Addressing Execution and Observation Error in Security Games

Manish Jain and **Zhengyu Yin** and **Milind Tambe**
Computer Science Department
Univerisity of Southern California
Los Angeles, CA 90089
{manish.jain, zhengyuy,tambe}@usc.edu

Fernando Ordóñez
Industrial and Systems Engineering
University of Southern California
Los Angeles, CA 90089
Industrial Engineering Department
University of Chile (Santiago)
fordon@usc.edu

Abstract

Attacker-defender Stackelberg games have become a popular game-theoretic approach for security with deployments for LAX Police, the FAMS and the TSA. Unfortunately, most of the existing solution approaches do not model two key uncertainties of the real-world: there may be noise in the defender's execution of the suggested mixed strategy and/or the observations made by an attacker can be noisy. In this paper, we analyze a framework to model these uncertainties, and demonstrate that previous strategies perform poorly in such uncertain settings. We also analyze RECON, a novel algorithm that computes strategies for the defender that are robust to such uncertainties, and explore heuristics that further improve RECON's efficiency.

Introduction

The use of game-theoretic concepts has allowed security forces to exert maximum leverage with limited resources. Indeed, game-theoretic scheduling softwares have been assisting the LAX police, the Federal Air Marshals service, and are under consideration by the TSA (Jain et al. 2010). They have been studied for patrolling (Agmon et al. 2008; Basilico, Gatti, and Amigoni 2009) and routing in networks (Kodialam and Lakshman 2003) and urban transportation networks (Tsai et al. 2010). At the backbone of these applications are attacker-defender Stackelberg games. The solution concept is to compute a strong Stackelberg equilibrium (SSE) (von Stengel and Zamir 2004; Kiekintveld et al. 2009; Conitzer and Sandholm 2006); specifically, the optimal mixed strategy for the defender. It is then assumed that the defender perfectly executes her SSE strategy and the attacker perfectly observes the strategy before choosing his action.

Unfortunately, in the real-world, execution and observation is not perfect due to unforeseen circumstances and/or human errors. For example, a canine unit protecting a terminal at LAX may be urgently called off to another assignment or alternatively a new unit could become available. Similarly, the attacker's observations can be noisy: he may occasionally not observe an officer patrolling a target, or mistake a passing car as a security patrol. Thus, in real-world deployments, the defender may have a noisy execution and the attacker's observations may be even more noisy. A naïve defender strategy can be arbitrarily bad in such uncertain domains, and thus, it is important to develop risk-averse solution techniques that can be used by deployed security systems like ARMOR and IRIS (Jain et al. 2010).

This paper models execution and observation uncertainty, and presents efficient solution techniques to compute risk-averse defender strategies. Previous work has failed to provide such efficient solution algorithms. While the COBRA algorithm (Pita et al. 2010) focuses on human subjects' inference when faced with limited observations of defender strategies, it does not consider errors in such observations. In contrast, Yin et. al (2010) consider the limiting case where an attacker has no observations and thus investigate the equivalence of Stackelberg vs Nash equilibria. Even earlier investigations have emphasized the value of commitment to mixed strategies in Stackelberg games in the presence of noise (Bagwell 1995; Huck and Mller 2000; Morgan and Vardy 2007; van Damme and Hurkens 1997). Alternate models where the attacker is uncertain about the utility of a target have also been proposed (Jenelius, Westin, and Holmgren 2010). Outside of Stackelberg games, models for execution uncertainty have been separately developed (Iyengar 2005; Archibald and Shoham 2009). Algorithms that model human behavior have also been proposed (Yang et al. 2011). Our research complements these efforts by providing a unified efficient algorithm that addresses both execution and observation uncertainties; in this way it also complements other research that addresses payoff uncertainties in such games (Jain et al. 2010).

Recent work by Yin et. al (2011) provides RECON, a mixed-integer linear programming formulation that computes the risk-averse strategy for the defender in domains with execution and observation uncertainty. RECON assumes that nature chooses noise to maximally reduce defenders utility, and RECON maximizes against this worst case. Yin et. al (2011) also provide two heuristics that speed up the computation of RECON. This paper extends that work and provides three key contributions: (1) an in-depth analysis of the robust strategies provided by RECON. (2) novel solution quality details for realistic security situations with execution errors obtained using simulations with an attacker who has limited observation capabilities, and (3) new and detailed runtime results of heuristics to improve RECON's

performance.

Background and Notation

A Stackelberg security game is a game between two players: a defender and an attacker. The defender wishes to deploy up to γ security resources to protect a set of targets T from the attacker. Each player has a set of *pure strategies*: the defender can *cover* a set of targets, and the attacker can *attack* one target. The payoffs for each player depend on the target attacked, and whether or not the attack was successful. $U_d^u(t_i)$ and $U_d^c(t_i)$ represent the utilities for the defender when t_i, the target attacked, was uncovered and covered respectively. The attacker's utilities are denoted similarly by $U_a^c(t_i)$ and $U_a^u(t_i)$. We use $\Delta U_d(t_i) = U_d^c(t_i) - U_d^u(t_i)$ to denote the difference between defender's covered and uncovered utilities. Similarly, $\Delta U_a(t_i) = U_a^u(t_i) - U_a^c(t_i)$. A key property of security games is that $\Delta U_d(t_i) > 0$ and $\Delta U_a(t_i) > 0$. Example payoffs for a game with five targets, t_1 and t_2, and one resource are given in Table 1.

	t_1	t_2	t_3	t_4	t_5
U_d^c	4	3	2	8	9
U_d^u	-1	-6	-3	-5	-6
U_a^c	-2	-5	-1	-7	-9
U_a^u	3	6	1	7	5

Table 1: Example Security Game with 5 targets.

