Scaling-Up Security Games with Boundedly Rational Adversaries: A Cutting-Plane Approach

Rong Yang, Albert Xin Jiang, Milind Tambe
University of Southern California
Los Angeles, USA
{*yangrong,jiangx,tambe*}*@usc.edu*

Fernando Ordóñez
University of Chile
Santiago, Chile
fordon@dii.uchile.cl

Abstract

To improve the current real-world deployments of Stackelberg security games (SSGs), it is critical now to efficiently incorporate models of adversary bounded rationality in large-scale SSGs. Unfortunately, previously proposed branch-and-price approaches fail to scale-up given the non-convexity of such models, as we show with a realization called COCOMO. Therefore, we next present a novel cutting-plane algorithm called BLADE to scale-up SSGs with complex adversary models,with three key novelties: (i) an efficient scalable separation oracle to generate deep cuts; (ii) a heuristic that uses gradient to further improve the cuts; (iii) techniques for quality-efficiency tradeoff.

1 Introduction

Recent deployments of attacker-defender Stackelberg security games (SSGs), such as for the US Federal Air Marshals Service (FAMS) or the US Coast Guard [Tambe, 2011] have led to significant research in integrating richer models of human (adversary) decision-making into SSGs [Pita *et al.*, 2010; Yang *et al.*, 2011]. These *bounded rationality models* [McKelvey and Palfrey, 1995; Camerer, 2011] step beyond assumptions of perfect adversary rationality in many SSG applications. In fact, a recent SSG application, the US Coast Guard's PROTECT system for randomized patrolling, uses one such adversary model: the quantal response (QR) model [McKelvey and Palfrey, 1995; Shieh *et al.*, 2012].

This paper focuses on scaling up SSG algorithms integrated with any of a family of discrete choice models [Train, 2003], an important class of bounded rationality models of adversary decision making, of which QR is an important representative. Unfortunately, algorithms for handling such bounded rationality models in SSGs fail when faced with massive scale. These algorithms [Yang *et al.*, 2012], require explicit enumeration of defender strategies, which is not feasible in massive-scale SSGs such as with the FAMS or the US Coast Guard in bigger ports. Previous work has provided branch-and-price (BnP) [Barnhart *et al.*, 1994] as a key technique to avoid explicit enumeration of defender strategies in SSGs [Jain *et al.*, 2010]; however, how well BnP would handle bounded rationality models is an unknown.

To address this shortcoming in previous work, our first contribution investigates the effectiveness of BnP in SSG algorithms handling bounded rationality. As we illustrate via a BnP algorithm called COCOMO, the non-convexity of the objective function given bounded rationality adversary models creates enormous hurdles in scale-up.

The second and main contribution of this paper is then a new algorithm called BLADE, which for the first time illustrates an efficient realization of the cutting-plane approach in SSGs [Kelley, 1960; Boyd and Vandenberghe, 2008]. The cutting-planes approach iteratively refines the solution space via cuts. Our key hypothesis is that with these cuts, BLADE can successfully exploit the structure of the solution space of defender strategies – generated due to the bounded rationality adversary models in SSGs – whereas BnP approaches are blind to this structure. BLADE is based on three novel ideas. First, we present a separation oracle that can effectively prune the search space via deep cuts. More importantly we show that to handle massive scale SSGs, not only must this separation oracle itself use a secondary oracle but that this two-level hierarchy of oracles is efficient. Second, we provide a novel heuristic to further speed-up BLADE by exploiting the SSG objective function to improve its cuts. Third, BLADE provides a technique for quality-efficiency tradeoff. As we experimentally demonstrate, BLADE is significantly more efficient than COCOMO.

2 Background and Notation

We consider an SSG [Conitzer and Sandholm, 2006; Yin *et al.*, 2010] where a defender has a total of M resources to protect a set of targets $\mathcal{T} = \{1, \ldots, |\mathcal{T}|\}$ from an attacker. Given a target i, the defender receives reward R_i^d if the adversary attacks a target that is covered by the defender; otherwise, the defender receives penalty P_i^d. Correspondingly, the attacker receives penalty P_i^a in the former case and reward R_i^a in the latter. SSGs may be non-zero-sum, but $R_i^d > P_i^d$ and $R_i^a > P_i^a$ for all i.

We denote the j^{th} defender pure strategy as A_j, which is an assignment of all the security resources. A_j is represented as a column vector $A_j = \langle A_{ij} \rangle^T$, where A_{ij} indicates whether target i is covered by A_j. For example, in an SSG with 4 targets and 2 resources, $A_j = \langle 1, 0, 0, 1 \rangle^T$ represents the pure strategy of assigning one resource to target 1 and another to target 4. Let $\mathcal{A} = \{A_j\}$ be the set of feasible assignments

of resources, i.e., the set of defender pure strategies. The defender's mixed-strategy can then be represented as a vector $\boldsymbol{a} = \langle a_j \rangle$, where $a_j \in [0,1]$ is the probability of choosing A_j.

A more compact representation of the defender's mixed-strategy is $\boldsymbol{x} = \langle x_i \rangle$, where $x_i = \sum_{A_j \in \mathcal{A}} a_j A_{ij}$ is the marginal probability that a security resource will be assigned to target i [Kiekintveld *et al.*, 2009]. Given x_i, the expected utility of the defender if target i is attacked is $U_i^d(x_i) = x_i R_i^d + (1 - x_i) P_i^d$ and the adversary's is $U_i^a(x_i) = x_i P_i^a + (1 - x_i) R_i^a$.

