Autonomous Agents and Multi-Agent Systems manuscript No.
(will be inserted by the editor)

An Extended Study on Multi-Objective Security Games

Matthew Brown · Bo An · Christopher Kiekintveld · Fernando Ordóñez · Milind Tambe

the date of receipt and acceptance should be inserted later

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. In such domains, decision makers have multiple competing objectives they must consider which may take different forms that are not readily comparable including safety, cost, and public perception. Thus, it can be difficult to know how to weigh the different objectives 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). Instead of a single optimal solution, MOSGs have a set of Pareto optimal (non-dominated) solutions referred to as the Pareto frontier, which can be generated by solving a sequence of constrained single-objective optimization problems (CSOP). The Pareto frontier allows the decision maker to analyze the tradeoffs that exist between the multiple 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; (iii) heuristics that achieve speed up by exploiting the structure of security games to further constrain the MILP; (iv) an approximate approach for solving a CSOP built off those same heuristics, increasing the scalability of our approach with quality guarantees. Additional contributions of this paper include proofs on the level of approximation, detailed experimental evaluation of the proposed approaches and heuristics, as well as a discussion on techniques for visualizing the Pareto frontier.[1]

Keywords Game Theory · Security · Multi-objective Optimization

Matthew Brown · Bo An · Milind Tambe
University of Southern California, Los Angeles, CA, 90089
E-mail: {mattheab, boa, tambe}@usc.edu

Christopher Kiekintveld
University of Texas at El Paso, El Paso, TX, 79968
E-mail: cdkiekintveld@utep.edu

Fernando Ordóñez
Universidad de Chile, Santiago, Chile,
E-mail: fordon@dii.uchile.cl

[1] A preliminary version of this work appeared as the conference paper [7].

1 Introduction

Game theory is an increasingly important paradigm for modeling security domains which feature complex resource allocation [4, 10]. Security games, an important class of attacker-defender Stackelberg games, are at the heart of several significant deployed decision-support applications. Such systems include ARMOR at the Los Angeles International Airport (LAX) [20], IRIS deployed by the US Federal Air Marshals Service [20], GUARDS developed for the US Transportation Security Administration [3], and PROTECT used at the Port of Boston by the US Coast Guard [3].

While multiple objectives may have been present in these domains, the games are modeled as having the defender optimizing a single objective as the necessary solution concepts did not exist. 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.[2] From the perspective of LASD, each one of these attacker types presents a unique threat. Fare evaders are directly responsible for lost revenue by not purchasing the appropriate tickets, criminals can commit crimes against property and persons which undermine the perceived safety of the metro system, and terrorists can inflict massive casualties, causing long-term system-wide disruptions, and spreading fear through the general public. Given that preventing these threats yield different types of benefit, protecting against each type of attacker could correspond to an objective for LASD.

With a diverse set of attacker types, selecting a security strategy is a significant challenge as no single strategy can maximize all of the objectives. Thus, tradeoffs must be made as increasing protection against one attacker type may increase the vulnerability to another attacker type. However, it is not clear how LASD should weigh the objectives when determining the security strategy to use. One could attempt to establish methods for converting the benefits of protecting against each attacker type into a single objective. 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 [3, 11, 20, 22, 32] 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. However, there are three potential factors limiting the applicability 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 the defender rewards for different attacker types, and (3) only one solution is given, hiding the trade-offs between the objectives from the end user.

We propose a new game model, multi-objective security games (MOSG), which combines game theory and multi-objective optimization. Such a model is suitable for domains like the LASD metro system, as the threats posed by the attacker types (ticketless travelers, criminals, and terrorists) 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 may 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 in-

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

stance, LASD has suggested that rather than having a single option handed to them, they would be interested in being presented with a set of alternative strategies from which they can make a final selection. Overall, there has been a growing trend towards multi-objective decision making in a wide variety of areas, including transportation [5] and energy [34]. We are pursuing along in the same direction but now from a game-theoretic perspective.

Our key contributions include (i) Iterative-ϵ-Constraints, an algorithm for generating the Pareto frontier for MOSGs by producing 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 speed up solving the MILPs; 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, detailed experimental results evaluating the effect of MOSG properties and algorithm parameters on performance, as well as several techniques for visualizing the Pareto frontier.

The structure of this article is as follows: Section 2 motivates our research by providing a detailed description of the LASD domain. Section 3 formally introduces the MOSG model as well as multi-objective optimization concepts such as the Pareto frontier and Pareto optimality. Section 4 explores the related work on the leading multi-objective optimization techniques. Section 5 introduces the Iterative-ϵ-Constraints algorithm for solving a series of CSOPs to generate the Pareto frontier. Section 6 presents the MILP formulation for solving each CSOP. Section 7 proposes heuristics which can be used to constrain our MILP formulation, including three algorithms (ORIGAMI-M, ORIGAMI-M-BS, and DIRECT-MIN-COV) for computing on lower bounds defender coverage. Section 8 introduces an approximate algorithm (ORIGAMI-A) for solving CSOPs based on the defender coverage heuristics. Section 9 provides experimental results for all of our algorithms and heuristics as well as analysis on the properties of the MOSG model. Section 10 discusses a number of approaches for visualizing the Pareto frontier as a step in the decision making process for selecting a security policy to implement. We conclude this paper and outline future research directions in Section 11.