A strategy profile $\langle \mathbf{x}, t_i \rangle$ for this game is a mixed strategy $\mathbf{x}$ for the defender, and the attacked target t_i. The mixed strategy $\mathbf{x} = \langle x_i \rangle$ is a vector of probabilities of defender *coverage* over all targets (Yin et al. 2010), such that the sum total of coverage is not more than the number of available resources γ. For example, a mixed strategy for the defender can be .25 coverage on t_1 and .75 coverage on t_2. We allow y_i, the defender's actual coverage on t_i, to vary from the intended coverage x_i by the amount α_i, that is, $|y_i - x_i| \leq \alpha_i$. Thus, if we set $\alpha_1 = 0.1$, it would mean that $0.15 \leq y_1 \leq 0.35$. Additionally, we assume that the attacker wouldn't necessarily observe the actual implemented mixed strategy of the defender; instead the attacker's perceived coverage for t_i, denoted by z_i, can vary by β_i from the implemented coverage y_i. Therefore, $|z_i - y_i| \leq \beta_i$. Thus, in our example, if y_1 was 0.3 and β_1 was set to 0.05, then $0.25 \leq z_1 \leq 0.35$. Table 2 summarizes the notation used in this paper.

For example, at LAX, ARMOR might generate a schedule for two canines to patrol Terminals 1, 2, 3, 4 with probabilities of 0.2, 0.8, 0.5, 0.5 respectively. However, a last-minute cargo inspection may require a canine unit to be called away from, say, Terminal 2 in its particular patrol, or an extra canine unit may become available by chance and get sent to Terminal 3. Additionally, an attacker may fail to observe a canine patrol on a terminal, or he may mistake an officer walking across as engaged in a patrol. Since each target is patrolled and observed independently, we assume that both execution and observation noise are independent per target.

Variable	Definition
T	Set of targets
$U_d^u(t_i)$	Defender's payoff if target t_i is uncovered
$U_d^c(t_i)$	Defender's payoff if target t_i is covered
$U_a^u(t_i)$	Attacker's payoff if target t_i is uncovered
$U_a^c(t_i)$	Attacker's payoff if target t_i is covered
γ	Number of defender resources
x_i	Defender's intended coverage of target t_i
y_i	Defender's actual coverage of target t_i
z_i	Attacker's observed coverage of target t_i
$\Delta U_d(t_i)$	$\Delta U_d(t_i) = U_d^c(t_i) - U_d^u(t_i)$
$\Delta U_a(t_i)$	$\Delta U_a(t_i) = U_a^u(t_i) - U_a^c(t_i)$
$D_i(x_i)$	Defender's expected utility for target t_i $D_i(x_i) = U_d^u(t_i) + \Delta U_d(t_i)x_i$
$A_i(x_i)$	Attacker's expected utility for target t_i $A_i(x_i) = U_a^u(t_i) - \Delta U_a(t_i)x_i$
α_i	Maximum execution error for target t_i
β_i	Maximum observation error for target t_i

Table 2: Notation

Again, consider the example in Table 1, suppose the defender has one resource. The SSE strategy for the defender would be protecting t_1 and t_2 with 0.5 probability each, making them indifferent for the attacker. The attacker breaks ties in defender's favor and chooses t_1 to attack, giving the defender an expected utility of 5. This SSE strategy is not robust to any noise – by deducting an infinitesimal amount of coverage probability from t_2, the attacker's best response changes to t_2, reducing the defender's expected utility to -5. RECON computes a risk-averse strategy, which provides the defender the maximum worst-case expected utility. For example, assuming no execution error and 0.1 observational uncertainty ($\alpha = 0$ and $\beta = 0.1$), the optimal risk-averse defender strategy is to protect t_1 with $0.4 - \epsilon$ probability and t_2 with $0.6 + \epsilon$ probability so that even in the worst-case, the attacker would choose t_1, giving the defender an expected utility of 4. Finding the optimal risk-averse strategy for large games remains difficult, as it is essentially a *bi-level* programming problem (Bard 2006).

Problem Statement

The objective of RECON is to find the optimal risk-averse strategy $\mathbf{x}$, maximizing the worst-case defender utility, $U_d^*(\mathbf{x})$ (Constraint (1) and (2)). Given a fixed maximum execution and observation noise, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ respectively, $U_d^*(\mathbf{x})$ is computed by the minimization problem from Constraint (3) to (6).

The overall problem is a bi-level programming problem. For a fixed defender strategy $\mathbf{x}$, the *second-level* problem from Constraint (3) to (6) computes the worst-case defender's executed coverage $\mathbf{y}$, the attacker's observed coverage $\mathbf{z}$, and the target attacked t_j. $\langle \mathbf{y}, \mathbf{z}, t_j \rangle$ is chosen such that the defender's expected utility $D_j(y_j)$ (see Table 2) is minimized, given that the attacker maximizes his believed utility[1] $A_j(z_j)$ (Constraint (4)). This robust optimization is

[1]The attacker's believed utility is computed using the strategy

similar in spirit to Aghassi and Bertsimas (2006), although that is in the context of simultaneous move games.

$$\max_{\mathbf{x}} \quad U_d^*(\mathbf{x}) \tag{1}$$

$$\text{s.t.} \quad \sum_{t_i \in T} x_i \leq \gamma,\ 0 \leq x_i \leq 1 \tag{2}$$

$$U_d^*(\mathbf{x}) = \min_{\mathbf{y},\mathbf{z},t_j} \quad D_j(y_j) \tag{3}$$

$$\text{s.t.} \quad t_j \in \arg\max_{t_i \in T} A_i(z_i) \tag{4}$$

$$-\alpha_i \leq y_i - x_i \leq \alpha_i,\ 0 \leq y_i \leq 1 \tag{5}$$

$$-\beta_i \leq z_i - y_i \leq \beta_i,\ 0 \leq z_i \leq 1 \tag{6}$$

Additionally, the above formulation also highlights the need to separately model both execution and observation noise. Indeed a problem with uncertainty defined as $\langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle$ is different from a problem with $\langle \boldsymbol{\alpha}' = 0, \boldsymbol{\beta}' = \boldsymbol{\alpha} + \boldsymbol{\beta} \rangle$ (or vice-versa), since the defender utility is different in the two problems.

Approach

RECON, *Risk-averse Execution Considering Observational Noise*, is a mixed-integer linear programming (MILP) formulation that compute the risk-averse defender strategy in the presence of execution and observation noise. It encodes the necessary and sufficient conditions of the second-level problem (Constraint (4)) as linear constraints. The intuition behind these constraints is to identify $S(\mathbf{x})$, the *best-response* action set for the attacker given a strategy $\mathbf{x}$, and then break ties against the defender. Additionally, RECON represents the variables $\mathbf{y}$ and $\mathbf{z}$ in terms of the variable $\mathbf{x}$ – it reduces the bi-level optimization problem to a single-level optimization problem. Here, we first define the term *inducible target* and then the associated necessary/sufficient conditions of the second level problem.

