Multi-Objective Optimization for Security Games

Matthew Brown[1], Bo An[1], Christopher Kiekintveld[2], Fernando Ordóñez[3], Milind Tambe[1]
[1]University of Southern California, Los Angeles, CA, 90089
[2]University of Texas at El Paso, El Paso, TX, 79968
[3]Universidad de Chile, Santiago, Chile
[1]{mattheab,boa,tambe}@usc.edu, [2]cdkiekintveld@utep.edu, [3]fordon@dii.uchile.cl

ABSTRACT

The burgeoning area of security games has focused on real-world domains where security agencies protect critical infrastructure from a diverse set of adaptive adversaries. There are security domains where the payoffs for preventing the different types of adversaries may take different forms (seized money, reduced crime, saved lives, etc) which are not readily comparable. Thus, it can be difficult to know how to weigh the different payoffs when deciding on a security strategy. To address the challenges of these domains, we propose a fundamentally different solution concept, multi-objective security games (MOSG), which combines security games and multi-objective optimization. Instead of a single optimal solution, MOSGs have a set of Pareto optimal (non-dominated) solutions referred to as the Pareto frontier. The Pareto frontier can be generated by solving a sequence of constrained single-objective optimization problems (CSOP), where one objective is selected to be maximized while lower bounds are specified for the other objectives. Our contributions include: (i) an algorithm, Iterative ϵ-Constraints, for generating the sequence of CSOPs; (ii) an exact approach for solving an MILP formulation of a CSOP (which also applies to multi-objective optimization in more general Stackelberg games); (iii) heuristics that achieve speedup by exploiting the structure of security games to further constrain a CSOP; (iv) an approximate approach for solving an algorithmic formulation of a CSOP, increasing the scalability of our approach with quality guarantees. Additional contributions of this paper include proofs on the level of approximation and detailed experimental evaluation of the proposed approaches.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed artificial intelligence—*Intelligent agents*

General Terms

Algorithms, Performance, Security

Keywords

Game Theory, Security, Multi-objective Optimization

1. INTRODUCTION

Game theory is an increasingly important paradigm for modeling security domains which feature complex resource allocation [5, 2]. Security games, a special class of attacker-defender Stackelberg games, are at the heart of several major deployed decision-support applications. Such systems include ARMOR at LAX airport [8], IRIS deployed by the US Federal Air Marshals Service [8], GUARDS developed for the US Transportation Security Administration [1], and PROTECT used in the Port of Boston by the US Coast Guard [1].

In these applications, the defender is trying to maximize a single objective. However, there are domains where the defender has to consider multiple objectives simultaneously. For example, the Los Angeles Sheriff's Department (LASD) needs to protect the city's metro system from ticketless travelers, common criminals, and terrorists.[1] From the perspective of LASD, each one of these attacker types provides a unique threat (lost revenue, property theft, and loss of life). Given this diverse set of threats, selecting a security strategy is a significant challenge as no single strategy can minimize the threat for all attacker types. Thus, tradeoffs must be made and protecting more against one threat may increase the vulnerability to another threat. However, it is not clear how LASD should weigh these threats when determining the security strategy to use. One could attempt to establish methods for converting the different threats into a single metric. However, this process can become convoluted when attempting to compare abstract notions such as safety and security with concrete concepts such as ticket revenue.

Bayesian security games have been used to model domains where the defender is facing multiple attacker types. The threats posed by the different attacker types are weighted according to the relative likelihood of encountering that attacker type. There are three potential factors limiting the use of Bayesian security games: (1) the defender may not have information on the probability distribution over attacker types, (2) it may be impossible or undesirable to directly compare and combine the defender rewards of different security games, and (3) only one solution is given, hiding the trade-offs between the objectives from the end user.

Thus, for many domains, including the LASD metro system, we propose a new game model, multi-objective security games (MOSG), which combines game theory and multi-objective optimization. The threats posed by the attacker types are treated as different objective functions which are not aggregated, thus eliminating the need for a probability distribution over attacker types. Unlike Bayesian security games which have a single optimal solution, MOSGs have a set of Pareto optimal (non-dominated) solutions which is referred to as the Pareto frontier. By presenting the Pareto frontier to the end user, they are able to better understand the structure of their problem as well as the tradeoffs between different security strategies. As a result, end users are able to make a more informed decision on which strategy to enact. For instance, LASD

[1]http://sheriff.lacounty.gov

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

has provided explicit feedback that rather than having a single option handed to them, they would prefer to be presented with a set of alternative strategies from which they can make a final selection.

In this paper, we describe the new MOSG solution concept and provide a set of algorithms for computing Pareto optimal solutions for MOSGs. Our key contributions include (i) Iterative ϵ-Constraints, an algorithm for generating the Pareto frontier for MOSGs by producing and solving a sequence of constrained single-objective optimization problems (CSOP); (ii) an exact approach for solving a mixed-integer linear program (MILP) formulation of a CSOP (which also applies to multi-objective optimization in more general Stackelberg games); (iii) heuristics that exploit the structure of security games to speedup solving CSOPs; and (iv) an approximate approach for solving CSOPs, which greatly increases the scalability of our approach while maintaining quality guarantees. Additionally, we provide analysis of the complexity and completeness for all of our algorithms as well as experimental results.

2. MOTIVATING DOMAINS

As mentioned earlier, LASD must protect the Los Angeles metro system from ticketless travelers, criminals, and terrorists. Each type of perpetrator is distinct and presents a unique set of challenges. Thus, LASD may have different payoffs for preventing the various perpetrators. Targeting ticketless travelers will increase the revenue generated by the metro system as it will encourage passengers to purchase tickets. Pursuing criminals will reduce the amount of vandalism and property thefts, increasing the overall sense of passenger safety. Focusing on terrorists could help to prevent or mitigate the effect of a future terrorist attack, potentially saving lives. LASD has finite resources with which to protect all of the stations in the city. Thus, it is not possible to protect all stations against all perpetrators at all times. Therefore, strategic decisions must be made such as where to allocate security resources and for how long. These allocations should be determined by the amount of benefit they provide to LASD. However, if preventing different perpetrators provides different, incomparable benefits to LASD, it may be unclear how to decide on a strategy. In such situations, a multi-objective security game model could be of use, since the set of Pareto optimal solutions can explore the trade-offs between the different objectives. LASD can then select the solution they feel most comfortable with based on the information they have.

3. MULTI-OBJECTIVE SECURITY GAMES

A multi-objective security game is a multi-player game between a defender and n attackers.[2] The defender tries to prevent attacks by covering targets $T = \{t_1, t_2, \ldots, t_{|T|}\}$ using m identical resources which can be distributed in a continuous fashion amongst the targets. The defender's strategy can be represented as a coverage vector $\mathbf{c} \in C$ where c_t is the amount of coverage placed on target t and represents the probability of the defender successfully preventing any attack on t [9]. $C = \{\langle c_t \rangle | 0 \leq c_t \leq 1, \sum_{t \in T} c_t \leq m\}$ is the defender's strategy space. The attacker i's mixed strategy $\mathbf{a}_i = \langle a_i^t \rangle$ is a vector where a_i^t is the probability of attacking t.