Our formulation exploits this compact marginal representation, but must then convert this representation to a mixed strategy over the actual defender pure strategies (i.e., $\langle a_j \rangle$). Such conversion in the presence of resource assignment constraints is NP-hard [Korzhyk *et al.*, 2010]; yet such constraints abound in security problems. For example, when air marshals are scheduled to protect two flights out of 100, not all $\binom{100}{2}$ schedules are feasible. There are *spatio-temporal constraints*: the air marshal's two flights have to be connected, e.g., an air marshal cannot fly from Los Angeles to New York and then from Chicago to Seattle. Furthermore, there might be *user-specified constraints* [An *et al.*, 2010]: FAMS might want to cover 50% of the flights to Chicago. Thus, the defender's optimization problem can be written as follows:

$$\texttt{P1}: \{\max_{\boldsymbol{x}} \sum_{i \in \mathcal{T}} U_i^d(x_i) q_i(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{X}_f \equiv \mathcal{X}_{f_1} \cap \mathcal{X}_{f_2}\}$$

$$\mathcal{X}_{f_1} := \{\boldsymbol{x} \mid \boldsymbol{x} = \sum_{A_j \in \mathcal{A}} a_j A_j, \sum_j a_j = 1, a_j \in [0,1]\} \quad (1)$$

$$\mathcal{X}_{f_2} := \{\boldsymbol{x} \mid B\boldsymbol{x} \leq \boldsymbol{b}\} \quad (2)$$

We denote the objective function in P1 as $F(\boldsymbol{x})$. In $F(\boldsymbol{x})$, $U_i^d(x_i)$ is the defender's expected utility if the adversary chooses target i; and $q_i(\boldsymbol{x})$ is the probability that the adversary chooses target i. $q_i(\boldsymbol{x})$ depends on the model used, e.g., assuming a QR model of the adversary, $q_i(\boldsymbol{x})$ is a logit function of $\boldsymbol{x}$. This leads to a nonlinear fractional optimization problem [Yang *et al.*, 2011], which in general is NP-hard [Vavasis, 1995]. Furthermore, $\mathcal{X}_f$ represents the feasible region of the marginal coverage vector, which is defined by the intersection of $\mathcal{X}_{f_1}$ which encompasses the spatio-temporal constraints, and $\mathcal{X}_{f_2}$ which encompasses the user-specified linear constraints on the marginals. Therefore, $\mathcal{X}_f = \mathcal{X}_{f1} \cap \mathcal{X}_{f2}$.

3 Generalized PASAQ

Before discussing scale-up, we generalize the existing algorithms to solve SSGs integrated with bounded rationality models, as they are specialized to the QR model. More specifically, we generalize PASAQ, the fastest current algorithm to compute optimal defender strategy against a QR adversary model [Yang *et al.*, 2012]. In PASAQ, the objective function of P1 is

$$F(\boldsymbol{x}) = \sum_i q_i(\boldsymbol{x}) U_i^d(x_i) = \sum_i \frac{e^{\lambda U_i^a(x_i)}}{\sum_j e^{\lambda U_j^a(x_j)}} U_i^d(x_i)$$

QR is a representative of a more general form of the discrete choice model [Train, 2003; Goeree *et al.*, 2005] for adversary response as shown in Equation (3). In SSGs, typically $f_i(x_i) \geq 0, \forall x_i \in [0,1]$ is a monotonically decreasing function of x_i, indicating that as the defender's marginal coverage on target i increases, the probability that the adversary chooses this target decreases, e.g., in QR, $f_i(x_i) = e^{\lambda U_i^a(x_i)}$ is an exponentially decreasing function of x_i.

$$q_i(\boldsymbol{x}) = \frac{f_i(x_i)}{\sum_i f_i(x_i)} \quad (3)$$

G-PASAQ generalizes PASAQ to solve P1 with the general form of $q_i(\boldsymbol{x})$ in Equation (3). As with PASAQ, G-PASAQ solves this non-linear fractional optimization problem using binary search. At each step of the binary search it solves a non-convex optimization problem whose objective is a sum of nonlinear functions of marginal variables x_i. Approximating each of these single-variable nonlinear functions as a piecewise-linear function with K segments, the non-convex problem is approximated by the MILP shown in Equation (4) - (9); this MILP solved in each iteration of the binary search.

$$\min_{\boldsymbol{x},z} \sum_{i \in \mathcal{T}} (r - P_i^d)(f_i(0) + \sum_{k=1}^{K} \gamma_{ik} x_{ik}) - \sum_{i \in \mathcal{T}} \alpha_i \sum_{k=1}^{K} \mu_{ik} x_{ik} \quad (4)$$

$$\text{s.t. } 0 \leq x_{ik} \leq 1/K, \quad \forall i, \quad k = 1 \ldots K \quad (5)$$

$$z_{ik}/K \leq x_{ik}, \quad \forall i, \quad k = 1 \ldots K-1 \quad (6)$$

$$x_{i(k+1)} \leq z_{ik}, \quad \forall i, \quad k = 1 \ldots K-1 \quad (7)$$

$$z_{ik} \in \{0,1\}, \quad \forall i, \quad k = 1 \ldots K-1 \quad (8)$$

$$\boldsymbol{x} \in \mathcal{X}_f \quad (9)$$

The objective function in Equation (4) is a piecewise linear approximation of $\sum_{i \in \mathcal{T}} (r - P_i^d) f_i(x_i) - \sum_{i \in \mathcal{T}} \alpha_i x_i f_i(x_i)$ where $\alpha_i = R_i^d - P_i^d$ is a constant, γ_{ik} is the slope of $f_i(x_i)$ in the k^{th} segments and μ_{ik} is the corresponding slope of $x_i f_i(x_i)$. The range of each x_i is divided into K segments, and x_i is replaced by the variables $\{x_{ik}, k = 1 \ldots k\}$ such that $x_i = \sum_{k=1}^{K} x_{ik}$. $\{z_{ik}, k = 1..K\}$ in Equation (6)-(8) are integer variables that decide the particular segment that x_i lies in. For example, assuming $K = 5$, there are 5 possible sets of values for $\{z_{ik}\}$ that satisfy the constraints in Equation (6)-(8). If we set $\{z_{i,1..3} = 1; z_{i,4} = 0\}$, then x_i is in the fourth segment, i.e., $x_i \in [0.6, 0.8]$. Equation (9) defines the feasible regions for $\boldsymbol{x}$. More details are in [Yang *et al.*, 2012].