This article is an extension of [7] and features a significant amount of new material. First, Section 7 now includes two new heuristic algorithms for computing lower bounds on defender coverage which can be used in both our exact and approximate CSOP solvers. ORIGAMI-M-BS expands the attack set using binary search, while DIRECT-MIN-COV avoids having to precompute the attack set but instead computes multiple defender coverage vectors. Second, Section 9 has been enhanced with additional experiments and analysis. We have added experiments on objective function clustering which looks at the effect of payoff similarity between a subset of objectives. We also include experiments on constraint computation which examines how often ORIGAMI-M violates a constraint that it has previously satisfied. Another experiment analyzes a CSOP pruning heuristic which further exploits the concept of Pareto dominance. The last set of experiments compares performance between ORIGAMI-M, ORGAMI-M-BS, and DIRECT-MIN-COV when either the number of targets or the ratio of targets to defender resources is varied. Third, we have included an entirely new section (Section 10) on different approaches and issues relating to visualizing the Pareto frontier. Fourth, Section 2 now has a more detailed description of our LASD domain which serves as motivation for the MOSG model. Finally, Section 4 has been significantly expanded, providing additional references as well as overview of the different approaches for solving multi-objective optimization problems.

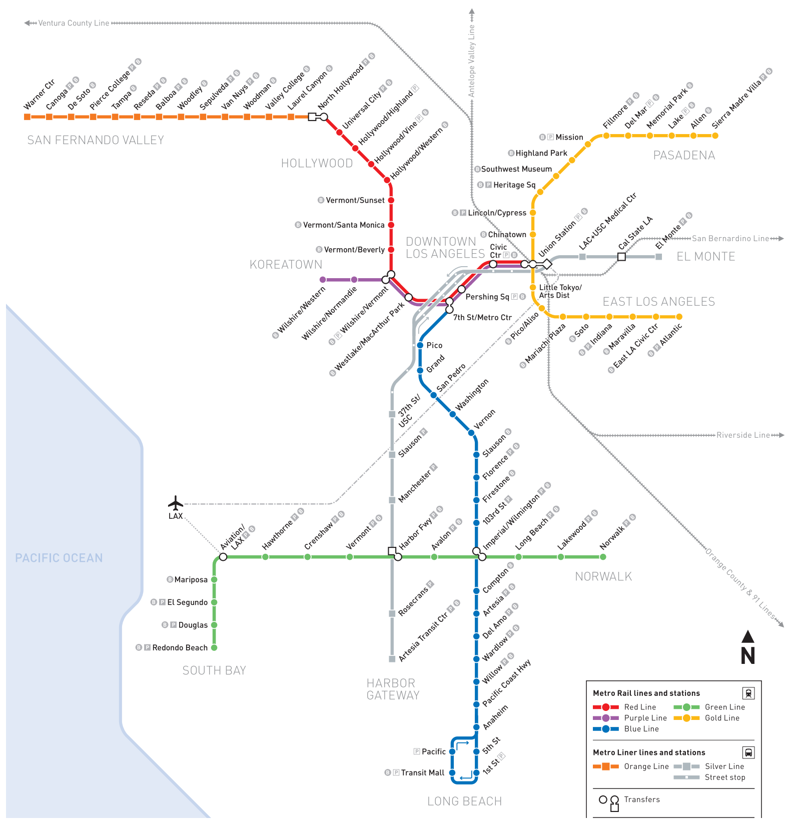

Fig. 1 Los Angeles rail system.

2 Motivating Domain

There are a variety of real-world security domains in which the defender has to consider multiple, and potentially conflicting, objectives when deciding upon a security policy. In this section, we focus on the one specific example of transportation security, in which LASD is responsible for protecting the Los Angeles metro system, shown in Figure 1.[3] The metro system consists of 70 stations and maintains a weekday ridership of over 300,000 passengers. The LASD is primarily concerned with protecting the metro system from three adversary types: ticketless travelers, criminals, and terrorists. A significant number of the rail stations feature barrier-free entrances that do not employ static security measures such as metal detectors or turnstiles. Instead randomized patrols and inspections are utilized in order to verify that passengers have purchased a valid ticket as well as to generally maintain security of the system. Thus, LASD must make decisions on how best to allocate their available security resources as well as on how frequently to visit each station.

Each of the three adversary types are distinct and present a unique set of challenges which may require different responses by LASD. For example, each adversary may have different preferences over the stations they choose to target. Ticketless travelers may choose

[3] http://www.metro.net/riding_metro/maps/images/rail_map.pdf

to fare evade at busier stations thinking that the larger crowds decrease the likelihood of having their ticket checked. Whereas, criminals may prefer to commit crimes at less frequented stations, as they believe the reduced crowds will result in a smaller security presence. Finally, terrorists may prefer to strike stations which hold economic or cultural significance, as they believe that such choice of targets can help achieve their political goals.

LASD may also have different motivation for preventing the various adversary types. It is estimated that fare evasion costs the Los Angeles metro system over $5 million in lost revenue each year [18]. Deploying security policies that target ticketless travelers can help to recuperate a portion of this lost revenue as it implicitly encourages passengers to purchase tickets. Pursuing criminals will reduce the amount of property damage and violent crimes, increasing the overall sense of passenger safety. In 2010, 1216 "part one crimes" were reported on the metro system, which includes homicide, rape/attempted rape, assault, robbery, burglary, grand theft, and petty theft.[4] Most significantly, the rail system experienced its first and only slaying when a man was fatally stabbed on the subway in August 2011. Finally, due to the highly sensitive nature of the information, statistics regarding the frequency and severity of any terrorist threats targeting the transit system are not made available to the public. However, the city of Los Angeles is well known to be a high priority target given the much publicized foiling of attempted terrorist attacks at LAX in 2000 and 2005. Additionally, trains and subway systems are common targets for terrorism, as evidenced by the devastating attacks on Madrid in 2004 and London in 2005. Thus, despite the relatively low likelihood of a terrorist attack, security measures designed to prevent and mitigate the effects of terrorism must always remain a priority, given the substantial number of lives at risk.