Definition 1. *A target t_j is weakly inducible by a mixed strategy $\mathbf{x}$ if there exists a strategy $\mathbf{z}$ with $0 \leq z_i \leq 1$ and $|z_i - x_i| \leq \alpha_i + \beta_i$ for all $t_i \in T$, such that t_j is the best response to $\mathbf{z}$ for the attacker, i.e., $t_j = \arg\max_{t_i \in T} A_i(z_i)$.*

Additionally, Yin et. al (2011) define the upper and lower bounds on the utility the attacker may believe to obtain for the strategy profile $\langle \mathbf{x}, t_i \rangle$. These bounds are then used to determine the *best response* set of the attacker.

Definition 2. *For the strategy profile $\langle \mathbf{x}, t_i \rangle$, the upper bound of attacker's believed utility is given by $A_i^+(x_i)$, which would be reached when the attacker's observed coverage of t_i reaches the lower bound* $\max\{0, x_i - \alpha_i - \beta_i\}$.

$$A_i^+(x_i) = \min\{U_a^u(t_i), A_i(x_i - \alpha_i - \beta_i)\} \tag{7}$$

Similarly, the lower bound of attacker's believed utility of attacking target t_i is denoted as $A_i^-(x_i)$, which is reached when the attacker's observed coverage probability on t_i reaches the upper bound $\min\{1, x_i + \alpha_i + \beta_i\}$.

$$A_i^-(x_i) = \max\{U_a^c(t_i), A_i(x_i + \alpha_i + \beta_i)\} \tag{8}$$

observed by the attacker, and it may not be achieved, since $\mathbf{z}$ can be different from $\mathbf{y}$, which can be different from $\mathbf{x}$.

Lemma 1. *A target t_j is weakly inducible by $\mathbf{x}$ if and only if $A_j^+(x_j) \geq \max_{t_i \in T} A_i^-(x_i)$.*

Proof. If t_j is weakly inducible, consider $\mathbf{z}$ such that $t_j = \arg\max_{t_i \in T} A_i(z_i)$. Since $z_j \geq \max\{0, x_j - \alpha_j - \beta_j\}$ and for all $t_i \neq t_j$, $z_i \leq \min\{1, x_i + \alpha_i + \beta_i\}$, we have:

$$A_j^+(x_j) = \min\{U_a^u(t_j), A_j(x_j - \alpha_j - \beta_j)\} \geq A_j(z_j)$$
$$\geq A_i(z_i) \geq \max\{U_a^c(t_i), A_i(x_i + \alpha_i + \beta_i)\} = A_i^-(x_i).$$

On the other hand, if $A_j^+(x_j) \geq A_i^-(x_i)$ for all $t_i \in T$, we can let $z_j = \max\{0, x_i - \alpha_j - \beta_j\}$ and $z_i = \min\{1, x_i + \alpha_i + \beta_i\}$ for all $t_i \neq t_j$, which satisfies $t_j = \arg\max_{t_i \in T} A_i(z_i)$. This implies t_j is weakly inducible. □

We also define $D_i^-(x_i)$, the lower bound on the defender's expected utility for the strategy profile $\langle \mathbf{x}, t_i \rangle$. This lower bound is used to determine the defender's worst-case expected utility.

Definition 3. *For the strategy profile $\langle \mathbf{x}, t_i \rangle$, $D_i^-(x_i)$ is achieved when the defender's implemented coverage on t_i reaches the lower bound* $\max\{0, x_i - \alpha_i\}$*, and is given by:*

$$D_i^-(x_i) = \max\{U_d^u(t_i), D_i(x_i - \alpha_i)\} \tag{9}$$

Lemma 2. *Let $S(\mathbf{x})$ be the set of all targets that are weakly inducible by $\mathbf{x}$, then $U_d^*(\mathbf{x}) = \min_{t_i \in S(\mathbf{x})} D_i^-(x_i)$.*

Proof. The proof proceeds by showing that the defender utility $U_d^*(\mathbf{x})$ is upper bounded by $\min_{t_i \in S(\mathbf{x})} D_i^-(x_i)$ since only the targets in $S(\mathbf{x})$ are weakly inducible; we omit the formal details since they can be found in Yin et. al (2011). □

Lemma (1) and (2) are the necessary and sufficient conditions for the second level optimization problem, reducing the bi-level optimization problem into a single level MILP.

RECON MILP

Now, we present the MILP formulation for RECON. It maximizes the defender utility, denoted as v_d. v_a represents the *highest lower-bound* on the believed utility of the attacker ($A_i^+(x_i)$), given in Constraint (11). The binary variable q_i is 1 if the target t_i is weakly inducible; it is 0 otherwise. Constraint (12) says that $q_i = 1$ if $A_i^+(x_i) \geq v_a$ (ϵ is a small positive constant which ensures that $q_i = 1$ when $A_i^+(x_i) = v_a$) and together with Constraint (11), encodes Lemma 1. The constraint that $q_i = 0$ if $A_i^+(x_i) < v_a$ could be added to RECON, however, it is redundant since the defender wants to set $q_i = 0$ in order to maximize v_d. Constraint (13) says that the defender utility v_d is less than $D_i^-(x_i)$ for all inducible targets, thereby implementing Lemma 2. Constraint (14) ensures that the allocated resources are no more than

the number of available resources γ, maintaining feasibility.