4 COCOMO– A Branch-and-Price Algorithm

G-PASAQ assumes that the set of defender pure strategies ($\mathcal{A}$) can be explicitly enumerated. In massive SSGs, $\mathcal{A}$ cannot be enumerated; COCOMO (COlumn generation for COmplex adversary MOdels) attempts in such cases to use the branch-and-price approach to scale-up G-PASAQ. COCOMO exploits the fact that the integer variables in G-PASAQ represent the particular piecewise linear segments each marginal x_i belongs to and defines

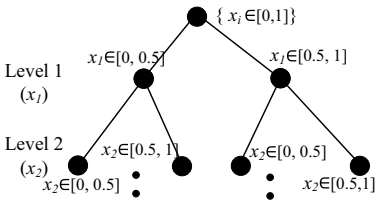


Figure 1: Branching Tree

a branching tree shown in Figure 1. Initially at the root node, all the integer variables are relaxed to be continuous, indicating that none of the x_i are set to any fixed ranges. The i^{th} level in the tree is associated with marginal x_i. If each marginal is divided into K segments, each node has K children. For example, in Figure 1, the two nodes at level 1 are associated with the two possible ranges of marginal x_1: the left node sets $x_1 \in [0, 0.5]$, realized by setting $z_{11} = 0$; the right node sets $x_1 \in [0.5, 1]$, realized by setting $z_{11} = 1$. As we move deeper, more integer variable values are set. The tree has a depth of $|\mathcal{T}|$ and $K^{|\mathcal{T}|}$ nodes in total.

CoCoMo starts from the root node in the branching queue and iterates until the queue is empty. In each iteration, the top node in the branching queue is first branched into a set of children. For each child node, the upper bound (*UB'*) and the lower bound (*LB'*) are estimated. If the two bounds are not close enough, the node is added to the branching queue. CoCoMo keeps a record of the best lower bound solution ($\tilde{LB}$) found so far, and uses that to prune all the unvisited nodes in the branching queue. In the end, the defender strategy associated with this best lower bound is returned as the solution.

Upper Bound Estimation: To generate tighter upper bounds, we run G-Pasaq at each node of CoCoMo, where the values of some variables z_{ik} are set to either 0 or 1 (see Figure 1). We obtain the upper bound by relaxing the rest of the integer variables to be continuous, resulting in an LP called `UpperBound-LP`. `UpperBound-LP` cannot escape the large number of variables a_j and A_j; hence we apply the standard *column generation* technique: we start by solving `UpperBound-LP` with a subset of columns, i.e., defender strategies A_j, and iteratively add more columns with negative *reduced cost*. Let's first rewrite Equation (9) based on the definition of $\mathcal{X}_f$ from Equation (1) and (2).

$$\sum_{k=1..K} x_{ik} - \sum_{A_j \in \mathcal{A}} a_j A_{ij} = 0, \ \forall i \in \mathcal{T} \tag{10}$$

$$\sum_{A_j \in \mathcal{A}} a_j = 1, \quad a_j \geq 0, A_j \in \mathcal{A} \tag{11}$$

$$\sum_{i \in \mathcal{T}} B_{mi} \sum_{k=1..K} x_{ik} \leq b_m, \ \forall m \tag{12}$$

The reduced cost of column A_j is $\omega^T A_j - \rho$, where ω and ρ are the duals of Equation (10) and (11) respectively. Given the optimal duals of the current iteration of `UpperBound-LP`, a separate *Slave* process provides a new column with the minimum reduced cost; the process iterates until convergence.

Slave: Given the spatio-temporal constraints, the *Slave* can often be formulated as a minimum-cost integer flow problem on a polynomial-sized network, e.g., [Jain *et al.*, 2010] provide such a *Slave* formulation with application to the FAMS domain.

Lower Bound Estimation: A subset of the columns will be generated while solving the `UpperBound-LP`. The lower bound of the same node is computed by running G-Pasaq with this subset of columns.

5 Blade– A Cutting-Plane Algorithm

Despite our effort for efficiency in CoCoMo, the need to run column generation at each of the $K^{|\mathcal{T}|}$ nodes ultimately leads to its inefficiency. Blade (Boosted piecewise Linear Approximation of DEfender strategy with arbitrary constraints) uses the cutting-plane approach to scale-up G-Pasaq, and avoids running column generation at each node. Algorithm 1 presents Blade. The *Master* is a *modified version* of `P1` with a relaxed defender strategy space, defined by the set of boundaries $\tilde{H}\boldsymbol{x} \leq \tilde{\boldsymbol{h}}$. In Line (2), $(\tilde{H}, \tilde{\boldsymbol{h}})$ is initialized with the user-specified constraints, $(B, \boldsymbol{b})$. The solution found by the *Master*, i.e., $\tilde{\boldsymbol{x}}$, provides an upper bound (*UB*) of the solution for `P1`. In each iteration, the *Separation Oracle* checks whether or not $\tilde{\boldsymbol{x}} \in \mathcal{X}_f$. If so, the optimal solution of `P1` has been found; otherwise, a new cutting plane $H_l\boldsymbol{x} \leq h_l$ is returned to further restrict the search space in the *Master*. The *Separation Oracle* also returns a feasible solution $\boldsymbol{x}_f$ 'closest' to the infeasible solution $\tilde{\boldsymbol{x}}$, which provides a lower bound ($LB = F(\boldsymbol{x}_f)$) of the solution for `P1`. In Line (9), we improve our lower bound estimation to further speed up the algorithm. The algorithm terminates when *UB* and *LB* are close enough, i.e., $UB - LB \leq \epsilon$.