U defines the payoff structure for an MOSG, with U_i defining the payoffs for the security game played between the defender and attacker i. $U_i^{c,d}(t)$ is the defender's utility if t is chosen by attacker i and is fully covered by a defender resource. If t is not covered, the defender's penalty is $U_i^{u,d}(t)$. The attacker's utility is denoted similarly by $U_i^{c,a}(t)$ and $U_i^{u,a}(t)$. A property of security games is that $U_i^{c,d}(t) > U_i^{u,d}(t)$ and $U_i^{u,a}(t) > U_i^{c,a}(t)$ which means that placing more coverage on a target is always beneficial for the defender and disadvantageous for the attacker [9]. For a strategy profile $\langle \mathbf{c}, \mathbf{a}_i \rangle$ for the game between the defender and attacker i, the expected utilities for both agents are given by:

$$U_i^d(\mathbf{c}, \mathbf{a}_i) = \sum_{t \in T} a_i^t U_i^d(c_t, t), \quad U_i^a(\mathbf{c}, \mathbf{a}_i) = \sum_{t \in T} a_t U_i^a(c_t, t)$$

where $U_i^d(c_t, t) = c_t U_i^{c,d}(t) + (1 - c_t) U_i^{u,d}(t)$ and $U_i^a(c_t, t) = c_t U_i^{c,a}(t) + (1 - c_t) U_i^{u,d}(t)$ are the payoff received by the defender and attacker i, respectively, if target t is attacked and is covered with c_t resources.

The standard solution concept for a two-player Stackelberg game is Strong Stackelberg Equilibrium (SSE) [14], in which the defender selects an optimal strategy based on the assumption that the attacker will choose an optimal response, breaking ties in favor of the defender. We denote $U_i^d(\mathbf{c})$ and $U_i^a(\mathbf{c})$ as the payoff received by the defender and attacker i, respectively, when the defender uses the coverage vector $\mathbf{c}$ and attacker i attacks the best target while breaking ties in favor of the defender.

With multiple attackers, the defender's utility (objective) space can be represented as a vector $U^d(\mathbf{c}) = \langle U_i^d(\mathbf{c}) \rangle$. An MOSG defines a multi-objective optimization problem:

$$\max_{\mathbf{c} \in C} \left(U_1^d(\mathbf{c}), \ldots, U_n^d(\mathbf{c}) \right)$$

Solving such multi-objective optimization problems is a fundamentally different task than solving a single-objective optimization problem. With multiple objectives functions there exist tradeoffs between the different objectives such that increasing the value of one objective decreases the value of at least one other objective. Thus for multi-objective optimization, the traditional concept of optimality is replaced by Pareto optimality.

DEFINITION 1. *(Dominance). A coverage vector $\mathbf{c} \in C$ is said to **dominate** $\mathbf{c}' \in C$ if $U_i^d(\mathbf{c}) \geq U_i^d(\mathbf{c}')$ for all $i = 1, \ldots, n$ and $U_i^d(\mathbf{c}) > U_i^d(\mathbf{c}')$ for at least one index i.*

DEFINITION 2. *(Pareto Optimality) A coverage vector $\mathbf{c} \in C$ is **Pareto optimal** if there is no other $\mathbf{c}' \in C$ that dominates $\mathbf{c}$. The set of non-dominated coverage vectors is called **Pareto optimal solutions** C^* and the corresponding set of objective vectors $\Omega = \{U^d(\mathbf{c}) | \mathbf{c} \in C^*\}$ is called the **Pareto frontier**.*

This paper gives algorithms to find Pareto optimal solutions in MOSGs. If there are a finite number of Pareto optimal solutions, it is preferable to generate all of them for the end-user. If there are an infinite number of Pareto optimal solutions, it is impossible to generate all the Pareto optimal solutions. In this case, it is necessary to generate a subset of Pareto optimal solutions that can approximate the true Pareto frontier with quality guarantees. The methods we present in this paper are a starting point for further analysis and additional preference elicitation from end users, all of which depends on fast approaches for generating the Pareto frontier.

4. RELATED WORK

MOSGs build on security games and multi-objective optimization. We have already reviewed (in Section 1) the relationship of MOSGs to previous work in security games and in particular Bayesian security games. In this section, we primarily review the research on multi-objective optimization. There are three representative approaches for generating the Pareto frontier in multi-objective optimization problems. Weighted summation [4], where

[2]The defender does actually face multiple attackers of different types, however, these attackers are not coordinated and hence the problem we address is different than in [10].

the objective functions are assigned weights and aggregated, producing a single Pareto optimal solution. The Pareto frontier can then be explored by sampling different weights. Another approach is multi-objective evolutionary algorithms (MOEA) [6]. Evolutionary approaches such as NSGA-II [7] are capable of generating multiple approximate solutions in each iteration. However, due to their stochastic nature, both weighted summation and MOEA cannot bound the level of approximation for the generated Pareto frontier. This lack of solution quality guarantees is unacceptable for security domains on which we are focused.

The third approach is the ϵ-constraint method in which the Pareto frontier is generated by solving a sequence of CSOPs. One objective is selected as the primary objective to be maximized while lower bound constraints are added for the secondary objectives. The original ϵ-constraint method [4] discretizes the objective space and solves a CSOP for each grid point. This approach is computationally expensive since it exhaustively searches the objective space of secondary objectives. There has been work to improve upon the original ϵ-constraint method. [11] proposes an adaptive technique for constraint variation that leverages information from solutions of previous CSOPs. However, this method requires solving $\mathcal{O}(k^{n-1})$ CSOPs, where k is the number of solutions in the Pareto frontier. Another approach, the augmented ϵ-constraint method [12] reduces computation by using infeasibility information from previous CSOPs. However, this approach only returns a predefined number of points and thus cannot bound the level of approximation for the Pareto frontier. Our approach for solving an MOSG builds upon the basic idea of the ϵ-constraint method. Security domains demand *both* efficiency as well as quality guarantees when providing decision support. Our approach only needs to solve $\mathcal{O}(nk)$ CSOPs and can provide approximation bounds.

5. ITERATIVE ϵ-CONSTRAINTS

The ϵ-constraint method formulates a CSOP for a given set of constraints $\mathbf{b}$, producing a single Pareto optimal solution. The Pareto frontier is then generated by solving multiple CSOPs produced by modifying the constraints in $\mathbf{b}$. This section presents Iterative ϵ-Constraints, an algorithm for systematically generating a sequence of CSOPs for an MOSG. These CSOPs can then be passed to a solver Φ to return solutions to the MOSG. The next two sections present 1) an exact MILP approach (Section 6) which can guarantee that each solution is Pareto optimal and 2) a faster approximate approach (Section 7) for solving CSOPs.