$$\max_{\mathbf{x},\mathbf{q},v_d,v_a} \quad v_d \tag{10}$$

$$\text{s.t.} \quad v_a = \max_{t_i \in T} A_i^-(x_i) \tag{11}$$

$$A_i^+(x_i) \leq v_a + q_i M - \epsilon \tag{12}$$

$$v_d \leq D_i^-(x_i) + (1 - q_i)M \tag{13}$$

$$\sum_i x_i \leq \gamma \tag{14}$$

$$x_i \in [0, 1] \tag{15}$$

$$q_i \in \{0, 1\} \tag{16}$$

The $\max$ function in Constraint (11) can be formulated using $|T|$ binary variables, $\{h_i\}$, in the following manner:

$$A_i^-(x_i) \leq v_a \leq A_i^-(x_i) + (1 - h_i)M \tag{17}$$

$$\sum_{t_i \in T} h_i = 1, \quad h_i \in \{0, 1\} \tag{18}$$

The $\min$ operation in $A_i^+(x_i)$ is also implemented similarly. For example, Equation (7) can be encoded as:

$$U_a^u(t_i) - (1 - \nu_i)M \leq A_i^+ \leq U_a^u(t_i)$$
$$A_i(x_i - \alpha_i - \beta_i) - \nu_i M \leq A_i^+ \leq A_i(x_i - \alpha_i - \beta_i)$$
$$\nu_i \in \{0, 1\}$$

We omit the details for expanding $A_i^-(x_i)$ and $D_i^-(x_i)$, they can be encoded in a similar fashion.

Analysis

Overall, RECON tends to put more protection on vulnerable targets (those give the defender very low utilities) to make them less likely to be attacked. COBRA does this too, however the difference is that RECON gives an optimal solution with a given noise of attacker's observations whereas using a fixed ϵ as bounded-rationality for the attacker in COBRA is just a heuristic which may be difficult to set in practice. A key property of RECON is that it computes the SSE where there is no uncertainty, that is, $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$. Furthermore, a MAXIMIN strategy is obtained when $\boldsymbol{\beta} = \mathbf{1}$ with $\boldsymbol{\alpha} = \mathbf{0}$. In such situations, even though the defender has no execution uncertainty, the attacker can choose to attack any target since $\mathbf{z}$ can be arbitrary. Additionally, $\boldsymbol{\alpha} = \mathbf{1}$ implies that the execution of the defender is independent of $\mathbf{x}$ and thus, any feasible $\mathbf{x}$ is optimal. Thus, RECON provides a full range of robust solutions of the problem by just setting noise between 0 and 1 which can be quite valuable for end users in practice.

Table 1 shows the payoff values for 5 targets in sample security game. In this sample security game, the optimal strategy for the defender assuming $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$ (i.e., no uncertainty) gives the defender a payoff of -0.31 units where the optimal action of the attacker is to attack target t_1. In this setting, the attack set of the attacker comprises of only the target t_1.[2] As the uncertainty increases, more targets enter the attack set, and so the defender has to distribute her resources across more targets.

[2]Attack set is the set of targets that represent the attacker's best response; the attacker is indifferent to all targets that belong to this set (Kiekintveld et al. 2009).

α	β	Attack Set	$U_d^*(\mathbf{x})$
0	0	$\{t_1\}$	-0.31
0	0.02	$\{t_1, t_4\}$	-0.49
0.01	0.01	$\{t_1\}$	-0.55
0.02	0	$\{t_1\}$	-0.60
0.05	0.05	$\{t_1, t_4\}$	-1.70
0	1	$\{t_1, t_2, t_3, t_4, t_5\}$	-2.31
1	$*$	$\{t_1, t_2, t_3, t_4, t_5\}$	-6.0

Table 3: RECON output for security game given in Table 1.

Table 3 shows sample results for RECON for different settings for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Firstly, as expected, higher uncertainty leads to lower defender reward (as we move down the table). Secondly, it is harder for the defender to protect against execution error as opposed to observation error. For example, the defender is able to achieve a higher payoff of -0.49 units when the observation error is 0.2 with $\boldsymbol{\alpha} = 0$, however, the reward drops to -0.6 when values of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are swapped. Finally, execution error of 1.0 implies that the defender's strategy is irrelevant; there is so much execution noise that $\mathbf{y}$ is independent of $\mathbf{x}$. Thus, in the worst case, the attacker will succeed in attacking any single target and the defender achieves the worst possible utility (here, worst utility for the defender equals $U_d^u(t_4) = -6$). On the other hand, COBRA with an ϵ value (bounded rationality) of 2 units has an attack set of targets $\{t_1, t_4\}$ with the expected defender utility of -1.38. This utility drops to -2.07 when the value of ϵ is changed to 3 with the attack set changing to $\{t_1, t_2, t_4, t_5\}$. This again shows that while COBRA can generate robust solutions, an over-estimate of ϵ can lead to a significant loss in expected defender utility.

Speeding up

As described above, RECON uses a MILP formulation to compute the risk-averse strategy for the defender. Integer variables increase the complexity of the linear programming problem; indeed solving integer programs is NP-hard. MILP solvers internally use branch-and-bound to evaluate integer assignments. Availability of good lower bounds implies that less combinations of integer assignments (branch-and-bound nodes) need to be evaluated. This is the intuition behind speeding up the execution of RECON. We provide two methods, a-RECON and i-RECON, to generate lower bounds.

a-RECON: a-RECON solves a *restricted* version of RECON. This restricted version has lower number of integer variables, and thus generates solutions faster. It replaces $A_i^+(x_i)$ by $A_i(x_i - \alpha_i - \beta_i)$ and $D_i^-(x_i)$ by $D_i(x_i - \alpha_i)$, thereby rewriting Constraints (12) and (13) as follows:

$$A_i(x_i - \alpha_i - \beta_i) \leq v_a + q_i M - \epsilon \tag{19}$$

$$v_d \leq D_i(x_i - \alpha_i) + (1 - q_i)M \tag{20}$$

a-RECON is indeed more *restricted* – the LHS of Constraint (19) in a-RECON is *no less than* the LHS of Constraint (12) in RECON; and the RHS of Constraint (20) is *no greater than* the RHS of Constraint (13) in a-RECON. Therefore, any solution generated by a-RECON is feasible in RECON, and

Algorithm 1: Pseudo code of i-RECON

```
k = 0, v_d^(0) = v_a^(0) = −∞;
while |v_a^(k+1) − v_a^(k)| ≤ η and |v_d^(k+1) − v_d^(k)| ≤ η do
    v_a^(k+1) = Solve(A-LP (v_d^(k), v_a^(k)));
    v_d^(k+1) = Solve(D-LP (v_d^(k), v_a^(k)));
    k = k + 1;
end
```

acts as a lower bound. We do not restrict $A_i^-(x_i)$ since the RHS of Constraint (17) is non-trivial for only one target.