Algorithm 1: Blade

```
1  Input: {R_i^d, P_i^d, R_i^a, P_i^a}, (B, b), ε;
2  (H̃, h̃) ← (B, b), feasible ← false;
3  UB ← M, LB ← −M;
4  while UB − LB > ε do
5      (UB, x̃) ← Master(H̃, h̃);
6      (feasible, H_l, h_l, x_f) ← SeparationOracle(x̃);
7      H̃ ← H̃ ∪ H_l, h̃ ← h̃ ∪ h_l;
8      if feasible ≠ true then
9          (LB, x_l) ← LowerBoundEstimator(H̃, h̃) ;
10 return x_l;
```

5.1 Master

We first reformulate `P1` by representing its feasible region using the set of bouandries instead of the extreme points:

$$\texttt{P1.1}: \{\max_{\boldsymbol{x}} F(\boldsymbol{x}) \mid H\boldsymbol{x} \leq \boldsymbol{h}; B\boldsymbol{x} \leq \boldsymbol{b}; 0 \leq x_i \leq 1, \forall i \in \mathcal{T}\}$$

H is a N-by-$|\mathcal{T}|$ matrix, where N is the number of linear boundaries of the convex hull. Each row, $H_l\boldsymbol{x} \leq h_l$, represents a linear boundary of $\mathcal{X}_{f1}$. In the presence of user-specified constraints, $B\boldsymbol{x} \leq \boldsymbol{b}$ is added to the boundary set of $\mathcal{X}_f$, as defined in Equation (2). However, we cannot directly solve `P1.1` because H and $\boldsymbol{h}$ are not initially given.

In Blade, the *Master* solves `P1.1` using G-Pasaq with a subset of the boundaries of $\mathcal{X}_f$. More specifically, Equation (9) is rewritten as Equation (12) and (13):

$$\sum_{i \in \mathcal{T}} \tilde{H}_{li} \sum_{k=1..K} x_{ik} \leq \tilde{h}_l, \ \forall l \tag{13}$$

$(\tilde{H}, \tilde{\boldsymbol{h}})$ in Equation (13) represents the subset of the boundaries for $\mathcal{X}_f$. The solution of *Master*, denoted as $\tilde{\boldsymbol{x}}$, then provides an upper bound on the solution of `P1.1`: $F(\tilde{\boldsymbol{x}}) \geq F(\boldsymbol{x}^*)$, where $\boldsymbol{x}^*$ denote the optimal solution of `P1.1`. As the algorithm keeps refining the feasible region by adding new boundaries to the *Master*, this upper bound gets tighter.

Given $\tilde{\boldsymbol{x}}$ as the relaxed solution from the *Master*, we check whether it belongs to $\mathcal{X}_f$. If so, we have found the optimal solution of `P1.1`. Otherwise, we further restrict the feasible region in the *Master* via a cut to separate the current infeasible solution and the original feasible region.

5.2 Separation Oracle

One standard approach for checking feasibility and generating cutting planes is to apply Farkas' Lemma, as in [Papadimitriou and Roughgarden, 2008]. However, the resulting cutting planes are not guaranteed to be ***deep*** cuts that touch the feasible region and therefore eliminate as much of the infeasible region as possible. Instead, we use a norm-minimization approach for the *Separation Oracle* in BLADE, which efficiently checks the feasibility of $\tilde{\boldsymbol{x}}$, and generates a deep cut to separate $\mathcal{X}_f$ from an infeasible $\tilde{\boldsymbol{x}}$. Additionally, our approach finds a feasible point that is closest to $\tilde{\boldsymbol{x}}$, allowing us to compute a lower bound on the optimal objective.

Check Feasibility: The *Separation Oracle* checks the feasibility of $\tilde{\boldsymbol{x}}$ by minimizing its distance to the feasible region. If the minimum distance is 0, $\tilde{\boldsymbol{x}}$ is within the feasible region. We choose 1-norm to measure the distance between $\tilde{\boldsymbol{x}}$ and any feasible point, as 1-norm leads to a Linear Program (LP), which allows the use of column generation to deal with large defender strategy space. We first show the `Min-1-Norm` LP in Equation (14)-(18),

$$\min_{\boldsymbol{a},\boldsymbol{z}} \quad \sum_{i\in\mathcal{T}} z_i \tag{14}$$

$$\text{s.t.} \quad \boldsymbol{z} + A\boldsymbol{a} \geq \tilde{\boldsymbol{x}} \tag{15}$$

$$\boldsymbol{z} - A\boldsymbol{a} \geq -\tilde{\boldsymbol{x}} \tag{16}$$

$$-BA\boldsymbol{a} \geq -\boldsymbol{b} \tag{17}$$

$$\sum_{A_j\in\mathcal{A}} a_j = 1, \quad a_j \geq 0, \ \forall A_j \in \mathcal{A} \tag{18}$$

In the above LP, a marginal coverage is represented by the set of defender pure strategies: $A\boldsymbol{a}$. Constraint (17) and (18) enforces that $A\boldsymbol{a}$ satisfies both the spatio-temporal constraints and the user-specified constraints. The 1-norm distance between the given marginal $\tilde{\boldsymbol{x}}$ and $A\boldsymbol{a}$ is represented by vector $\boldsymbol{z}$. This is obtained by combining Constraints (15) and (16): $-\boldsymbol{z} \leq |A\boldsymbol{a} - \tilde{\boldsymbol{x}}| \leq \boldsymbol{z}$. The objective function minimizes the 1-norm of $\boldsymbol{z}$, therefore the 1-norm distance between $\tilde{\mathrm{x}}$ and any given feasible marginal is minimized.