LASD is required to simultaneously consider all of the threats posed by the different adversary types in order to design effective and robust security strategies. Thus, defending against each adversary type can be viewed as an objective for LASD. While these objectives are not strictly conflicting (e.g. checking tickets at a station may lead to a reduction in crime), focusing security measures too much on one adversary may neglect the threat posed by the others. As LASD has finite resources with which to protect all of the stations in the city, it is not possible to protect all stations against all adversaries 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 protecting against different adversaries provides different, incomparable benefits to LASD, it may be unclear how to specify such a decision as maximizing a single objective for automated analysis (as in ARMOR and similar systems). Instead, a more interactive process whereby the decision support system presents possible solutions to the decision-makers for further analysis and human judgment may be preferable

For a domain such as the Los Angeles metro system, an MOSG model could be of use, as it can capture the preferences and threats of the adversary types as well as the benefit to LASD of preventing these threats. Solving the MOSG produces a set of candidate solutions with each solution corresponding to a security policy and a set of expected payoffs for LASD, one for each adversary. Thus, different solutions can be compared to better understand the trade-offs between the different objectives. LASD can then select the security policy they feel most comfortable with based on the information they have available. For this type of evaluation process to occur, we must be able to both generate and visualize the Pareto frontier. Our research focuses primarily on developing efficient algorithms for solving MOSGs and generating the Pareto frontier (Sections 5 through 8), but we also touch on issues relating to visualization (Section 10).

[4] http://thesource.metro.net/2011/09/21/statistics-on-crime-on-metro-buses-and-trains/

3 Multi-Objective Security Games

A multi-objective security game (MOSG) is a multi-player game between a defender and n attacker types.[5] 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 MOSG model adopts the Stackelberg framework in which the defender acts first by committing to a strategy that the attackers are able to observe and best respond. 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 [22]. This formulation assumes that the covering of each target costs the same amount of resources, specifically one defender resource. It is this assumption that allows for the equivalence between the amount of resources placed on a target and the probability of that target being covered. Thus, given a budget of m resources, the defender could choose to fully protect m targets. However, given the Stackelberg paradigm, such a deterministic strategy would perform poorly, as the attackers can easily select one of the targets that are known to be unprotected. Therefore, the defender has incentive to consider mixed strategies where resources are allocated to a larger set of partially protected targets. While an attacker is still able to observe this mixed strategy, when the MOSG is actually played there is uncertainty on the attacker's part as to whether a target will be covered or not. More formally, $C = \{\langle c_t \rangle | 0 \leq c_t \leq 1, \sum_{t \in T} c_t \leq m\}$ describes the defender's strategy space. The mixed strategy for attacker type i, $\mathbf{a}_i = \langle a_i^t \rangle$, 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 type i. $U_i^{c,d}(t)$ is the defender's utility if t is chosen by attacker type i and is fully covered ($c_t = 1$). If t is uncovered ($c_t = 0$), 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 [22]. For a strategy profile $\langle \mathbf{c}, \mathbf{a}_i \rangle$ for the game between the defender and attacker type 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 type 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) [39], in which the defender commits first to an optimal strategy based on the assumption that the attacker will be able to observe this strategy and then 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 type i, respectively, when the defender uses the coverage vector $\mathbf{c}$ and attacker type i attacks the best target while breaking ties in favor of the defender.

[5] The defender may actually face multiple attackers of different types, however, these attackers are not coordinated and hence the problem we address is different than in [24].

With multiple attacker types, 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:

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

We associate a different objective with each attacker type because, as pointed out in Section 2, protecting against different attacker types may yield types of payoff to the defender which are not directly comparable. This is in contrast to Bayesian security games, which uses probabilities to combine the objectives into a single weighted objective, making the assumption about identical units of measure for each attacker type.

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. For many multi-objective optimization problems, the Pareto frontier contains a large or even infinite number of solutions. In these situations, 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. This analysis can include creating a visual representation of the Pareto frontier, a topic we discuss in Section 10.

4 Related Work

MOSGs build on both security games as well as 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. The techniques for solving multi-objective optimization problems can be broken down into three categories [16]: *a priori*, *interactive*, and *a posteriori* methods. This classification is determined by the phase in which the decision maker expresses their preferences.

If the preferences of the decision maker are known *a priori* [35,41] then this information can be incorporated into the solution process by assigning each objective i a weight w_i according to its relative importance and then solving the maximization problem

$$\max_{\mathbf{c}\in C} \sum_{i=1}^{n} w_i U_i^d(\mathbf{c}).$$

This weighted summation technique [8] effectively turns a multi-objective optimization problem into a single-objective optimization problem which implies the existence of a single optimal solution. However, it is often difficult for the decision maker to both know and articulate their preferences, especially if prior knowledge as to the shape of the solution space is limited. Bayesian security games are solved using this formulation with the weights in w representing the probability distribution over attacker types. Another issue is that not all preferences over multiple objectives can be expressed as simple weighted summations, more complex preferences may be desired.

Interactive methods [2,30,36] involve alternating between computation and dialogue phases. In the computation phase, a set of solutions are computed and presented to the decision maker. In the dialogue phase, the decision maker is asked about their preferences over the set of solutions. The decision maker can thus guide the search process with their responses toward a preferable solution. By using preference elicitation, only a subset of the Pareto frontier needs to be generated and reviewed. The drawback is that the decision maker never has the opportunity to view the entire Pareto frontier at once and could potentially miss out on a more preferable solution. In addition, solutions must be computed in an online manner which also requires synchronization between the system and the decision maker.

