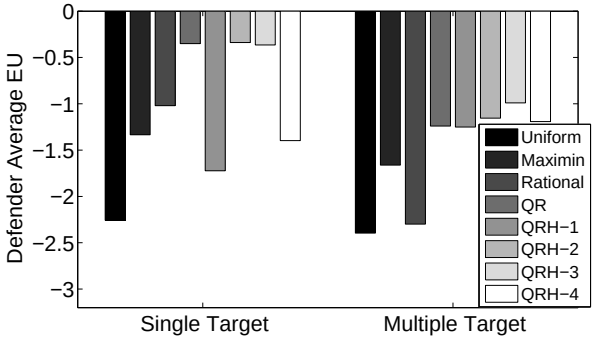
Average Performance: We first evaluated the average defender expected utility, $U^d_{exp}(\boldsymbol{x})$, of different defender strategies based on all 40 subjects choices:

$$U^d_{exp}(\boldsymbol{x}) = \frac{1}{40} \sum_{A_i \in \mathcal{A}} N_i U^d_i(\boldsymbol{x})$$

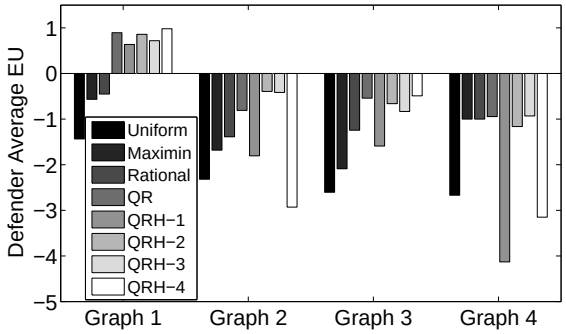
where N_i is the number of subjects that chose path A_i.

Figure 4(a) displays the average performance of the different models in all the single-target games on the left group of bars, and the average model performances in all the multi-target graphs on the right group of bars. In both cases, the QR model outperforms the three baseline models (Uniform, Maximin and Rational). Among the four QRH models, QRH-2 and QRH-3 outperform the three baseline models in both single-target graphs and multi-target graphs. There are two interesting observations from the figure.

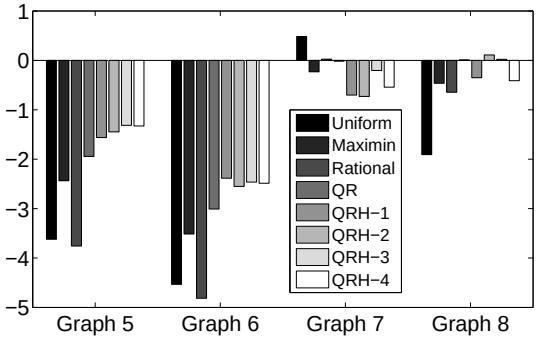
- Between the QR model and the QRH models, we see that in the single-target graphs, none of the QRH models achieves better performance than the QR model; while in the mutli-target graphs, all four QRH models outperform QR. This is an unexpected result since the QRH models are trained on the single-target graph data and do not use features that come up in the multi-target graphs.
- The rational model did worse in the multi-target graphs than it did in the single-target graphs, as compared to the QR and QRH models. For the single-target graphs the average defender expected utility achieved by the rational model was closer to that of the QR and QRH models, and it even outperformed two of the QRH models (QRH-1 and QRH-4). While in the multi-target graphs, the rational model was significantly outperformed by both QR and QRH models. This is also a surprising



(a)



(b) Single Target Graphs



(c) Multiple Target Graphs

Fig. 4. Average Defender Expected Utility

result, since the QR and QRH models are trained on the single-target graphs and are thus expected to perform better in the singe-target graphs.

Table 7. Fitting Performance: log-likelihood of different models

	Multi-Target (8 game instance)	Single-Target (8 game instance)	Total
QR	-198.53	-370.56	-569.09
QRH-1	-416.76	-359.45	-776.22
QRH-2	-216.89	-381.93	-598.82
QRH-3	-231.35	-340.58	-571.93
QRH-4	-406.94	-365.10	-772.04

A possible reason for the above two interesting observations is that as the graph becomes more complex (i.e. more targets and more paths), it becomes more difficult for humans to compute the actual expected utility of each path so they are more likely to rely on heuristics.

We also show the performance of different models in each graph type: Figures 4(b) shows the average defender expected utility achieved by the eight models in the four single-target graphs; and 4(c) displays the results in the four multi-target graphs. We can see from Figure 4(b) that the rational model was outperformed by the QRH-3 model in all of the four graphs; it was also outperformed by the QR model in 3 of the 4 graphs except for in graph 2 where the two models have roughly the same performance. In the multi-target graphs, Figure 4(c) shows that the rational model was outperformed by both the QR and QRH models in three of the four graphs, except for in graph 7 where all of the models have very similar performances.