i-RECON: i-RECON uses an iterative method to obtain monotonically increasing lower bounds $v_d^{(k)}$ of RECON. Using the insight that Constraint (19) is *binding* only when $q_i = 0$, and (20) when $q_i = 1$, i-RECON rewrites Constraints (19) and (20) as follows:

$$x_i \geq \begin{cases} \tau_{a,i}(v_a) = \frac{U_a^u(t_i) - v_a + \epsilon}{\Delta U_a(t_i)} + \alpha_i + \beta_i & \text{if } q_i = 0 \\ \tau_{d,i}(v_d) = \frac{v_d - U_d^u(t_i)}{\Delta U_d(t_i)} + \alpha_i & \text{if } q_i = 1 \end{cases} \quad (21)$$

which says that $q_i = 0$ implies $x_i \geq \tau_{a,i}(v_a)$ and $q_i = 1$ implies $x_i \geq \tau_{d,i}(v_d)$.[3]

Constraint (21) can also be represented as follows (Yin et al. 2011):

$$\begin{aligned} x_i \geq\ & \min\{\tau_{d,i}(v_d), \tau_{a,i}(v_a)\} \\ =\ & \tau_{d,i}(v_d) + \min\{0, \tau_{a,i}(v_a) - \tau_{d,i}(v_d)\} \end{aligned} \quad (22)$$

Observe that $\tau_{d,i}(v_d)$ is increasing in v_d where as $\tau_{a,i}(v_a)$ is decreasing in v_a (refer Constraint (21)), and hence $\tau_{d,i}(v_d) - \tau_{a,i}(v_a)$ is *increasing* in both v_d and v_a. i-RECON generates an increasing sequence of $\{\Delta\tau_i^{(k)} = \tau_{d,i}(v_d^{(k)}) - \tau_{a,i}(v_a^{(k)})\}$ by finding increasing sequences of $v_d^{(k)}$ and $v_a^{(k)}$. As is described in Yin et. al (2011), substituting $\tau_{d,i}(v_d) - \tau_{a,i}(v_a)$ with $\{\Delta\tau_i^{(k)}\}$ in Constraint (22) guarantees correctness. Since a higher value of $\Delta\tau_i^{(k)}$ implies a lower value of $\min\{0, -\Delta\tau_i^{(k)}\}$, a weaker restriction is imposed by Constraint (22), leading to a better lower bound $v_d^{(k+1)}$.

Given $v_d^{(k)}$ and $v_a^{(k)}$, i-RECON uses D-LP to compute the $v_d^{(k+1)}$, and A-LP to compute $v_a^{(k+1)}$. The pseudo-code for i-RECON is given in Algorithm 1. D-LP is the following maximization linear program, which returns the solution vector $\langle \mathbf{x}, v_d, \hat{v}_a \rangle$, such that v_d is the desired lower bound.

$$\begin{aligned} \max_{\mathbf{x}, v_d, \hat{v}_a}\ & v_d \\ \text{s.t. Constraint}&(11), (14) \text{ and } (15) \\ & x_i \geq \tau_{d,i}(v_d) + \min\{0, -\Delta\tau_i^{(k)}\} && (23) \\ & v_d \geq v_d^{(k)}; \quad \hat{v}_a \geq v_a^{(k)} && (24) \end{aligned}$$

Constraint (24) is added to D-LP to ensure that we get a monotonically increasing solution in every iteration. Similarly, given $v_d^{(k)}$ and $v_a^{(k)}$, A-LP is the following minimization problem. It minimizes v_a to guarantee that Constraint (23) in D-LP remains a restriction to Constraint (22) for the next iteration, ensuring D-LP always provides a lower bound of RECON. More details are in Proposition 1 which proves the correctness of i-RECON.

$$\begin{aligned} \min_{\mathbf{x}, v_d, v_a}\ & v_a \\ \text{s.t. Constraint}&(11), (14) \text{ and } (15) \\ & x_i \geq \tau_{a,i}(v_a) + \min\{\Delta\tau_i^{(k)}, 0\} && (25) \\ & v_a \geq v_a^{(k)} && (26) \end{aligned}$$

Proposition 1. *Both* D-LP *and* A-LP *are feasible and bounded for every iteration k until i-*RECON *converges.*

Proof. We omit the proof of the proposition; the details can be found in Yin et. al (2011). □

Experimental Results

We provide two sets of experimental results: (1) we compare the solution quality of RECON, ERASER, and COBRA under uncertainty: ERASER (Jain et al. 2010) is used to compute the SSE solution, where as COBRA (Pita et al. 2010) is one of the latest algorithms that addresses attacker's observational error.[4] We also provide these solution quality results obtained using simulations with execution and observational uncertainty. (2) We provide the runtime results of RECON, showing the effectiveness of the two heuristics a-RECON and i-RECON. In all test instances, we set the number of defender resources to 20% of the number of targets. Payoffs $U_d^c(t_i)$ and $U_a^u(t_i)$ are chosen uniformly randomly from 1 to 10 while $U_d^u(t_i)$ and $U_a^c(t_i)$ are chosen uniformly randomly from -10 to -1. The results were obtained using CPLEX on a standard 2.8GHz machine with 2GB main memory, and averaged over 39 trials. All comparison results are statistically significant under t-test ($p < 0.05$).