Lemma 1. *Given a marginal $\tilde{\mathbf{x}}$, let $(\mathbf{z}^*, \mathbf{a}^*)$ be the optimal solution of the corresponding `Min-1-Norm` LP. $\tilde{\mathbf{x}}$ is feasible if and only if $\sum_{i\in\mathcal{T}} z_i^* = 0$. Furthermore, $A\mathbf{a}^*$ provides the feasible marginal with the minimum 1-norm distance to $\tilde{\mathbf{x}}$.*

Generate Cut: If $\tilde{\boldsymbol{x}}$ is infeasible, we need to further restrict the relaxed region in the *Master*. Theoretically, any hyperplane that separates $\tilde{\boldsymbol{x}}$ from the feasible region could be used. In practice, a deep cut is preferred. Let $\boldsymbol{w}$, $\boldsymbol{v}$, $\boldsymbol{g}$ and u be the dual variables of Constraints (15), (16), (17) and (18) respectively; and let $\boldsymbol{y} = \boldsymbol{w} - \boldsymbol{v}$.

Lemma 2. *Given an infeasible marginal $\tilde{\mathbf{x}}$, let $(\mathbf{y}^*, \mathbf{g}^*, u^*)$ be the dual values at the optimal solution of the corresponding `Min-1-Norm` LP. The hyperplane $(\mathbf{y}^*)^T\mathbf{x} - (\mathbf{g}^*)^T\mathbf{b} + u^* = 0$ separates $\tilde{\mathbf{x}}$ and $\mathcal{X}_f$:*

$$(\mathbf{y}^*)^T\tilde{\mathbf{x}} - (\mathbf{g}^*)^T\mathbf{b} + u^* > 0 \tag{19}$$

$$(\mathbf{y}^*)^T\mathbf{x} - (\mathbf{g}^*)^T\mathbf{b} + u^* \leq 0, \quad \forall \mathbf{x} \in \mathcal{X}_f \tag{20}$$

Proof. The dual of the `Min-1-Norm` LP is:

$$\max_{\boldsymbol{y},u,\boldsymbol{g}} \quad \tilde{\boldsymbol{x}}^T\boldsymbol{y} - \boldsymbol{b}^T\boldsymbol{g} + u \tag{21}$$

$$\text{s.t.} \quad A^T\boldsymbol{y} - A^TB^T\boldsymbol{g} + u \leq 0 \tag{22}$$

$$1 \geq \boldsymbol{y} \geq -1, \ \boldsymbol{g} \geq 0 \tag{23}$$

Equation (19) can be proved using LP duality. Since $\tilde{\boldsymbol{x}}$ is infeasible, the minimum of the corresponding `Min-1-Norm` LP is strictly positive. Therefore, the maximum of the dual LP is also strictly positive.

We now prove the contrapositive of Equation (20):

$$(\boldsymbol{y}^*)^T\boldsymbol{x} - (\boldsymbol{g}^*)^T\boldsymbol{b} + u^* > 0 \Rightarrow \boldsymbol{x} \textit{ is not feasible}$$

Given any $\boldsymbol{x}'$, there is a corresponding LP with the same formulation as that in Equation (21)-(23). Let $(\boldsymbol{y}', u', \boldsymbol{g}')$ be the optimal solution of this LP. Note that, $(\boldsymbol{y}^*, u^*, \boldsymbol{g}^*)$ is a feasible solution of this LP. Therefore,

$$(\boldsymbol{y}')^T\boldsymbol{x} - (\boldsymbol{g}')^T\boldsymbol{b} + u' \geq (\boldsymbol{y}^*)^T\boldsymbol{x} - (\boldsymbol{g}^*)^T\boldsymbol{b} + u^* > 0$$

This indicates that the minimum 1-norm distance between $\boldsymbol{x}$ and $\mathcal{X}_f$ is strictly positive. Hence, $\boldsymbol{x}$ is infeasible. □

Lemma 3. *Equation (20) is a* **deep** *cut that touches the feasible convex hull $\mathcal{X}_f$.*

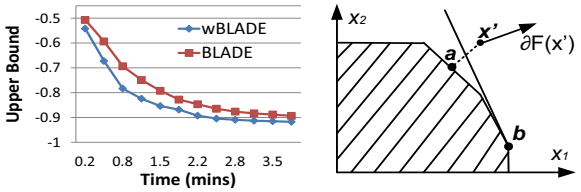
Proof. For simplicity, we consider the cases without user-specified constraints. The cut in Equation (20) then becomes $(\boldsymbol{y}^*)^T\boldsymbol{x} + u^* \leq 0$. Let a_j be the dual of the j^{th} constraint in Equation (22) and $\boldsymbol{a}^* = \langle a_j^* \rangle$ be the dual at the optimal solution of LP in Equation (21)-(23). According to the LP duality, $A\boldsymbol{a}^*$ is the optimal solution of the `Min-1-Norm` LP. Therefore, $A\boldsymbol{a}^*$ is the feasible marginal with the minimum 1-Norm distance to $\tilde{\boldsymbol{x}}$. Furthermore, $\forall a_j^* > 0$, the corresponding constraint in Equation (22) is active, i.e. $(\boldsymbol{y}^*)^T A_j + u^* = 0$. Hence, the extreme point A_j is on the cutting-plane. □

Therefore, by solving either the `Min-1-Norm` LP or its dual LP, the *Separation Oracle* can not only check the feasibility of a given marginal, but also generate a deep cut. We choose to solve the dual LP in Equation (21)-(23), since it gives the constraint directly as shown in Equation (20). However, since in our case the set of the defender's pure strategies is too large to be enumerated, the constraints of the LP cannot be enumerated.We solve the LP using a constraint generation approach, outlined in Algorithm 2. Specifically, we solve the LP in Equation (21)-(23) with a subset of constraints first, and use a *Secondary Oracle* to check whether the relaxed solution violates any of the other constraints.