Model Fitting Performance: We also evaluated the fitting performance of the five trained models. Table 7 reports the total log-likelihood of different models in the multi-target games and the single-target games. In the single target graphs, the QR model achieved much better fitting performance than the QRH model. At the same time, in the multiple target graphs, three of the QRH models (QRH-1, QRH-3, QRH-4) achieved better fitting performance than the QR model. In particular, the QRH-3 model achieved best fitting performance in the multiple target graphs, it also achieved highest defender expected utility, as shown in Figure 4(a). This indicates that to be able to more accurately predict human player's choice will help design better defender strategies.

7 Conclusion

We presented an initial study of human behavior models of adversaries in graph-based security games. In particular, we considered two behavior models, quantal response (QR) and quantal response with heuristics (QRH). In order to train our models and to evaluate their performances, we developed a web-based game that simulates the decision tasks faced by the attacker, posted the game on Amazon Mechanical Turk and collected data on how humans played the game. Once we trained the models, we then computed defender strategies that optimize defender utility against each of these models, and evaluated their performances against human players. Overall, the QR model and two of the QRH models (QRH-2 and QRH-3) outperformed the baseline models (Uniform, Maximin and Rational) in almost all the graphs. We also observed some

interesting phenomena in the models' relative performances going from single-target graphs to multi-target graphs: while the QR and QRH models are trained on single-target graphs, the relative performances of these models (especially QRH) are actually stronger in multi-target graphs. One future direction is to explore more sophisticated feature-selection models such as L1-regularization techniques, which will allow us to select from a large set of candidate features. Another future direction is to tackle large graphs in which it becomes infeasible for the human attacker to list all possible paths.

8 Acknowledgments

This research was supported by Army Research Office under the grant # W911NF-10-1-0185.

References

1. Pita, J., Jain, M., Ordonez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., Kraus, S.: Deployed armor protection: The application of a game theoretic model for security at the los angeles international airport. In AAMAS (2008)
2. Tsai, J., Rathi, S., Kiekintveld, C., Ordonez, F., Tambe, M.: Iris - a tool for strategic security allocation in transportation networks. In AAMAS (2009)
3. Chandran, R., Beitchman, G.: Battle for Mumbai ends, death toll rises to 195. Times of India (November 2008)
4. Ali, S.A.: Rs 18L seized in nakabandi at Vile Parle. Times of India (August 2009)
5. Tsai, J., Yin, Z., Kwak, J., Kempe, D., Kiekintveld, C., Tambe, M.: Urban security: Game-theoretic resource allocation in networked physical domains. In AAAI (2010)
6. Jain, M., Korzhyk, D., Vanek, O., Conitzer, V., Pechoucek, M., Tambe, M.: A double oracle algorithm for zero-sum security games on graphs. In AAMAS (2011)
7. Washburn, A., Wood, K.: Two-person zero-sum games for network interdiction. Operations Research **43**(2) (1995) 243–251
8. Kahneman, D., Tvesky, A.: Prospect theory: An analysis of decision under risk. Econometrica **47**(2) (1979) 263–292
9. Camerer, C.F., Ho, T., Chong, J.: A cognitive hierarchy model of games. QJE **119**(3) (2004) 861–898
10. McKelvey, R.D., Palfrey, T.R.: Quantal response equilibria for normal form games. Games and Economic Behavior **2** (1995) 6–38
11. Stahl, D.O., Wilson, P.W.: Experimental evidence on players' models of other players. JEBO **25**(3) (1994) 309–327
12. Yang, R., Kiekintveld, C., Ordonez, F., Tambe, M., John, R.: Improving resource allocation strategy against human adversaries in security games. In IJCAI (2011)
13. Paruchuri, P., Pearce, J.P., Marecki, J., Tambe, M., Ordonez, F., Kraus, S.: Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In AAMAS (2008)
14. Pita, J., Jain, M., Ordonez, F., Tambe, M., Kraus, S.: Solving stackelberg games in the real-world: Addressing bounded rationality and limited observations in human preference models. Artificial Intelligence Journal **174**(15) (2010) 1142–1171
15. Gigerenzer, G., Todd, P., the ABC Research Group: Simple Heuristics that make us smart. Oxford University Press (1999)
16. Burgess, R., Darken, C.: Realistic human path planning using fluid simulation. In: Proceedings of Behavior Representation in Modeling and Simulation (BRIMS). (2004)