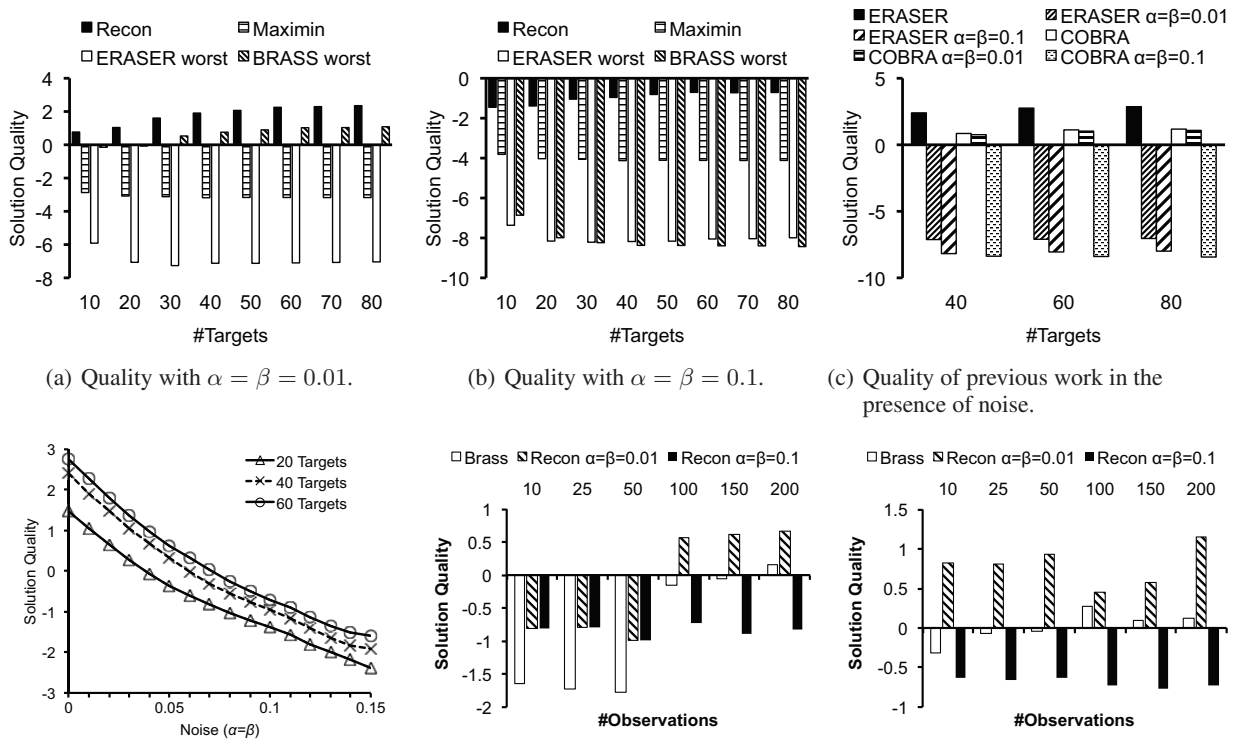
Measuring robustness of a given strategy:

Given a defender mixed strategy $\mathbf{x}$, a maximum execution error $\boldsymbol{\alpha}$, and a maximum possible observation error $\boldsymbol{\beta}$, the worst-case defender utility is computed using the following MILP. This MILP finds the executed strategy $\mathbf{y}$, and the observed strategy $\mathbf{z}$ that hurt the defender the most.

$$\begin{aligned} \min_{\mathbf{y}, \mathbf{z}, \mathbf{q}, v_d, v_a}\ & v_d && (27) \\ \text{s.t. }& v_d \geq D_i(y_i) - (1 - q_i)M && (28) \\ & A_i(z_i) \leq v_a \leq A_i(z_i) + (1 - q_i)M && (29) \\ & -\alpha_i \leq y_i - x_i \leq \alpha_i && (30) \\ & -\beta_i \leq z_i - y_i \leq \beta_i && (31) \\ & \sum_{t_i \in T} q_i = 1 && (32) \\ & y_i, z_i \in [0, 1]; \quad q_i \in \{0, 1\} && (33) \end{aligned}$$

[3] This is *not* equivalent to the unconditional equation $x_i \geq \max\{\tau_{a,i}(v_a), \tau_{d,i}(v_d)\}$.

[4] The bounded rationality parameter ϵ in COBRA is set to 2 as suggested by Pita et al.(2010). The bias parameter α is set to 1 since our experiments are not tested against human subjects.



(a) Quality with $\alpha = \beta = 0.01$.

(b) Quality with $\alpha = \beta = 0.1$.

(c) Quality of previous work in the presence of noise.

(d) Quality of RECON with increasing noise.

(e) Solution quality of RECON and COBRA in simulation for 10 targets.

(f) Solution quality of RECON and COBRA in simulation for 20 targets.

Figure 1: Experimental results: Quality of solution.

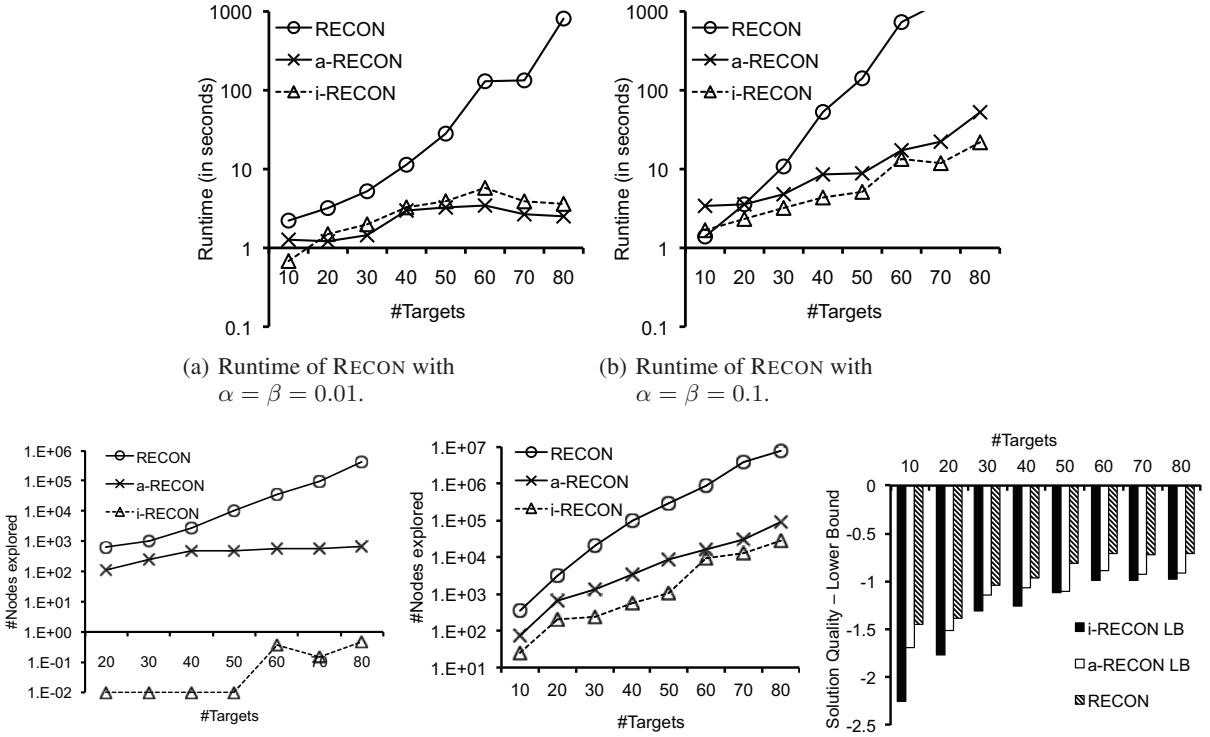
This MILP is similar to the second-level optimization problem given in Constraints (3) to (6). Here, $\mathbf{x}$ is the input defender strategy whose robustness has to be evaluated, while $\mathbf{y}$ and $\mathbf{z}$ are continuous variables and again represent defender's execution and attacker's observation. In addition, the variables v_d and v_a denote the defender's utility and the attacker's utility with respect to $\mathbf{y}$ and $\mathbf{z}$. Finally, the integer variables q_i enforce the best response constraint of the attacker, i.e. $v_a = \max_{t_i \in T} A_i(z_i)$.