Finally, there will be instances where the preferences of the decision maker are only known *a posteriori*. In this situation, the entire Pareto frontier (or a representative subset) is generated and presented to the decision maker. While this approach is the most expensive computationally, it provides the most information, enabling the decision maker to make an informed decision. The *a posteriori* techniques are also known as generative methods and will be the focus of this paper. The three most common generative approaches are weighted summation [23], evolutionary algorithms [9], and the ϵ-constraint method [15].

When weighted summation [8] and its successors are used as a generative approach, the true weights of the decision maker are not known. Thus, it is necessary to sample many different combinations of weights in order to generate the Pareto frontier. Solving for one assignment of weights, w, produces a Pareto optimal solution. Since the weight vector is an artificial construct which may not have any real meaning in the optimization problem, it is difficult to know how to update the weights in order to generate different solutions on the Pareto frontier. Another limitation of weighted summation is that it is only guaranteed to find Pareto-optimal solutions in the convex region of the Pareto frontier. The weighted p-power method [28] and the weighted minimax method [27] were introduced as improved versions of weighted summation capable of handling nonconvex problems.

Another approach for generating the Pareto frontier which has seen significant application [1,14,37] is multi-objective evolutionary algorithms (MOEA) [12]. This class of algorithms is inspired by biological concepts such as reproduction, mutation, recombination, and selection. A population of candidate solutions is maintained and evolved over multiple generations, where the likelihood of survival for individual solutions is determined by a fitness function. A key advantage of evolutionary algorithms such as NSGA-II [13], SPEA-2 [42], and GDE3 [25] is that there is no need to solve optimization problems as the assignment of decision variables are passed down genetically from generation to generation. However, due to the stochastic nature of evolutionary algorithms, the solutions returned by these approaches are not Pareto-optimal but rather approximate solutions. Additionally, it is not possible to bound this level of approximation, making evolutionary algorithms unsuitable for the security domains on which we focus, where quality guarantees are critical.

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 other secondary objectives. By varying the constraints, different solutions on the Pareto frontier can be generated. The original ϵ-constraint method [8] discretizes the objective space and solves a CSOP for each grid point. This approach is computationally expensive since it exhaustively searches the high-dimensional space formed by the secondary objectives. There has been work to improve upon the original ϵ-constraint method. In [26], an adaptive constraint variation scheme is proposed which is able make use of information obtained from previously computed subproblems. However, the exponential complexity of $\mathcal{O}(k^{n-1})$, where k is the number of solutions in the Pareto frontier, limits its application as the Pareto frontier can be large or even continuous for many real world optimization problems. Another approach, the augmented ϵ-constraint method [31] 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.

Security domains demand *both* efficiency as well as solution quality guarantees when providing decision support. Given these requirements, our approach for solving an MOSG utilizes and improves upon the ϵ-constraint method. Iterative-ϵ-Constraints, which will be expanded upon in Section 5, combines the following innovations: (1) using a recursive, tree-based algorithm to search the objective space instead of a predefined grid, (2) dynamically generating CSOPs using adaptive constraints from previously computed CSOPs, and (3) exploiting infeasibility information to avoid unnecessary computation. As a result, our approach only needs to solve $\mathcal{O}(nk)$ CSOPs and can provide approximation bounds. Also, our work is one of the first to consider multi-objective optimization in the context of game theory, where the decision maker needs to predict the response of multiple adversaries to evaluate the objectives.

5 Iterative-ϵ-Constraints

Using the ϵ-constraint method, we translate a multi-objective optimization problem into the following constrained single-objective optimization problem (CSOP) by transforming all but one of the optimizations into a set of constraints $\mathbf{b}$.

$$\begin{aligned}
\max_{\mathbf{c}\in C} \quad & U_1^d(\mathbf{c}) \\
& U_2^d(\mathbf{c}) \geq b_2 \\
& U_3^d(\mathbf{c}) \geq b_3 \\
& \quad \dots \\
& U_n^d(\mathbf{c}) \geq b_n
\end{aligned}$$

This allows for the use of standard optimization techniques to solve for a single Pareto optimal solution, which is a vector of payoffs $\mathbf{v} = (U_1^d(\mathbf{c}), \dots, U_n^d(\mathbf{c}))$. The Pareto frontier is then generated by solving multiple CSOPs produced by modifying the constraints in $\mathbf{b}$.

This section presents Iterative-ϵ-Constraints (Algorithm 1), an algorithm for systematically generating a sequence of CSOPs for an MOSG. After each CSOP is generated, it is passed to a solver Φ and if a solution is found that information is used to generate additional CSOPs. In Section 6, we present a MILP approach which guarantees the Pareto optimality of each CSOP solution. While in Section 8, we introduce a faster, approximate approach for solving CSOPs.

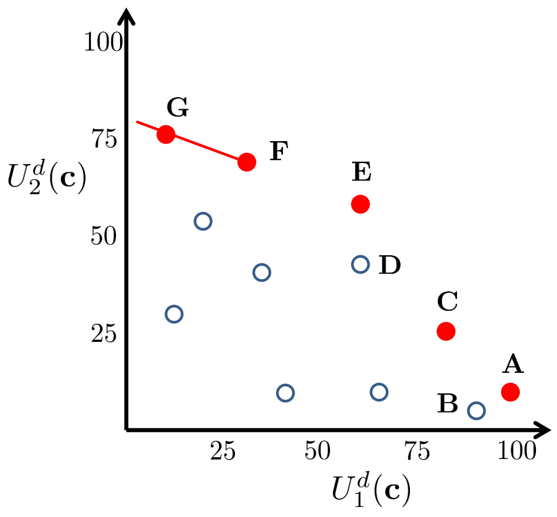

Fig. 2 Pareto frontier for a bi-objective MOSG.