5.1 Algorithm for Generating CSOPs

Iterative ϵ-Constraints uses the following four key ideas: 1) The Pareto frontier for an MOSG can be found by solving a sequence of CSOPs. For each CSOP, $U_1^d(\mathbf{c})$ is selected as the primary objective, which will be maximized. Lower bound constraints $\mathbf{b}$ are then added for the secondary objectives $U_2^d(\mathbf{c}), \ldots, U_n^d(\mathbf{c})$. 2) The sequence of CSOPs are iteratively generated by exploiting previous Pareto optimal solutions and applying Pareto dominance. 3) It is possible for a CSOP to have multiple coverage vectors $\mathbf{c}$ that maximize $U_1^d(\mathbf{c})$ and satisfy $\mathbf{b}$. Thus, lexicographic maximization is used to ensure that CSOP solver Φ only returns Pareto optimal solutions. 4) It may be impractical (even impossible) to generate all Pareto optimal points if the frontier contains a large number of points, e.g., the frontier is continuous. Therefore, a parameter ϵ is used to discretize the objective space, trading off solution efficiency versus the degree of approximation in the generated Pareto frontier.

We now present a simple MOSG example with two objectives and $\epsilon = 5$. Figure 5.1 shows the objective space for the problem as well as several points representing the objective vectors for

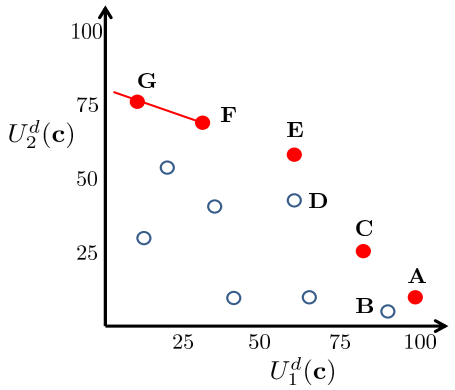

Figure 1: Pareto Frontier for a Bi-Objective MOSG

different defender coverage vectors. In this problem, U_1^d will be maximized while b_2 constrains U_2^d. The initial CSOP is unconstrained (i.e., $b_2 = -\infty$), thus the solver Φ will maximize U_1^d and return solution *A* = (100,10). Based on this result, we know that any point $\mathbf{v} = \{v_1, v_2\}$ (e.g., *B*) is not Pareto optimal if $v_2 < 10$, as it would be dominated by *A*. We then generate a new CSOP, updating the bound to $b_2 = 10 + \epsilon$. Solving this CSOP with Φ produces solution *C*=(80, 25) which can be used to generate another CSOP with $b_2 = 25 + \epsilon$. Both *D*=(60,40) and *E*=(60,60) satisfy b_2 but only *E* is Pareto optimal. Lexicographic maximization ensures that only *E* is returned and dominated solutions are avoided (details in Section 6). The method then updates $b_2 = 60 + \epsilon$ and Φ returns *F*=(30,70), which is part of a continuous region of the Pareto frontier from $U_2^d = 70$ to $U_2^d = 78$. The parameter ϵ causes the method to select a subset of the Pareto optimal points in this continuous region. In particular this example returns *G*=(10,75) and in the next iteration ($b_2 = 80$) finds that the CSOP is infeasible and terminates. The algorithm returns a Pareto frontier of *A*, *C*, *E*, *F*, and *G*.

Algorithm 1 systematically updates a set of lower bound constraints $\mathbf{b}$ to generate the sequence of CSOPs. Each time we solve a CSOP, a portion of the $n-1$ dimensional space formed by the secondary objectives is marked as searched with the rest divided into $n-1$ subregions (by updating $\mathbf{b}$ for each secondary objective). These $n-1$ subregions are then recursively searched by solving CSOPs with updated bounds. This systematic search forms a branch and bound search tree with a branching factor of $n-1$. As the depth of the tree increases, the CSOPs are more constrained, eventually becoming infeasible. If a CSOP is found to be infeasible, no child CSOPs are generated because they are guaranteed to be infeasible as well. The algorithm terminates when the entire secondary objective space has been searched.

Algorithm 1: Iterative-ϵ-Constraints($\mathbf{b} = \{b_2, \ldots, b_n\}$)

```
1  if b ∉ previousBoundsList then
2      append(previousBoundsList, b) ;
3      c ← Φ(b) ;
4      if c is a feasible solution then
5          v ← {U_1^d(c), ..., U_n^d(c)};
6          for 2 ≤ i ≤ n do
7              b' ← b;
8              b'_i ← v_i + ε ;
9              if b' ≱ s, ∀s ∈ infeasibleBoundsList then
10                 Iterative-ε-Constraints(b') ;
11     else append(infeasibleBoundsList, b) ;
```

Two modifications are made to improve the efficiency of the al-

gorithm. 1) Prevent redundant computation resulting from multiple nodes having an identical set of lower bound constraints by recording the lower bound constraints for all previous CSOPs in a list called $previousBoundsList$. 2) Prevent the solving of CSOPs which are known to be infeasible based on previous CSOPs by recording the lower bound constraints for all infeasible CSOPs in a list called $infeasibleBoundsList$.

5.2 Approximation Analysis

Assume the full Pareto frontier is Ω and the objective space of the solutions found by the Iterative ϵ-Constraints method is Ω_ϵ.

THEOREM 3. *Solutions in Ω_ϵ are non-dominated, i.e., $\Omega_\epsilon \subseteq \Omega$.*

PROOF. Let $\mathbf{c}^*$ be the coverage vector such that $U^d(\mathbf{c}^*) \in \Omega_\epsilon$ and assume that it is dominated by a solution from a coverage vector $\bar{\mathbf{c}}$. That means $U_i^d(\bar{\mathbf{c}}) \geq U_i^d(\mathbf{c}^*)$ for all $i = 1, \ldots, n$ and for some j, $U_j^d(\bar{\mathbf{c}}) > U_j^d(\mathbf{c}^*)$. This means that $\bar{\mathbf{c}}$ was a feasible solution for the CSOP for which $\mathbf{c}^*$ was found to be optimal. Furthermore, the first time the objectives differ, the solution $\bar{\mathbf{c}}$ is better and should have been selected in the lexicographic maximization process. Therefore $\mathbf{c}^* \notin \Omega_\epsilon$ which is a contradiction. □

Given the approximation introduced by ϵ, one immediate question is to characterize the efficiency loss. Here we define a bound to measure the largest efficiency loss:

$$\rho(\epsilon) = \max_{\mathbf{v} \in \Omega \setminus \Omega_\epsilon} \min_{\mathbf{v}' \in \Omega_\epsilon} \max_{1 \leq i \leq n} (v_i - v_i')$$

This approximation measure is widely used in multi-objective optimization (e.g. [3]). It computes the maximum distance between any point $\mathbf{v} \in \Omega \setminus \Omega_\epsilon$ on the frontier to its "closest" point $\mathbf{v}' \in \Omega_\epsilon$ computed by our algorithm. The distance between two points is the maximum difference of different objectives.