Secondary Oracle: The secondary oracle is executed at Line (6) in Algorithm 2. If any constraint in Equation (22) is violated, the oracle returns the one that is most violated, i.e., A_l with the most negative value of the LHS of Equation (22); otherwise, we have found the optimal solution of the LP. The secondary oracle is similar to the *Slave* in COCOMO.

Algorithm 2: Separation Oracle

Input: $\{R_i^d, P_i^d, R_i^a, P_i^a\}$, (B, b), $\tilde{\boldsymbol{x}}$, $A^{(0)}$;
$A \leftarrow A^{(0)}$, $A_l \leftarrow A_1$;
while $A_l \neq \emptyset$ **do**
  $A \leftarrow A \cup A_l$;
  $(\boldsymbol{y}^*, u^*, \boldsymbol{g}^*) \leftarrow$ *Solve-Separation-Oracle-LP*(A);
  $A_l \leftarrow$`SecondaryOracle`$(\boldsymbol{y}^*, u^*, \boldsymbol{g}^*)$;
return $(\boldsymbol{y}^*, u^*, \boldsymbol{g}^*)$;



(a) Upper bounds over time (b) Weighted 1-norm

Figure 2: Minimizing weighted 1-norm distance

5.3 wBLADE

The convergence of BLADE depends on how fast the cuts generated by the *Separation Oracle* approximate the feasible set around the optimal solution of `P1.1`. We propose wBLADE, which modifies the *Separation Oracle* by changing the norm used to determine the distance to $\mathcal{X}_f$ for one that takes the objective function into account, to bias the cut generated toward the optimal solution. Formally, given the solution $\tilde{\boldsymbol{x}}$ from the *Master*, instead of searching for the feasible point with the 1-norm distance, which is uniform in all dimensions, we modify the objective function of the `Min-1-Norm` LP in Equation (14) as:

$$\sum_{i \in \mathcal{T}} (\nabla_i F(\tilde{\boldsymbol{x}}) + \xi) z_i \tag{24}$$

where $\nabla_i F(\tilde{\boldsymbol{x}})$ is the gradient of objective function $F(\boldsymbol{x})$ at point $\tilde{\boldsymbol{x}}$ with respect to x_i; ξ is a pre-defined constant to ensure that $\nabla_i F(\tilde{\boldsymbol{x}}) + \xi > 0, \forall i$ so the objective remains a norm. We refer to this modified LP as `Min-Weighted-Norm` LP.

Lemma 4. *A marginal* $\tilde{\mathbf{x}}$ *is feasible if and only if the minimum of the corresponding* `Min-Weighted-Norm` *LP is 0.*

Proof. We already showed that $z_i \geq 0$ represents the absolute difference between $\tilde{\boldsymbol{x}}$ and the feasible point $A\boldsymbol{a}$ on the i^{th} dimension. Combining with $\nabla_i F(\tilde{\boldsymbol{x}}) + \xi > 0, \forall i$, we have $\sum_i (\nabla_i F(\tilde{\boldsymbol{x}}) + \xi) z_i \geq 0$. According to Lemma 1, $\tilde{\boldsymbol{x}}$ is feasible if and only if the minimum of $\sum_i z_i$ is 0. Hence, if there exists $(\boldsymbol{z}^*, \boldsymbol{a}^*)$ such that $\sum_i z_i^* = 0$, we have $\sum_i (\nabla_i F(\tilde{\boldsymbol{x}}) + \xi) z_i^* = 0$; and vice versa. □

To provide some intuition into why tighter bounds can be obtained by solving `Min-Weighted-Norm` LP, we consider the case when $\nabla_i F(\tilde{\boldsymbol{x}}) > 0$ and $\tilde{\boldsymbol{x}} \geq A\boldsymbol{a}$. First we note that these are typical situations in security games, where having more defense resources tends to benefit the defender. This is the case even if the attacker is boundedly rational, as in the quantal response model. Therefore for most values of $\tilde{\boldsymbol{x}}$ the gradient $\nabla_i F(\tilde{\boldsymbol{x}})$ will be positive. As a result, a solution $\tilde{\boldsymbol{x}}$ of the relaxed problem solved by the master will tend to use more resources than what is feasible, i.e., $\tilde{\boldsymbol{x}} \geq A\boldsymbol{a}$. These properties are confirmed in our numerical experiments.

Then, if we have $\nabla_i F(\tilde{\boldsymbol{x}}) > 0$ and $\tilde{\boldsymbol{x}} \geq A\boldsymbol{a}$, the `Min-Weighted-Norm` LP is equivalent to minimizing $\nabla F(\tilde{\boldsymbol{x}}) \cdot (\tilde{\boldsymbol{x}} - A\boldsymbol{a})$ and hence also to maximizing

$$F(\tilde{\boldsymbol{x}}) + \nabla F(\tilde{\boldsymbol{x}})(A\boldsymbol{a} - \tilde{\boldsymbol{x}}) \tag{25}$$

Equation (25) is the first-order Taylor approximation of $F(\boldsymbol{x})$, maximizing which should provide a good lower bound if $\tilde{\boldsymbol{x}}$ is close to the feasible region.