We have made the code for all of our algorithms available to the community for download.[6] In addition to our algorithms, this code base also provides a framework for implementing new multi-objective domains and solvers.

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 can be 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 needed to ensure that the 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 or 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 2 shows the objective space for the problem as well as several points representing the objective payoff vectors for different defender coverage vectors. In this problem, U_1^d will be maximized while b_2 constrains U_2^d, meaning that the utility of the second objective U_2^d should be no less than b_2. 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) in the objective space 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

[6] http://teamcore.usc.edu/people/mattheab/multi/

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.

Iterative-ϵ-Constraints 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 $n-1$ 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 all of the leaf nodes in the search tree are infeasible, meaning the entire secondary objective space has been searched.

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

```
1  if $\mathbf{b} \notin$ previousBoundsList then
2      append(previousBoundsList, $\mathbf{b}$) ;
3      $\mathbf{c} \leftarrow \Phi(\mathbf{b})$ ;
4      if $\mathbf{c}$ is a feasible solution then
5          $\mathbf{v} \leftarrow \{U_1^d(\mathbf{c}), \ldots, U_n^d(\mathbf{c})\}$;
6          for $2 \leq i \leq n$ do
7              $\mathbf{b}' \leftarrow \mathbf{b}$;
8              $b'_i \leftarrow v_i + \epsilon$ ;
9              if $\mathbf{b}' \ngeq \mathbf{s}, \forall \mathbf{s} \in$ infeasibleBoundsList then
10                 Iterative-$\epsilon$-Constraints($\mathbf{b}'$) ;
11     else append(infeasibleBoundsList, $\mathbf{b}$) ;
```

Figure 3 shows the type of search tree generated by Iterative-ϵ-Constraints. In this simple example, there are three objectives and thus the search tree has a branching factor of 2. The number at the top of each node represents the order in which the nodes were processed. Along each branch, we show information about $\mathbf{b}$ and $\mathbf{v}$ being passed down from parent to child. This information is used to create the set of lower bound constraints for the child CSOP which is then passed to the solver Φ. In total, seven CSOPs are computed with three feasible CSOPs (Iterations 1, 2, and 4) and four infeasible CSOPs (Iterations 3, 5, 6, and 7). Figure 4 shows the process taking place within a CSOP with four objectives, where a vector $\mathbf{v}$ of $n-1$ objective lower bounds is used to formulate the constraints of a CSOP which maximizes the remaining, primary objective. This CSOP is then passed to CSOP solver Φ which produces a vector $\mathbf{v}$ of n objective payoff values.

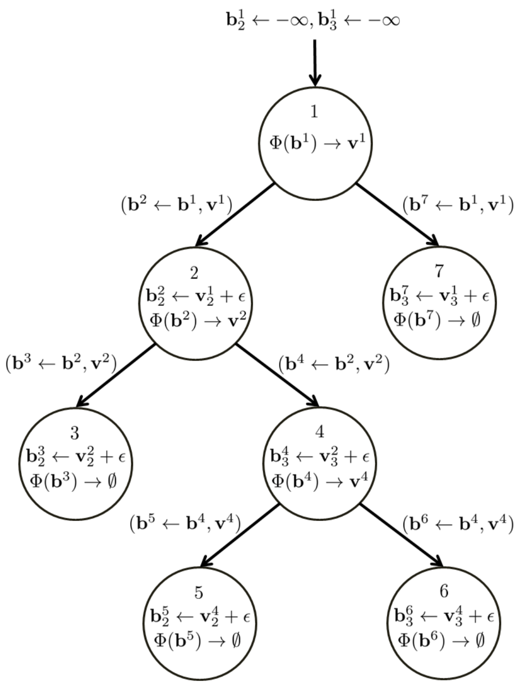
$\mathbf{b}_2^1 \leftarrow -\infty, \mathbf{b}_3^1 \leftarrow -\infty$
1
$\Phi(\mathbf{b}^1) \rightarrow \mathbf{v}^1$
$(\mathbf{b}^2 \leftarrow \mathbf{b}^1, \mathbf{v}^1)$
$(\mathbf{b}^7 \leftarrow \mathbf{b}^1, \mathbf{v}^1)$
2
$\mathbf{b}_2^2 \leftarrow \mathbf{v}_2^1 + \epsilon$
$\Phi(\mathbf{b}^2) \rightarrow \mathbf{v}^2$
7
$\mathbf{b}_3^7 \leftarrow \mathbf{v}_3^1 + \epsilon$
$\Phi(\mathbf{b}^7) \rightarrow \emptyset$
$(\mathbf{b}^3 \leftarrow \mathbf{b}^2, \mathbf{v}^2)$
$(\mathbf{b}^4 \leftarrow \mathbf{b}^2, \mathbf{v}^2)$
3
$\mathbf{b}_2^3 \leftarrow \mathbf{v}_2^2 + \epsilon$
$\Phi(\mathbf{b}^3) \rightarrow \emptyset$
4
$\mathbf{b}_3^4 \leftarrow \mathbf{v}_3^2 + \epsilon$
$\Phi(\mathbf{b}^4) \rightarrow \mathbf{v}^4$
$(\mathbf{b}^5 \leftarrow \mathbf{b}^4, \mathbf{v}^4)$
$(\mathbf{b}^6 \leftarrow \mathbf{b}^4, \mathbf{v}^4)$
5
$\mathbf{b}_2^5 \leftarrow \mathbf{v}_2^4 + \epsilon$
$\Phi(\mathbf{b}^5) \rightarrow \emptyset$
6
$\mathbf{b}_3^6 \leftarrow \mathbf{v}_3^4 + \epsilon$
$\Phi(\mathbf{b}^6) \rightarrow \emptyset$

Fig. 3 Example Iterative-ϵ-Constraints search tree for three objectives.