THEOREM 4. $\rho(\epsilon) \leq \epsilon$.

PROOF. It suffices to prove this theorem by showing that for any $\mathbf{v} \in \Omega \setminus \Omega_\epsilon$, there is at least one point $\mathbf{v}' \in \Omega_\epsilon$ such that $v_1' \geq v_1$ and $v_i' \geq v_i - \epsilon$ for $i > 1$.

Algorithm 2 recreates the sequence of CSOP problems generated by Iterative ϵ-Constraints but ensuring that the bound $\mathbf{b} \leq \mathbf{v}$ throughout. Since Algorithm 2 terminates when we do not update $\mathbf{b}$, this means that $v_i' + \epsilon > v_i$ for all $i > 1$. Summarizing, the final solution $\mathbf{b}$ and $\mathbf{v}' = U^d(\Phi(\mathbf{b}))$ satisfy $\mathbf{b} \leq \mathbf{v}$ and $v_i' > v_i - \epsilon$ for all $i > 1$. Since $\mathbf{v}$ is feasible for the CSOP with bound $\mathbf{b}$, but $\Phi(\mathbf{b}) = \mathbf{v}' \neq \mathbf{v}$ then $v_1' \geq v_1$. □

Given Theorem 4, the maximum distance for every objective between any missed Pareto optimal point and the closest computed Pareto optimal point is bounded by ϵ. Therefore, as ϵ approaches 0, the generated Pareto frontier approaches the complete Pareto frontier in the measure $\rho(\epsilon)$. For example if there are k discrete solutions in the Pareto frontier and the smallest distance between any two is δ then setting $\epsilon = \delta/2$ will make $\Omega_\epsilon = \Omega$. In this case, since each solution corresponds to a non-leaf node in our search tree, the number of leaf nodes is no more than $(n-1)k$. Thus our algorithm will solve at most $\mathcal{O}(nk)$ CSOPs.

6. MILP APPROACH

In Section 5, we introduced a high level search algorithm for generating the Pareto frontier by producing a sequence of CSOPs. In this section we present an exact approach for defining and solving a mixed-integer linear program (MILP) formulation of a CSOP for MOSGs. We then go on to show how heuristics that exploit the structure and properties of security games can be used to improve the efficiency of our MILP formulation.

Algorithm 2: For $\mathbf{v} \in \Omega \setminus \Omega_\epsilon$, find $\mathbf{v}' \in \Omega_\epsilon$ satisfying $v_1' \geq v_1$ and $v_i' \geq v_i - \epsilon$ for $i > 1$

```
Let b be the constraints in the root node, i.e., b_i = −∞ for i > 1 ;
repeat
    c ← Φ(b), v' ← U^d(c), b' ← b;
    for each objective i > 1 do
        if v'_i + ε ≤ v_i then
            b_i ← v'_i + ε ;
            break;
until b = b';
return Φ(b) ;
```

$$\begin{aligned}
&\max && d_\lambda && (1)\\
&1 \leq j \leq n, \forall t \in T: && d_j - U_j^d(c_t, t) \leq M(1 - a_j^t) && (2)\\
&1 \leq j \leq n, \forall t \in T: && 0 \leq k_j - U_j^a(c_t, t) \leq M(1 - a_j^t) && (3)\\
&1 \leq j < \lambda: && d_j = d_j^* && (4)\\
&\lambda < j \leq n: && d_j \geq b_j && (5)\\
&1 \leq j \leq n, \forall t \in T: && a_j^t \in \{0, 1\} && (6)\\
&\forall j \in A: && \textstyle\sum_{t \in T} a_j^t = 1 && (7)\\
&\forall t \in T: && 0 \leq c_t \leq 1 && (8)\\
& && \textstyle\sum_{t \in T} c_t \leq m && (9)
\end{aligned}$$

Figure 2: Lexicographic MILP Formulation for a CSOP

6.1 Exact MILP Method

As stated in Section 5, to ensure Pareto optimality of solutions lexicographic maximization is required to sequentially maximizing all the objective functions. Thus, for each CSOP we must solve n MILPs in the worst case where each MILP is used to maximize one objective. For the λ^{th} MILP in the sequence, the objective is to maximize the variable d_λ, which represents the defender's payoff for security game λ. This MILP is constrained by having to maintain the previously maximized values d_j^* for $1 \leq j < \lambda$ as well as satisfy lower bound constraints b_k for $\lambda < k \leq n$.

We present our MILP formulation for a CSOP for MOSGs in Figure 2. This is similar to the MILP formulations for security games presented in [9] and elsewhere with the exceptions of Equations (4) and (5). Equation (1) is the objective function, which maximizes the defender's payoff for objective λ, d_λ. Equation (2) defines the defender's payoff. Equation (3) defines the optimal response for attacker j. Equation (4) constrains the feasible region to solutions that maintain the values of objectives maximized in previous iterations of lexicographic maximization. Equation (5) guarantees that the lower bound constraints in $\mathbf{b}$ will be satisfied for all objectives which have yet to be optimized.

If a mixed strategy is optimal for the attacker, then so are all the pure strategies in the support of that mixed strategy. Thus, we only consider the pure strategies of the attacker [13]. Equations (6) and (7) constrain attackers to pure strategies that attack a single target. Equations (8) and (9) specify the feasible defender strategy space.

Once the MILP has been formulated, it can be solved using an optimization software package such as CPLEX. It is possible to increase the efficiency of the MILP formulation by using heuristics to constrain the decision variables. A simple example of a general heuristic which can be used to achieve speedup is placing an upper bound on the defender's payoff for the primary objective. Assume d_1 is the defender's payoff for the primary objective in the parent CSOP and d_1' is the defender's payoff for the primary objective in

Variable	Definition	Dimension
λ	Current Objective	–
m	Number of Defender Resources	–
n	Number of Attacker Types	–
Z	Huge Positive Constant	–
T	Set of Targets	$\|T\|$
$\mathbf{a}$	Attacker Coverage a_j^t	$n \times \|T\|$
$\mathbf{b}$	Objective Bounds b_j	$(n-1) \times 1$
$\mathbf{c}$	Defender Coverage c_t	$\|T\| \times 1$
$\mathbf{d}$	Defender Payoff d_j	$n \times 1$
$\mathbf{d}^*$	Maximized Defender Payoff d_j^*	$n \times 1$
$\mathbf{k}$	Attacker Payoff k_j	$n \times 1$
U^d	Defender Payoff Structure $U_j^d(c_t, t)$	$n \times \|T\|$
U^a	Attacker Payoff Structure $U_j^a(c_t, t)$	$n \times \|T\|$

Figure 3: MILP Formulation Definitions

the child CSOP. As each CSOP is a maximization problem, it must hold that $d_1 \geq d_1'$ because the child CSOP is more constrained than the parent CSOP. Thus, the value of d_1 can be passed to the child CSOP to be used as an upper bound on the objective function.