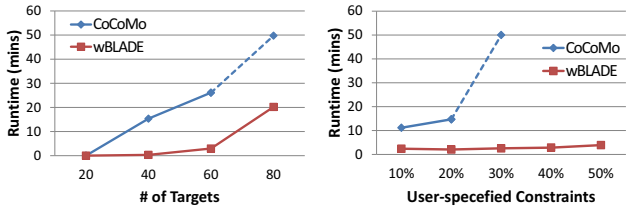
Regarding the cuts generated, since $\nabla F(\tilde{\boldsymbol{x}}) > 0$ we can take $\xi = 0$. In this case the `Min-Weighted-Norm` LP is looking for the projection point in $\mathcal{X}_f$ on the highest level-set perpendicular to $\nabla F(\tilde{\boldsymbol{x}})$. The cut generated, therefore, will be this highest level-set perpendicular to $\nabla F(\tilde{\boldsymbol{x}})$. Since the gradient points in the direction of maximum increase of its function, for sufficiently smooth functions and sufficiently close projection point, points with higher function values than the projection point would be eliminated by this cut. Figure 2(b) demonstrates an example: the shaded polyhedron represents the feasible region $\mathcal{X}_f$. Given the infeasible point $\boldsymbol{x}'$ and its gradient $\nabla F(\boldsymbol{x}')$, the solid line perpendicular to $\nabla F(\boldsymbol{x}')$ approximates the level-set of $F(\boldsymbol{x})$. Therefore, the half-space on the direction of $\nabla F(\boldsymbol{x}')$ of that line would tend to have points with higher value of $F(\boldsymbol{x})$ than the other half-space. This suggests that such a cut would prune more infeasible points with high objectives, leading to a tighter upper bound. Confirming this intuition, Figure 2(a) shows that over 30 random samples the upper-bound decreases faster in wBLADE: x-axis marks iterations in time, y-axis plots the upper bound.

5.4 Quality and Runtime Trade-off

The feasible point returned by the *Separation Oracle* provides a lower bound on `P1.1`. A better lower bound can be achieved by solving a restricted version of `P1.1` (Line 9 in Algorithm 1). This can be done by replacing $\mathcal{X}_{f1}$ with the convex hull formed by the subset of defender pure strategies generated in solving the *Separation Oracle*. These upper and lower bounds allow us to trade off between solution quality and runtime by controlling the threshold ϵ: as soon as $UB - LB \leq \epsilon$ the algorithm returns the feasible solution associated with *LB*, which is guaranteed to be within ϵ of the optimal objective.

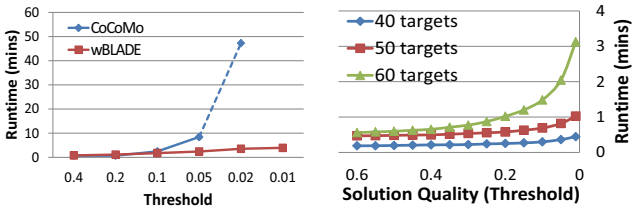
6 Experimental results

In this section, we compare CoCoMo and BLADE assuming two different bounded rationality models. We take FAMS as our example domain. For each setup, we tried 30 game instances. In each game, payoffs R_i^d and R_i^a are random integers from 1 to 10, while P_i^d and P_i^a are random integers from -10 to -1; the feasible schedules for each unit of resources are generated by randomly selecting 2 targets for each schedule (we assume that each air marshal can cover 2 flights on a single trip, similar to [Jain *et al.*, 2010]). In all experiments, the deployment-to-saturation (d:s) ratio is set to 0.5, which is shown to be computationally harder than any other d:s ratio [Jain *et al.*, 2012]. Furthermore, we set the number of piecewise linear segments to be 15 for each $f_i(x_i)$, given that 10



(a) 20% User Constraints,Threshold=0.02

(b) 60 Targets, Threshold=0.02



(c) 60 Targets, 20% User Constraints

(d) Runtime vs Solution Quality

Figure 4: Comparing CoCoMo and BLADE, QR Model

segments provide a sufficiently good approximation [Yang *et al.*, 2012]. The results were obtained using CPLEX v12.2 on a standard 2.8GHz machine with 4GB main memory.

The BEST BLADE Given the two versions of BLADE, one with the non-weighted *Separation Oracle* and another with the weighted *Separation Oracle*, we investigate whether it would be more effective to combine them such that two cuts are generated in each iteration. While this combined version CBLADE could generate more cuts per iteration reducing the total number of iterations, the runtime of each iteration might be longer. Our first set of experiment investigates the efficiency of the three BLADE algorithms. Figure 3 shows the average runtime of these three algorithms with different number of targets. WBLADE achieves the shortest runtime, as shown in Figure 3. Furthermore, although on average CBLADE takes less iterations to converge, it generates more cuts than both BLADE and WBLADE. For example, with 60 targets, CBLADE takes 17 iterations on average to converge, while WBLADE and BLADE take 23 and 29 iterations respectively. However, the total cuts generated by CBLADE is 34 (2 cuts per iteration) which is more than WBLADE and BLADE. Given this result, we will use WBLADE as the representative of the BLADE family in the rest of the experiments.

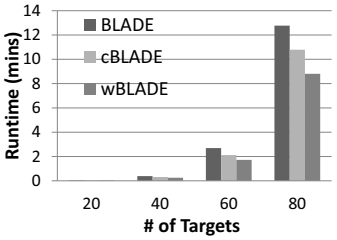
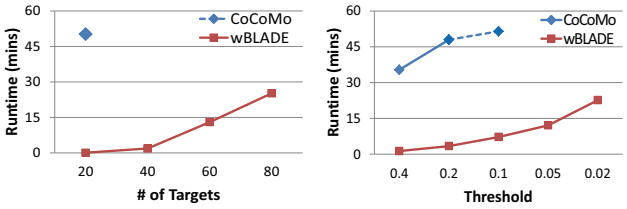


Figure 3: Runtime Comparison of the BLADE family

Quantal Response Model Figure 4(a), 4(b) and 4(c) present the average runtime of CoCoMo and WBLADE assuming a QR model of the adversary with λ parameter set to 0.76, as in [Yang *et al.*, 2011]. In the experiment, we set