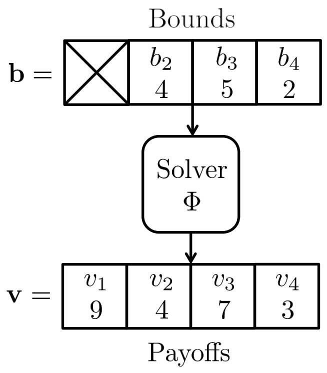

Fig. 4 Internal process for an example CSOP with four objectives.

5.2 Search Tree Pruning

By always going from less constrained CSOPs to more constrained CSOPs, Iterative-ϵ-Constraints is guaranteed to terminate. However, there are several issues which can cause the algorithm to be inefficient. The first issue is redundant computation caused by multiple CSOPs having identical sets of lower bound constraints. When this occurs, the set of child CSOPs generated for each duplicate parent CSOP would also be identical. Given the recursive nature of the algorithm, these duplicate CSOPs can result in an exponential increase in the number of CSOPs that are solved. This issue can be addressed by recording the lower bound constraints for all previous CSOPs in a list called $\text{previousBoundsList}$ and pruning any new CSOP which matches an element in this list. The second issue is the unnecessary computation of CSOPs which are known to be infeasible based on previously computed CSOPs. This can be achieved by recording the lower bound constraints for all CSOPs previously found to be infeasible in a list called $\text{infeasibleBoundsList}$ and pruning any new CSOP for which all lower bounds constraints are greater than or equal to the lower bound constraints of a CSOP in the list. These two heuristics form the baseline pruning rules that are used when evaluating Iterative-ϵ-Constraints in Section 9.

It is possible to further exploit the concept of Pareto dominance in order to create a more effective pruning heuristic. For example, it is possible for two sets of lower bound constraints, $\mathbf{b}^1$ and $\mathbf{b}^2$, to result in the same vector of objective payoffs $\mathbf{v}$. This situation is obviously undesirable not only due to the time spent on the CSOPs corresponding to $\mathbf{b}^1$ and $\mathbf{b}^2$ but also because both CSOPs will have a full set of child CSOPs that need to be processed. While generating some duplicate solutions is unavoidable, steps can be taken to reduce their occurrence. Solving a CSOP creates a mapping of constraints to payoffs, $\Phi(\mathbf{b}) \rightarrow \mathbf{v}$. Each such mapping provides useful information as it creates a dominated region in which no additional CSOPs need to be solved. Specifically, if we have a mapping $\Phi(\mathbf{b}) \rightarrow \mathbf{v}$, then we can prune any CSOP corresponding to $\mathbf{b}'$ such that $\mathbf{b}' \geq \mathbf{b}$ and $\mathbf{b}' \leq \mathbf{v}$. This is the case because for any such $\mathbf{b}'$ the payoffs found by solving the CSOP are guaranteed to be $\mathbf{v}$. Since $\mathbf{b}' \geq \mathbf{b}$, $\mathbf{b}'$ is inherently at least as constrained as $\mathbf{b}$. Given that the CSOP is a maximization problem, if $\mathbf{b}$ maps to $\mathbf{v}$ then a more constrained problem $\mathbf{b}' \leq \mathbf{v}$ must also map to $\mathbf{v}$. Thus, in Iterative-ϵ-Constraints, we can record all of the constraint-payoff mappings in $solutionsMap$. Then before attempting to solve a CSOP corresponding to $\hat{\mathbf{b}}$, we first check to see if $\hat{\mathbf{b}}$ resides within any of the dominated regions defined by any of the mappings in $solutionsMap$. We compare this more sophisticated pruning rule to the baseline pruning rule in Section 9.5.

5.3 Approximation Analysis

When the Pareto frontier contains a large or infinite number of points, it may be undesirable or impossible to produce the entire Pareto frontier. Thus, the set of solutions returned in such situations is an approximation of the true Pareto frontier. In this section, we prove that the solutions found by Iterative-ϵ-Constraints are Pareto optimal, if Φ is exact, and then provide formal bounds on the level of approximation in the generated Pareto frontier. We refer to the full Pareto frontier as Ω and the set of solutions found by Iterative-ϵ-Constraints as 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 $i = 1, \dots, n$ 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.

We have just shown that each solution in Ω_ϵ is indeed Pareto optimal. However, the use of ϵ introduces a degree of approximation in the generated Pareto frontier. Specifically, by not generating the full Pareto frontier, we are approximating the shape of Ω. One immediate question is to characterize the efficiency loss caused by this approximation. Here we define a bound to measure the largest efficiency loss as a function of ϵ:

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

This approximation measure is widely used in multi-objective optimization (e.g. [6]). 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. Here, 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 by ensuring the bounds $\mathbf{b} \leq \mathbf{v}$ 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. This is a significant improvement over [26], which solves $\mathcal{O}(k^{n-1})$ CSOPs as a result of recomputing each cell in an adaptive grid every time a solution is found. Our approach limits recomputing regions of objective space through our pruning heuristics and by moving from less constrained to more constrained CSOPs.

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 $i > 1$

```
1 Let $\mathbf{b}$ be the constraints in the root node, i.e., $b_i = -\infty$ for $i > 1$ ;
2 repeat
3     $\mathbf{c} \leftarrow \Phi(\mathbf{b})$, $\mathbf{v}' \leftarrow U^d(\mathbf{c})$, $\mathbf{b}' \leftarrow \mathbf{b}$;
4     for each objective $i > 1$ do
5         if $v'_i + \epsilon \leq v_i$ then
6             $b_i \leftarrow v'_i + \epsilon$ ;
7             break;
8 until $\mathbf{b} = \mathbf{b}'$;
9 return $\Phi(\mathbf{b})$ ;
```

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. In Section 7, we 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.