As noted earlier, this MILP is a slight variation of the optimization problem formulated in [9] for security games. The same variations can be made to more generic Stackelberg games, such as those used for DOBSS [13], giving a formulation for multi-objective Stackelberg games in general.

6.2 Exploiting Game Structures

In addition to placing bounds on the defender payoff, it is possible to constrain the defender coverage in order to improve the efficiency of our MILP formulation. Thus, we introduce an approach for translating constraints on defender payoff into constraints on defender coverage. This approach, ORIGAMI-M, achieves this translation by computing the minimum coverage needed to satisfy a set of lower bound constraints $\mathbf{b}$ such that $U_i^d(\mathbf{c}) \geq b_i$ for $1 \leq i \leq n$. This minimum coverage is then added to the MILP in Figure 2 as constraints on the variable $\mathbf{c}$, reducing the feasible region and leading to significant speedup as verified in experiments.

ORIGAMI-M is a modified version of the ORIGAMI algorithm [9] and borrows many of its key concepts. At a high level, ORIGAMI-M starts off with an empty defender coverage vector $\mathbf{c}$, a set of lower bound constraints $\mathbf{b}$, and m defender resources. We try to compute a coverage $\mathbf{c}$ which uses the minimum defender resources to satisfy constraints $\mathbf{b}$. If a constraint b_i is violated, i.e., $U_i^d(\mathbf{c}) < b_i$, ORIGAMI-M updates $\mathbf{c}$ by computing the minimum additional coverage necessary to satisfy b_i. Since we focus on satisfying the constraint on one objective at a time, the constraints for objectives that were satisfied in previous iterations may become unsatisfied again. The reason is that additional coverage may be added to the target that was attacked by this attacker type, causing it to become less attractive relative to other alternatives for the attacker, and possibly reducing the defender's payoff by changing the target that is attacked. Therefore, the constraints in $\mathbf{b}$ must be checked repeatedly until quiescence (no chances are made to $\mathbf{c}$ for any b_i). If all m resources are exhausted before $\mathbf{b}$ is satisfied, then the CSOP is infeasible.

The process for calculating minimum coverage for a single constraint b_i is built on two properties of security games [9]: (1) the attacker chooses the optimal target; (2) the attacker breaks ties in favor of the defender. The set of optimal targets for attacker i for coverage $\mathbf{c}$ is referred to as the attack set, $\Gamma_i(\mathbf{c})$. Accordingly, adding coverage on target $t \notin \Gamma_i$ does not affect the attacker i's strategy or payoff. Thus, if $\mathbf{c}$ does not satisfy b_i, we only consider adding coverage to targets in Γ_i. Γ_i can be expanded by increasing coverage such that the payoff for each target in Γ_i is equivalent to the payoff for the next most optimal target. Adding an additional target to the attack set cannot hurt the defender since the defender receives the optimal payoff among targets in the attack set.

Algorithm 3: ORIGAMI-M($\mathbf{b}$)

```
1  c ← empty coverage vector ;
2  while b_i > U_i^d(c) for some bound b_i do
3      sort targets T in decreasing order of value by U_i^a(c_t, t);
4      left ← m − Σ_{t∈T} c_t, next ← 2;
5      while next ≤ |T| do
6          addedCov[t] ← empty coverage vector;
7          if max_{1≤t<next} U_i^{c,a}(t) > U_i^a(c_next, t_next) then
8              x ← max_{1≤t<next} U_i^{c,a}(t);
9              noninducibleNextTarget ← true;
10         else
11             x ← U_i^a(c_next, t_next);
12         for 1 ≤ t < next do
13             addedCov[t] ← (x − U_i^{u,a}(t)) / (U_i^{c,a}(t) − U_i^{u,a}(t)) − c_t;
14         if Σ_{t∈T} addedCov[t] > left then
15             resourcesExceeded ← true;
16             ratio[t] ← 1 / (U_i^{u,a}(t) − U_i^{c,a}(t)), ∀1 ≤ t < next;
17             addedCov[t] = ratio[t]·left / Σ_{1≤t≤next} ratio[t], ∀1 ≤ t < next;
18         if U_i^d(c + addedCov) ≥ b_i then
19             c' ← MIN-COV(i, c, b);
20             if c' ≠ null then
21                 c ← c'
22             break;
23         else if resourcesExceeded ∨ noninducibleNextTarget then
24             return infeasible;
25         else
26             c_t += addedCov[t], ∀t ∈ T;
27             left −= Σ_{t∈T} addedCov[t];
28             next++;
29     if next = |T| + 1 then
30         if left > 0 then
31             c ← MIN-COV(i, c, b);
32             if c = null then
33                 return infeasible;
34         else
35             return infeasible;
36 return c ;
```

The idea for ORIGAMI-M is to expand the attack set Γ_i until b_i is satisfied. The order in which the targets are added to Γ_i is by decreasing value of $U_i^a(c_t, t)$. Sorting these values, so that $U_i^a(c_1, t_1) \geq U_i^a(c_2, t_2) \geq \cdots \geq U_i^a(c_{|T|}, t_{|T|})$, we have that $\Gamma_i(\mathbf{c})$ starts only with target t_1. Assume that the attack set includes the first q targets. To add the next target, the attacker's payoff for all targets in Γ_i must be reduced to $U_i^a(c_{q+1}, t_{q+1})$ (Line 11). However, it might not be possible to do this. Once a target t is fully covered by the defender, there is no way to decrease the attacker's payoff below $U_i^{c,a}(t)$. Thus, if $\max_{1 \leq t \leq q} U_i^{c,a}(t) > U_i^a(c_{q+1}, t_{q+1})$ (Line 7), then it is impossible to induce the adversary i to attack target t_{q+1}. In that case, we must reduce the attacker's payoff for targets in the attack set to $\max_{1 \leq t \leq q} U_i^{c,a}(t)$ (Line 8). Then for each target $t \in \Gamma_i$, we compute the amount of additional coverage, $addCov[t]$, necessary to reach the required attacker payoff (Line 13). If the total amount of additional coverage exceeds the amount of remaining coverage, then $addedCov$ is recomputed and each target in the attack set is assigned ratio of the remaining coverage so to maintain the attack set (Line 17). There is then a check to see

if $\mathbf{c} + addedCov$ satisfies b_i (Line 18). If b_i is still not satisfied, then the coverage $\mathbf{c}$ is updated to include $addedCov$ (Line 26) and the process is repeated for the next target (Line 28).