Figure 1(a) and Figure 1(b) presents the comparisons between the worst-case utilities of RECON, ERASER and COBRA under two uncertainty settings – low uncertainty ($\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0.01}$) and high uncertainty ($\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0.1}$). Also, MAXIMIN utility is provided as a benchmark. Here x-axis shows the number of targets and y-axis shows the defender's worst-case utility. RECON significantly outperforms MAXIMIN, ERASER and COBRA in both uncertainty settings. For example, in high uncertainty setting, for 80 targets, RECON on average provides a worst-case utility of -0.7, significantly better than MAXIMIN (-4.1), ERASER (-8.0) and COBRA (-8.4).

Next, in Figure 1(c), we present the ideal defender utilities of ERASER and COBRA assuming no execution and observational uncertainties, comparing to their worst-case utilities (computed as described above). Again, x-axis is the number of targets and y-axis is the defender's worst-case utility. As we can see, ERASER is not robust – even 0.01 noise reduces the solution quality significantly. For instance, for 80 targets with low uncertainty, ERASER on average has a worst-case utility of -7.0 as opposed to an ideal utility of 2.9. Similarly, COBRA is not robust to large amount of noise (0.1) although it is robust when noise is low (0.01). Again, for 80 targets, COBRA on average has an ideal utility of 1.2,however, its worst-case utility drops to -7.0 in high uncertainty setting.

Moreover, in Figure 1(d), we show the quality of RECON with increasing noise from $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$ to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0.15}$. The x-axis shows the amount of noise while the y-axis shows the defender's utility returned by RECON. The three lines represent 20, 40, and 60 targets respectively. As we can see, while ERASER and COBRA cannot adapt to noise even when bounds on noise are known *a-priori*, RECON is more robust and provides significantly higher defender utility.

Finally, we also conducted simulations considering uncertainty arising from limited number of observations available to an attacker. The simulation was set up such that in each *trial* for each input game, the attacker will get a fixed number of observations based on which he will estimate the defender's strategy, and eventually compute his best response. Each observation of the attacker is a sample of the defender strategy, where each sample is the exact set of targets protected by the defender resources. The attacker chooses his best response t^* after the given number of observations; the defender draws another sample from her strategy and the payoff of the defender is $U_d^c(t^*)$ or $U_d^u(t^*)$ depending on



(a) Runtime of RECON with $\alpha = \beta = 0.01$.

(b) Runtime of RECON with $\alpha = \beta = 0.1$.

(c) Number of nodes explored by different algorithms with $\alpha = \beta = 0.01$.

(d) Number of nodes explored by different algorithms with $\alpha = \beta = 0.1$.

(e) Comparison of lower bounds provided by different algorithms with $\alpha = \beta = 0.1$.

Figure 2: Experimental results: Number of nodes explored by 3 variants of RECON.

whether or not t^* is covered in this sample. For each game, the above process is repeated 30 times and the defender utility is the average defender payoff over these 30 trials.

The attacker's observed coverage probability for any target follows a binomial distribution, depending on the number of observations. More the number of observations, more accurate is the attacker's observed strategy compared to the intended defender strategy. For example, a target with an intended defender coverage probability of 0.5 will have a 0.66 probability of being observed as covered 4 to 6 times in 10 observations (0.1 noise). However, the probability of observing the same target covered between 40 to 60 times in 100 observations goes up to 0.94.

The simulation results are presented in Figures 1(e) and 1(f), where the y-axis shows the defender utility averaged over 30 random security games and the x-axis shows the number of observations allowed to the attacker. We exclude MAXIMIN and ERASER from this experiment, since their worst case performance in the presence of uncertainty is significantly poor (refer Figure 1(a)). We compare RECON with $\boldsymbol{\alpha = \beta = 0.01}$ and $\boldsymbol{\alpha = \beta = 0.1}$ against COBRA. For example, for 10 targets and 100 observations, COBRA obtains a solution quality of -0.15 whereas RECON with $\boldsymbol{\alpha = \beta = 0.01}$ obtains 0.56. On the other hand, RECON with $\boldsymbol{\alpha = \beta = 0.1}$ obtains -0.73. While RECON with $\boldsymbol{\alpha = \beta = 0.01}$ always outperforms COBRA, RECON with $\boldsymbol{\alpha = \beta = 0.1}$ loses out since it sacrifices the expected solution quality for worst case protection.

Runtime results of RECON:

Figures 2(a) and 2(b) show the runtime results of the three variants of RECON— RECON without any lower bounds, and with lower bounds provided by a-RECON and i-RECON respectively. The x-axis shows the number of targets and the y-axis (in logarithmic scale) shows the total runtime in seconds. Both a-RECON and i-RECON heuristics help reduce the total runtime significantly in both uncertainty settings – the speedup is of orders of magnitude in games with large number of targets. For instance, for cases with 80 targets and high uncertainty, RECON without heuristic lower bounds takes $3,948$ seconds, whereas RECON with a-RECON lower bound takes a total runtime of 52 seconds and RECON with i-RECON lower bound takes a total runtime of 22 seconds.