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

We present our MILP formulation for a CSOP for MOSGs in Figure 5. This is similar to the MILP formulations for security games presented in [22] and elsewhere with the exception of the key Equations 4 and 5. Equation 1 is the objective function, which maximizes the defender's payoff for objective λ, d_λ. In Equations 2 and 3, M is a large constant relative to the maximum payoff value for any objective. Equation 2 defines the defender's expected payoff d_i for each objective i based on the target selected by attacker type i. The constraint places an upper bound of $U_i^d(c_t, t)$ on d_i, but only for the attacked target. For every other target, M on the right hand side causes the constraint to be arbitrarily satisfied.

Similarly, Equation 3 defines the expected payoff k_i for attacker type i based on the target selected for attack. The first part of the constraint specifies that $k_i - U_i^a(c_t, t) \geq 0$, which implies that k_i must be at least as large as the maximal payoff for attacking any target. The second part forces $k_i - U_i^d(c_t, c) \leq 0$ for the target selected by attacker type i. If the selected target is not maximal, this constraint is violated.

Taken together, Equations 1-3 imply that the strategies for both the defender and attacker type λ are best-responses with respect to each other. However, the same cannot be said about the defender's strategy with respect to all of the other attacker types because the defender's payoffs for those objectives are not included in the objective function. It is for this reason that lexicographic maximization is necessary, ensuring that defender strategy is the best response with respect to all attacker types and the constraints in $\mathbf{b}$.

Equation 4 constrains the feasible region to solutions that maintain the values of objectives maximized in previous iterations of the 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 [32]. Equations 6 and 7 constrain attackers to pure strategies that attack a single target. Equations (8) specifies that the coverage for each target c_t is in the range [0,1]. Finally, Equation 9 ensures the amount of defender coverage used is no greater than the total number of defender resources, m.

As noted earlier, this MILP is a modified version of the optimization problem formulated in [22] and is specific for security games. Similar modifications can be made to more generic Stackelberg games, such as those used for the Decomposed Optimal Bayesian Stackelberg Solver (DOBSS) [32], giving a formulation for generalized multi-objective Stackelberg games beyond security games.

$$
\begin{array}{rcr}
\max & d_\lambda & (1)\\
1 \le i \le n, \forall t \in T: & d_i - U_i^d(c_t,t) \le M(1-a_i^t) & (2)\\
1 \le i \le n, \forall t \in T: & 0 \le k_i - U_i^a(c_t,t) \le M(1-a_i^t) & (3)\\
1 \le j < \lambda: & d_j = d_j^* & (4)\\
\lambda < k \le n: & d_k \ge b_k & (5)\\
1 \le i \le n, \forall t \in T: & a_i^t \in \{0,1\} & (6)\\
\forall j \in A: & \sum_{t\in T} a_i^t = 1 & (7)\\
\forall t \in T: & 0 \le c_t \le 1 & (8)\\
 & \sum_{t\in T} c_t \le m & (9)
\end{array}
$$

Fig. 5 Lexicographic MILP formulation for a CSOP.

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 $U_j^a(c_t,t)$	$n \times \|T\|$

Fig. 6 MILP formulation definitions.

7 Improving MILP Efficiency

Once the MILP has been formulated as specified in Section 6, 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 the child CSOP. As each CSOP is a maximization problem, it must hold that $d_1 \ge d_1'$ 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 d_1'.

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 three approaches for translating constraints on defender payoff into constraints on defender coverage. These approaches (ORIGAMI-M, ORIGAMI-M-BS, and DIRECT-MIN-COV) achieve this translation by computing the minimum coverage needed to satisfy a set of lower bound constraints $\mathbf{b}$ such that $U_i^d(\mathbf{c}) \ge b_i$, for $1 \le i \le n$. This minimum coverage is then added to the MILP in Figure 5 as constraints on the variable $\mathbf{c}$, reducing the feasible region and leading to significant speedup as verified in experiments.

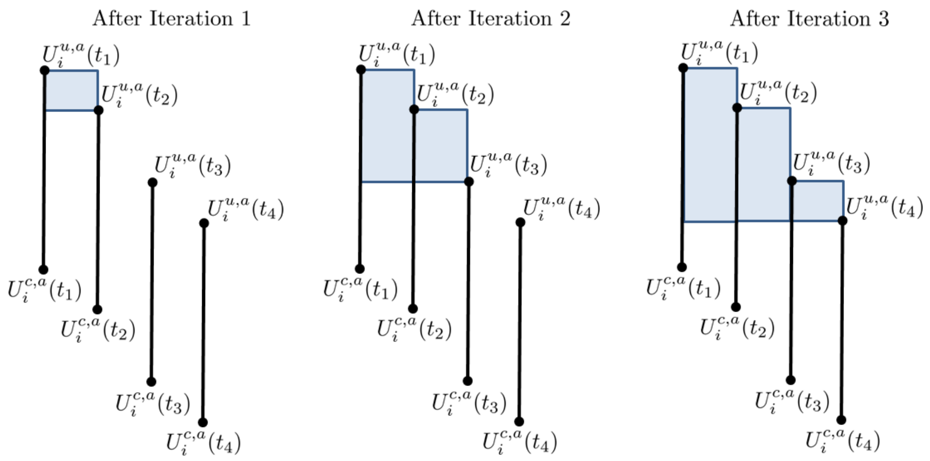

Fig. 7 Example of ORIGAMI-M incrementally expanding the attack set by increasing coverage.