Algorithm 4: MIN-COV$(i, \mathbf{c}, \mathbf{b})$

```
1  Input: Game index i, initial coverage c, lower bound b;
2  c* ← null;
3  minResources ← m;
4  foreach t' ∈ Γ_i(c) do
5      c' ← c ;
6      c'_{t'} = (b_i − U_i^{u,a}(t')) / (U_i^{c,a}(t') − U_i^{u,a}(t'));
7      foreach t ∈ T \ {t'} do
8          if U_i^a(c'_t, t) > U_i^a(c'_{t'}, t') then
9              c'_t = (U_i^a(c'_{t'}, t') − U_i^{u,a}(t)) / (U_i^{c,a}(t) − U_i^{u,a}(t));
10     if U_i^d(c') ≥ b_i and Σ_{t∈T} c'_t ≤ minResources then
11         c* ← c';
12         minResources ← Σ_{t∈T} c'_t ;
13 return c*
```

Then if $\mathbf{c} + addedCov$ expands Γ_i and exceeds b_i, it may be possible to use less defender resources and still satisfy b_i. Thus we use the algorithm MIN-COV to compute, $\forall t' \in \Gamma_i$, the amount of coverage needed to induce an attack on t' which yields a defender payoff of b_i. For each t', MIN-COV generates a defender coverage vector $\mathbf{c}'$, which is initialized to the current coverage $\mathbf{c}$. Coverage $c'_{t'}$ is updated such that the defender payoff for t' is b_i, yielding an attacker payoff $U_i^a(c'_{t'}, t')$ (Line 6). The coverage for every other target $t \in T \setminus \{t'\}$ is updated, if needed, to ensure that t' remains in Γ_i, i.e. $U_i^a(c'_{t'}, t') \geq U_i^a(c'_t, t)$ (Line 9). After this process, $\mathbf{c}'$ is guaranteed to satisfy b_i. From the set of defender coverage vectors, MIN-COV returns the $\mathbf{c}'$ which uses the least amount of defender resources. If while computing the additional coverage to added, either Γ_i is the set of all targets or all m security resources are exhausted, then both b_i and the CSOP are infeasible.

If $\mathbf{b}$ is satisfiable, ORIGAMI-M will return the minimum coverage vector $\mathbf{c}^*$ that satisfies $\mathbf{b}$. This coverage vector can be used to replace Equation (8) with $c_t^* \leq c_t \leq 1$.

7. ORIGAMI-A

In the previous section, we showed heuristics to improve the efficiency of our MILP approach. However, solving MILPs, even when constrained, is computationally expensive. Thus, we present ORIGAMI-A, an extension to ORIGAMI-M which eliminates the computational overhead of MILPs for solving CSOPs. The key idea of ORIGAMI-A is to translate a CSOP into a feasibility problem which can be solved using ORIGAMI-M. We then generate a series of these feasibility problems using binary search in order to approximate the optimal solution to the CSOP. As a result, this algorithmic approach is much more efficient.

ORIGAMI-M computes the minimum coverage vector necessary to satisfy a set of lower bound constraints $\mathbf{b}$. As our MILP approach is an optimization problem, lower bounds are specified for the secondary objectives but not the primary objective. We can convert this optimization problem into a feasibility problem by creating a new set of lower bounds constraints $\mathbf{b}^+$ by adding a lower bound constraint b_1^+ for the primary objective to the constraints $\mathbf{b}$. We set $b_1^+ = \min_{t \in T} U_1^{u,d}(t)$, the lowest defender payoff for leaving a target uncovered. Now instead of finding the coverage $\mathbf{c}$ which maximizes $U_1^d(\mathbf{c})$ and satisfies $\mathbf{b}$, we can use ORIGAMI-M to determine if there exists a coverage vector $\mathbf{c}$ such that $\mathbf{b}^+$ is satisfied.

Algorithm 5: ORIGAMI-A$(\mathbf{b}, \alpha)$

```
1  c ← empty coverage vector;
2  b_1^+ ← min_{t∈T} U_1^{u,d}(t);
3  b^+ ← {b_1^+} ∪ b ;
4  for 1 ≤ i ≤ n do
5      lower ← b_i^+;
6      upper ← max_{t∈T} U_i^{c,d}(t);
7      while upper − lower > α do
8          b_i^+ ← (upper+lower)/2;
9          c' ← ORIGAMI-M(b^+);
10         if c' = violated then
11             upper ← b_i^+;
12         else
13             c ← c', lower ← b_i^+;
14     b_i^+ ← U_i^d(c);
15 return c ;
```

ORIGAMI-A finds an approximately optimal coverage vector $\mathbf{c}$ by using ORIGAMI-M to solve a series of feasibility problems. This series is generated by sequentially performing binary search on the objectives starting with initial lower bounds defined in $\mathbf{b}^+$. For objective i, the lower and upper bounds for the binary search are, respectively, b_i^+ and $\max_{t \in T} U_1^{c,d}(t)$, the highest defender payoff for covering a target. At each iteration, $\mathbf{b}^+$ is updated by setting $b_i^+ = (upper + lower)/2$ and then passed as input to ORIGAMI-M. If $\mathbf{b}^+$ is found to be feasible, then the lower bound is updated to b_i^+ and $\mathbf{c}$ is updated to the output of ORIGAMI-M, otherwise the upper bound is updated to b_i^+. This process is repeated until the difference between the upper and lower bounds reaches the termination threshold, α. Before proceeding to the next objective, b_i^+ is set to $U_i^d(\mathbf{c})$ in case the binary search terminated on an infeasible problem. After searching over each objective, ORIGAMI-A will return a coverage vector $\mathbf{c}$ such that $U_1^d(\mathbf{c}^*) - U_1^d(\mathbf{c}) \leq \alpha$, where $\mathbf{c}^*$ is the optimal coverage vector for a CSOP defined by $\mathbf{b}$.

The solutions found by ORIGAMI-A are no longer Pareto optimal. Let Ω_α be the objective space of the solutions found by ORIGAMI-A. We can bound its efficiency loss using the approximation measure $\rho(\epsilon, \alpha) = \max_{\mathbf{v} \in \Omega} \min_{\mathbf{v}' \in \Omega_\alpha} \max_{1 \leq i \leq n} (v_i - v'_i)$.

THEOREM 5. $\rho(\epsilon, \alpha) \leq \max\{\epsilon, \alpha\}$.

PROOF. Similar to the proof of Theorem 4, for each point $\mathbf{v} \in \Omega$, we can use Algorithm 2 to find a CSOP with constraints $\mathbf{b}$ which is solved using ORIGAMI-A with coverage $\mathbf{c}$ such that 1) $b_i \leq v_i$ for $i > 1$ and 2) $v'_i \geq v_i - \epsilon$ for $i > 1$ where $\mathbf{v}' = U^d(\mathbf{c})$.

Assume that the optimal coverage is $\mathbf{c}^*$ for the CSOP with constraints $\mathbf{b}$. It follows that $U_1^d(\mathbf{c}^*) \geq v_1$ since the coverage resulting in point $\mathbf{v}$ is a feasible solution to the CSOP with constraints $\mathbf{b}$. ORIGAMI-A will terminate if the difference between lower bound and upper bound is no more than α. Therefore, $v'_1 \geq U_1^d(\mathbf{c}^*) - \alpha$. Combining the two results, it follows that $v'_1 \geq v_1 - \alpha$.