(a) 20% User Constraints, Threshold=0.02

(b) 60 Targets, 20% User Constraints

Figure 5: Runtime Comparison, QR-Sigmoid model

50 minutes as the runtime limit. The dashed line in the figures indicates that at that point at least some game instances were not completed within this time limit; and the absence of any markers afterward implies the trend continues. CoCoMo cannot scale to 80 targets, as shown in Figure 4(a). In Figure 4(b), we vary the amount of user-specified constraints (i.e. percentage of the number of targets) while fixing the number of targets to be 60. The constraints were randomly generated inequalities of the marginal coverage vector $\langle \boldsymbol{x} \rangle$. Increasing the amount of user-specified constraints doesn't impact the runtime of WBLADE, but significantly slows down CoCoMo. We then vary the threshold from 0.4 to 0.01 as shown in Figure 4(c). CoCoMo is only able to converge when the threshold is larger than 0.05;in comparison, the runtime of WBLADE slowly increases as the threshold decreases. Thus, WBLADE obtain much better solution quality within significantly shorter amount of time than CoCoMo.

We further investigate the tradeoff between solution quality and runtime of WBLADE and show the result in Figure 4(d). We gradually increase the solution quality by decreasing the threshold under different number of targets, illustrating runtime-quality tradeoff.

A More Complex Bounded Rationality Model We now set $f_i(x_i) = \frac{1}{1+e^{-\lambda_i x_i}}$, a more complex model than QR.We investigate the impact of model complexity on the runtime of CoCoMo and WBLADE. Figure 5(a) and 5(b) display the runtime comparison of CoCoMo and WBLADE. As shown in Figure 5(a), while CoCoMo could not finish running within 50 minutes in any of the settings, the runtime of WBLADE was less than 7 seconds for 20 targets. In Figure 5(b), we show that the runtime of BLADE gradually increases as the threshold decreases. In comparison, CoCoMo is only able to finish running when the threshold is sufficiently large (≥ 0.4) leading to poor solution quality. Thus, as the bounded rationality model becomes more complex, BLADE's advantage over CoCoMo is further magnified.

7 Summary

SSGs and their applications have emerged as a thriving area of research, but scalability, particularly in the presence of complex adversary models remains an open challenge. BLADE is today the only algorithm that can handle large-scale SSGs given bounded-rationality adversary models, and for the first time provides an efficient realization of the cutting-plane approach within SSGs.

Acknowledgments. This research was supported by Army Research Office under Grant W911NF-10-1-0185 and MURI

Grant W911NF-11-1-0332. F. Ordóñez would also like to acknowledge the support of Conicyt, through Grant ACT87.

References

[An *et al.*, 2010] B. An, M. Jain, M. Tambe, and C. Kiekintveld. Mixed-initiative optimization in security games: A preliminary report. *In Proceeding of the AAAI Spring Symposium*, 2010.

[Barnhart *et al.*, 1994] C. Barnhart, E. Johnson, g. Nemhauser, M. Savelsbergh, and P. Vance. Branch and price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1994.

[Boyd and Vandenberghe, 2008] S. Boyd and L. Vandenberghe. Localization and cutting-plane methods. *Lecture Notes*, 2008.

[Camerer, 2011] C.F. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, 2011.

[Conitzer and Sandholm, 2006] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *EC*, pages 82–90, 2006.

[Goeree *et al.*, 2005] J. K. Goeree, C. A. Holt, and T. R. Palfrey. Regular quantal response equilibrium. *Experimental Economics*, 8(4):347–367, December 2005.

[Jain *et al.*, 2010] M. Jain, E. Kardes, C. Kiekintveld, F. Ordonez, and M. Tambe. Security games with arbitrary schedules: A branch and price approach. *In AAAI*, 2010.

[Jain *et al.*, 2012] M. Jain, K. Leyton-Brown, and M. Tambe. The deployment-to-saturation ratio in security games. *In AAAI*, 2012.

[Kelley, 1960] J. E. Kelley. The cutting-plane method for solving convex programs. *The Society for Industrial and Applied Mathematics*, 8(4):703–713, 1960.

[Kiekintveld *et al.*, 2009] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordonez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. *In AAMAS*, 2009.

[Korzhyk *et al.*, 2010] D. Korzhyk, V. Conitzer, and R. Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. *In AAAI*, 2010.

[McKelvey and Palfrey, 1995] R. D. McKelvey and T. R. Palfrey. Quantal response equilibria for normal form games. *Games and Economic Behavior*, 2:6–38, 1995.

[Papadimitriou and Roughgarden, 2008] C.H. Papadimitriou and T. Roughgarden. Computing correlated equilibria in multi-player games. *Journal of the ACM*, 55(3):14, July 2008.

[Pita *et al.*, 2010] J. Pita, M. Jain, F. Ordonez, M. Tambe, and S. Kraus. Solving stackelberg games in the real-world: Addressing bounded rationality and limited observations in human preference models. *Artificial Intelligence Journal*, 174(15):1142–1171, 2010.

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

[Tambe, 2011] M. Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, New York, NY, 2011.

[Train, 2003] K. Train. *Discrete Choice Models with Simulation*. Cambridge University Press, 2003.

[Vavasis, 1995] S. A. Vavasis. Complexity issues in global optimization: a survey. In *Handbook of Global Optimization*, pages 27–41. In R. Horst and P.M. Pardalos, editors, Kluwer, 1995.

[Yang *et al.*, 2011] R. Yang, C. Kiekintveld, F. Ordonez, M. Tambe, and R. John. Improving resource allocation strategy against human adversaries in security games. *In IJCAI*, 2011.

[Yang *et al.*, 2012] R. Yang, F. Ordonez, and M. Tambe. Computing optimal strategy against quantal response in security games. *In Proceedings of AAMAS*, 2012.

[Yin *et al.*, 2010] Z. Yin, D. Korzhyk, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. nash in security games: Interchangeability, equivalence, and uniqueness. *In AAMAS*, 2010.