Furthermore, we can see in Figure 2(a) and 2(b), a-RECON works better than i-RECON in low uncertainty setting while i-RECON works better than a-RECON in high uncertainty setting. The reason can be explained by the fact that a-RECON needs to explore significantly more nodes in higher uncertainty setting since the weakly inducible set tends to be bigger. We summarize the number of nodes CPLEX explored for the three algorithms in Figures 2(c) and 2(d) where the y-axis (in logarithmic scale) shows the

number of nodes explored and the x-axis shows the number of targets. As an instance, for 80 targets and high uncertainty, RECON explores 7.6×10^6 nodes with no lower bound, 9.4×10^4 nodes with a-RECON lower bound (total number of nodes explored by RECON MILP and a-RECON MILP), and 2.8×10^4 nodes with i-RECON lower bound.

We further compare the performance of a-RECON and i-RECON in terms of their runtime and lower bounds returned. As shown in Figure 2(e), both a-RECON and i-RECON can provide lower bounds close to the optimum even in a high uncertainty setting, justifying their effectiveness of speeding the RECON MILP. Comparing to i-RECON, a-RECON provides a slightly tighter lower bound, however, the difference is diminishing with increasing number of targets.

Conclusions

Game-theoretic scheduling assistants are now being used daily to schedule checkpoints, patrols and other security activities by agencies such as LAX police, FAMS and the TSA. Augmenting the game-theoretic framework to handle the fundamental challenge of uncertainty is pivotal to increase the value of such scheduling assistants. In this paper, we have analyzed the robust solutions provided by RECON, a recent model that computes risk-averse strategies for the defender in the presence of execution and observation uncertainty. Our experimental results show that RECON is robust to such noise in expectation as well as simulation, where the performance of existing algorithms can be arbitrarily bad. Additionally, we have extensively analyzed two heuristics, a-RECON and i-RECON, that further speed up the performance of RECON. This research complements other research focused on handling other types of uncertainty such as in payoff and decision making (Kiekintveld, Tambe, and Marecki 2010), and could ultimately be part of a single unified robust algorithm.

Acknowledgements

This research is supported by the United States Department of Homeland Security through Center for Risk and Economic Analysis of Terrorism Events (CREATE).

References

Aghassi, M., and Bertsimas, D. 2006. Robust game theory. *Math. Program.* 107:231–273.

Agmon, N.; Sadov, V.; Kaminka, G. A.; and Kraus, S. 2008. The Impact of Adversarial Knowledge on Adversarial Planning in Perimeter Patrol. In *AAMAS*, volume 1.

Archibald, C., and Shoham, Y. 2009. Modeling billiards games. In *AAMAS*.

Bagwell, K. 1995. Commitment and observability in games. *Games and Economic Behavior* 8:271–280.

Bard, J. F. 2006. *Practical Bilevel Optimization: Algorithms and Applications (Nonconvex Optimization and Its Applications)*. Springer-Verlag New York, Inc.

Basilico, N.; Gatti, N.; and Amigoni, F. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*.

Conitzer, V., and Sandholm, T. 2006. Computing the optimal strategy to commit to. In *ACM EC-06*, 82–90.

Huck, S., and Mller, W. 2000. Perfect versus imperfect observability–an experimental test of Bagwell's result. *Games and Economic Behavior* 31(2):174 – 190.

Iyengar, G. 2005. Robust dynamic programming. *Mathematics of Operation Research* 30.

Jain, M.; Tsai, J.; Pita, J.; Kiekintveld, C.; Rathi, S.; Tambe, M.; and Ordóñez, F. 2010. Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshals Service. *Interfaces* 40:267–290.

Jenelius, E.; Westin, J.; and Holmgren, A. J. 2010. Critical infrastructure protection under imperfect attacker perception. *International Journal of Critical Infrastructure Protection* 3(1):16–26.

Kiekintveld, C.; Jain, M.; Tsai, J.; Pita, J.; Tambe, M.; and Ordóñez, F. 2009. Computing optimal randomized resource allocations for massive security games. In *AAMAS*.

Kiekintveld, C.; Tambe, M.; and Marecki, J. 2010. Robust bayesian methods for stackelberg security games (extended abstract). In *AAMAS Short Paper*.

Kodialam, M., and Lakshman, T. 2003. Detecting network intrusions via sampling: A game theoretic approach. In *INFOCOM*.

Morgan, J., and Vardy, F. 2007. The value of commitment in contests and tournaments when observation is costly. *Games and Economic Behavior* 60(2):326–338.

Pita, J.; Jain, M.; Tambe, M.; Ordóñez, F.; and Kraus, S. 2010. Robust solutions to stackelberg games: Addressing bounded rationality and limited observations in human cognition. *AIJ* 174:1142–1171.

Tsai, J.; Yin, Z.; Kwak, J.; Kempe, D.; Kiekintveld, C.; and Tambe, M. 2010. Urban security: Game-theoretic resource allocation in networked physical domains. In *AAAI*.

van Damme, E., and Hurkens, S. 1997. Games with imperfectly observable commitment. *Games and Economic Behavior* 21(1-2):282 – 308.

von Stengel, B., and Zamir, S. 2004. Leadership with commitment to mixed strategies. Technical Report LSE-CDAM-2004-01, CDAM Research Report.

Yang, R.; Kiekintveld, C.; Ordóñez, F.; Tambe, M.; and John, R. 2011. Improved computational models of human behavior in security games. In *International Conference on Autonomous Agents and Multiagent Systems (Ext. Abstract)*.

Yin, Z.; Korzhyk, D.; Kiekintveld, C.; Conitzer, V.; and Tambe, M. 2010. Stackelberg vs. Nash in security games: interchangeability, equivalence, and uniqueness. In *AAMAS*.

Yin, Z.; Jain, M.; Ordóñez, F.; and Tambe, M. 2011. Risk-averse strategies for security games with execution and observational uncertainty. In *AAAI*.