Therefore, for any point missing in the frontier $\mathbf{v} \in \Omega$, we can find a point $\mathbf{v}' \in \Omega_\alpha$ such that 1) $v'_1 \geq v_1 - \alpha$ and $v'_i \geq v_i - \epsilon$ for $i > 1$. It then follows that $\rho(\epsilon, \alpha) \leq \max\{\epsilon, \alpha\}$. □

8. EVALUATION

We perform our evaluation by running the full algorithm in order to generate the Pareto frontier for randomly-generated MOSGs. For our experiments, the defender's covered payoff $U_i^{c,d}(t)$ and attacker's uncovered payoff $U_i^{u,a}(t)$ are uniformly distributed integers between 1 and 10 for all targets. Conversely, the defender's uncovered payoff $U_i^{u,d}(t)$ and attacker's covered payoff $U_i^{c,a}(t)$

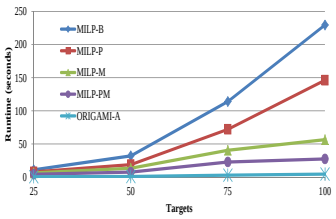

Figure 4: Scaling up targets

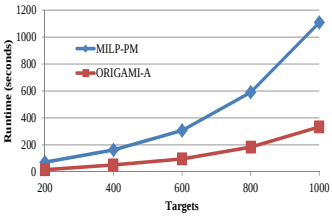

Figure 5: More target scale up

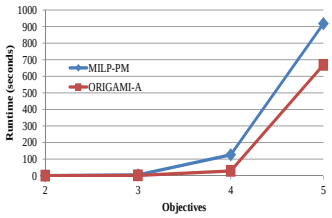

Figure 6: Scaling up objectives

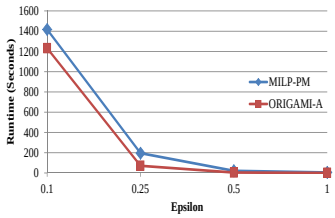

Figure 7: Scaling down epsilon

are uniformly distributed integers between -1 and -10. Unless otherwise mentioned, the setup for each experiment is 3 objectives, 25 targets, $\epsilon = 1.0$, and $\alpha = 0.001$. The amount of defender resources m is fixed at 20% of the number of targets. For experiments comparing multiple formulations, all formulations were tested on the same set of MOSGs. A maximum cap on runtime for each sample is set at 1800 seconds. We solved our MILP formulations using CPLEX version 12.1. The results were averaged over 30 trials.

8.1 Runtime Analysis

We evaluted five MOSG formulations. We refer to the baseline MILP formulation as MILP-B. The MILP formulation adding a bound on the defender's payoff for the primary objective is MILP-P. MILP-M uses ORIGAMI-M to compute bounds on defender coverage. MILP-P can be combined with MILP-M to form MILP-PM. The algorithmic approach using ORIGAMI-A will be referred to by name. For the number of targets, we evaluate all five formulations for solving CSOPs. We then select ORIGAMI-A and the fastest MILP formulation, MILP-PM, to evaluate the remaining factors.

Effect of the Number of Targets:This section presents results showing the efficiency of our different formulations as the number of targets is increased. In Figure 4, the x-axis represents the number of the targets in the MOSG. The y-axis is the number of seconds needed by Iterative ϵ-Constraints to generate the Pareto frontier using the different formulations for solving CSOPs. Our baseline MILP formulation, MILP-B, has the highest runtime for each number of targets we tested. By adding an upper bound on the defender payoff for the primary objective, MILP-P yields a runtime savings of 36% averaged over all numbers of targets compared to MILP-B. MILP-M uses ORIGAMI-M to compute lower bounds for defender coverage, resulting in a reduction of 70% compared to MILP-B. Combining the insights from MILP-P and MILP-M, MILP-PM achieves an even greater reduction of 82%. Removing the computational overhead of solving MILPs, ORIGAMI-A is the most efficient formulation with a 97% reduction. For 100 targets, ORIGAMI-A requires 4.53 seconds to generate the Pareto frontier, whereas the MILP-B takes 229.61 seconds, a speedup of >50 times. Even compared to fastest MILP formulation, MILP-PM at 27.36 seconds, ORIGAMI-A still achieves a 6 times speedup. T-test yields p-value<0.001 for all comparison of different formulations when there are 75 or 100 targets.

We conducted an additional set of experiments to determine how MILP-PM and ORIGAMI-A scale up for an order of magnitude increase in the number of targets by testing on MOSGs with between 200 and 1000 targets. Based on the trends seen in the data, we can concluded that ORIGAMI-A significantly outperforms MILP-PM for MOSGs with large number of targets. Therefore, the number of targets in an MOSG is not a prohibitive bottleneck for generating the Pareto frontier using ORIGAMI-A.

Effect of the Number of Objectives: Another key factor on the efficiency of Iterative ϵ-Constraints is the number of objectives which determines the dimensionality of the objective space that Iterative ϵ-Constraints must search. We ran experiments for MOSGs with between 2 and 6 objectives. For these experiments, we fixed the number of targets at 10. Figure 6 shows the effect of scaling up the number of objectives. The x-axis represents the number of objectives, whereas the y-axis indicates the average time needed to generate the Pareto frontier. For both MILP-PM and ORIGAMI-A, we observe an exponential increase in runtime as the number of objectives is scaled up. For both approaches, the Pareto frontier can be computed in under 5 seconds for 2 and 3 objectives. Whereas, with 6 objectives neither approach is able to generate the Pareto frontier before the runtime cap of 1800 seconds. These results show that the number of objectives, and not the number of targets, is the key limiting factor in solving MOSGs.

Effect of Epsilon: A third critical factor on the running time of Iterative ϵ-Constraints is the value of the ϵ parameter which determines the granularity of the search process through the objective space. In Figure 7, results are shown for ϵ values of 0.1, .25, .5, and 1.0. Both MILP-PM and ORIGAMI-A see a sharp increase in runtime as the value of ϵ is decreased due to the rise in the number of CSOPs solved. For example, with $\epsilon = 1.0$ the average Pareto frontier consisted of 49 points, whereas for $\epsilon = 0.1$ that number increased to 8437. Due to the fact that ϵ is applied to the $n-1$ dimensional objective space, the increase in the runtime resulting from decreasing ϵ is exponential in the number of secondary objectives. Thus, using small values of ϵ can be computationally expensive, especially if the number of objectives is large.

Effect of the Similarity of Objectives: In previous experiments, all payoffs were sampled from a uniform distribution resulting in independent objective functions. However, it is possible that in a security setting, the defender could face multiple attacker types which share certain similarities, such as the same relative preferences over a subset of targets. To evaluate the effect of objective similarity on runtime, we used a single security game to create a Gaussian function with standard deviation σ from which all the payoffs for an MOSG are sampled. Figure 8 shows the results for using ORIGAMI-A to solve MOSGs with between 3 and 7 objectives using σ values between 0 and 2.0 as well as for uniformly distributed objectives. For $\sigma = 0$, the payoffs for all security games are the same, resulting in Pareto frontier consisting of a single point. In this extreme example, the number of objectives does not impact the runtime. However, as the number of objectives increases, less dissimilarity between the objectives is needed before the runtime starts increasing dramatically. For 3 and 4 objectives, the amount of similarity has negligible impact on runtime. The experiments with 5 objectives time out after 1800 seconds for the uniformly distributed objectives. Whereas, 6 objectives times out at $\sigma = 1.0$ and 7 objectives at $\sigma = 0.5$. We conclude that it is possible to scale to larger number of objectives if there is similarity between the attacker types.

8.2 Solution Quality Analysis

Effect of Epsilon: If the Pareto frontier is continuous, only a subset of that frontier can be generated. Thus, it is possible that one of the Pareto optimal points not generated by Iterative ϵ-Constraints

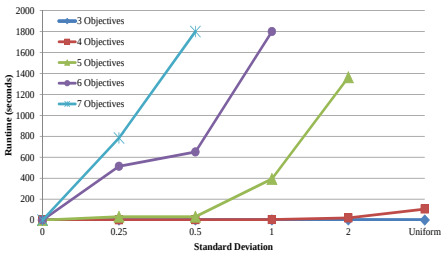

Figure 8: Objective similarity

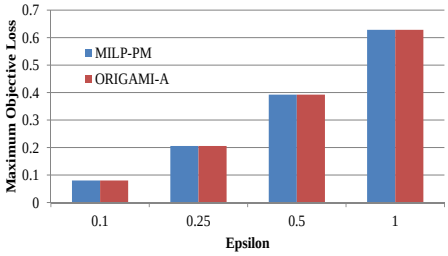

Figure 9: Epsilon solution quality

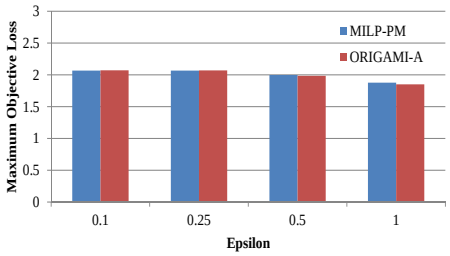

Figure 10: Comparison against uniformly weighted Bayesian security games

would be the most preferred solution, were it presented to the end user. In Section 5.2, we proved that the maximum utility loss for each objective resulting from this situation could be bounded by ϵ. We conducted experiments to empirically verify our bounds and to determine if the actual maximum objective loss was less than ϵ.

Ideally, we would compare the Pareto frontier generated by Iterative ϵ-Constraints to the true Pareto frontier. However, the true Pareto frontier may be continuous and impossible for us to generate, thus we simulate the true frontier by using $\epsilon = 0.001$. Due to the computational complexity associated with such a value of ϵ, we fix the number of objectives to 2. Figure 9 shows the results for ϵ values of 0.1, .25, .5, and 1.0. The x-axis represent the value of ϵ, whereas the y-axis represents the maximum objective loss when comparing the generated Pareto frontier to the true Pareto frontier. We observe that the maximum objective loss is less than ϵ for each value of ϵ tested. At $\epsilon = 1.0$, the average maximum objective loss is only 0.63 for both MILP-PM and ORIGAMI-A. These results verify that the bounds for our algorithms are correct and that in practice we are able to generate a better approximation of the Pareto frontier than the bounds would suggest.

Comparison against Uniform Weighting: We introduced the MOSG model, in part, because it eliminates the need to specify a probability distribution over attacker types a priori. However, even if the probability distribution is unknown it is still possible to use the Bayesian security game model with a uniform distribution. We conducted experiments to show the potential benefit of using MOSG over Bayesian security games in such cases. We computed the maximum objective loss sustained by using the Bayesian solution as opposed to a point in the Pareto frontier generated by Iterative ϵ-Constraints. If $\mathbf{v}'$ is the solution to a uniformly weighted Bayesian security game then the equation for maximum objective loss is $\max_{v \in \Omega_\epsilon} \max_i (v_i - v'_i)$. Figure 10 shows the results for ϵ values of 0.1, .25, .5, and 1.0. At $\epsilon = 1.0$, the maximum objective loss were 1.87 and 1.85 for MILP-PM and ORIGAMI-A. Decreasing ϵ all the way to 0.1 increases the maximum objective loss by less than 12% for both algorithms. These results suggests that ϵ has limited impact on maximum objective loss, which is a positive result as it implies that solving an MOSG with a large ϵ can still yield benefits over a uniform weighted Bayesian security game.

9. CONCLUSION

We built upon insights from game theory and multi-objective optimization to introduce a new model, multi-objective security games (MOSG), for domains where security forces must balance multiple objectives. Contributions include: 1) Iterative ϵ-Constraints, a high-level approach for transforming MOSGs into a sequence of CSOPs, 2) exact MILP formulations, both with and without heuristics, for solving CSOPs, and 3) ORIGAMI-A, an approximate approach for solving CSOPs. We then provided bounds for both the complexity as well as the solution quality of our approaches; additionally we provided detailed experimental comparison of the different approaches presented.

10. ACKNOWLEDGEMENT

This research was supported by the United States Department of Homeland Security through the National Center for Border Security and Immigration (NCBSI).

11. REFERENCES

[1] B. An, J. Pita, E. Shieh, M. Tambe, C. Kiekintveld, and J. Marecki. Guards and protect: next generation applications of security games. *ACM SIGecom Exchanges*, 10(1):31–34, 2011.

[2] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, pages 57–64, 2009.

[3] K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner. Approximation-guided evolutionary multi-objective optimization. In *IJCAI*, pages 1198–1203, 2011.

[4] V. Chankong and Y. Haimes. *Multiobjective decision making: theory and methodology*, volume 8. North-Holland New York, 1983.

[5] V. Conitzer and D. Korzhyk. Commitment to correlated strategies. In *AAAI*, pages 632–637, 2011.

[6] K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. Wiley, 2001.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Trans. on Evolutionary Computation*, 6(2):182–197, 2002.

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

[9] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordonez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, pages 689–696, 2009.

[10] D. Korzhyk, V. Conitzer, and R. Parr. Security games with multiple attacker resources. In *IJCAI*, pages 273–279, 2011.

[11] M. Laumanns, L. Thiele, and E. Zitzler. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942, 2006.

[12] G. Mavrotas. Effective implementation of the [epsilon]-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455–465, 2009.

[13] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games with security: An efficient exact algorithm for bayesian stackelberg games. In *AAMAS*, pages 895–902, 2008.

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