7.1 ORIGAMI-M

ORIGAMI-M (Algorithm 3), is a modified version of the ORIGAMI algorithm [22] and borrows many of its key concepts. The "M" in the algorithm name refers to the fact that ORIGAMI-M is designed for security games with multiple objectives. 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. The goal is to update $\mathbf{c}$ such that it uses the minimum amount of defender resources to satisfy the constraints in $\mathbf{b}$. 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 constraints one objective at a time, the constraints for other objectives that were satisfied in previous iterations may become unsatisfied again. The reason is that the additional coverage may alter the targets selected for attack by one or more attacker types, possibly reducing the defender's payoff for those objectives below their once satisfied constraints. Therefore, all of the constraints in $\mathbf{b}$ must be checked repeatedly until there are no violated constraints. If all m defender resources are exhausted before $\mathbf{b}$ is satisfied, then the CSOP is infeasible.

The process for calculating the minimum coverage for a single constraint b_i is built on two assumption of security games [22]: (1) the attacker chooses the target that is optimal with respect its own payoffs; (2) if multiple targets are optimal, the attacker breaks ties by choosing the target that yields the highest defender payoff. The first property intuitively establishes that the attacker is a fully rational decision maker. The second property may seem less intuitive given the adversarial nature of the defender and the attacker. In theory, the player acting first in a Stackelberg game may force the adversary to play specific inducible actions in the follower's optimal set of actions by the threat of a slight perturbation of the optimal strategy, as described in [39]. In practice, the assumption that the attacker breaks ties in favor of the defender has been used in a number of real-world applications of Stackelberg security games. There has been work to remove these assumptions with models that consider uncertainty about the attacker, such as the imperfect rationality of human decision making [33,40]. However, we focus on the base model with standard assumptions for our initial multi-objective work and leave extensions for handling these types of uncertainty to future work.

The set of optimal targets for attacker type i, given coverage $\mathbf{c}$, is referred to as the attack set, $\mathit{\Gamma}_i(\mathbf{c})$. Accordingly, adding coverage on target $t \notin \mathit{\Gamma}_i$ does not affect the attacker type i's strategy or payoff. Thus, if $\mathbf{c}$ does not satisfy b_i, we only consider adding coverage to targets in $\mathit{\Gamma}_i$. $\mathit{\Gamma}_i$ can be expanded by increasing coverage such that the payoff for each target $t \in \mathit{\Gamma}_i$ is equivalent to the payoff for the target $t' \notin \mathit{\Gamma}_i$ with the highest payoff as defined by $U_i^a(c_{t'}, t')$. Adding an additional target to the attack set can only benefit the defender since the defender receives the optimal payoff among targets in the attack set.

Figure 7 shows a simple example of ORIGAMI-M with four targets. The vertical axis is the payoff for attacker type i, $U_i^a(\mathbf{c})$, while each target t is represented as the range $[U_i^{c,a}(t), U_i^{u,a}(t)]$. The blue rectangles depict the amount of coverage placed on each target. Before Iteration 1, the targets are sorted in descending order according to $U_i^a(\mathbf{c})$, resulting in the ordering $t_1 > t_2 > t_3 > t_4$ as well as $\mathit{\Gamma}_i = \{t_1\}$. After Iteration 1, enough coverage has been added to t_1 that $U_i^a(c_1, t_1) = U_i^{u,a}(t_2)$, meaning $\mathit{\Gamma}_i$ has been expanded to include t_2. In Iteration 2, coverage is placed on both t_1 and t_2 in order to push attacker type i's payoff for these targets down to $U_i^{u,a}(t_3)$, adding t_3 to $\mathit{\Gamma}_i$. The process is again repeated in Iteration 3 with coverage now being added to t_1, t_2, and t_3 until t_4 can be induced into $\mathit{\Gamma}_i$.

The idea for ORIGAMI-M is to expand the attack set $\mathit{\Gamma}_i$ until b_i is satisfied. Targets are added to $\mathit{\Gamma}_i$ in descending order according to attacker payoff, $U_i^a(c_t, t)$, which requires sorting the list of targets (Line 3). The attack set $\mathit{\Gamma}_i$ initially contains only the first target in this sorted list, while the variable next represents the size that the attack set will be expanded to. In order to add the next target to $\mathit{\Gamma}_i$, the attacker's payoff for all targets in $\mathit{\Gamma}_i$ must be reduced to $U_i^a(c_{\mathrm{next}}, t_{\mathrm{next}})$ (Line 12). 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 $\max_{1 \le t < \mathrm{next}} U_i^{c,a}(t) > U_i^a(c_{\mathrm{next}}, t_{\mathrm{next}})$ (Line 8), then it is impossible to induce attacker type i to choose target t_{next}. In that case, we can only reduce the attacker's payoff for targets in the attack set to $\max_{1 \le t < \mathrm{next}} U_i^{c,a}(t)$ (Line 9) and set the $\mathrm{noninducibleNextTarget}$ flag (Line 10). Then for each target $t \in \mathit{\Gamma}_i$, we compute the amount of additional coverage, $\mathrm{addedCov}[t]$, necessary to reach the required attacker payoff (Line 14). If the total amount of additional coverage exceeds the amount of remaining coverage (Line 15), denoted by variable $\mathrm{covLeft}$, then the $\mathrm{resourcesExceeded}$ flag is set (Line 16) and $\mathrm{addedCov}$ is recomputed with each target in $\mathit{\Gamma}_i$ being assigned a ratio of the remaining coverage so as to maintain the attack set (Line 18).

Once the $\mathrm{addedCov}$ vector has been computed, we check to see if $\mathbf{c} + \mathrm{addedCov}$ satisfies b_i (Line 19). If it does, there may exist a coverage $\mathbf{c}'$ which uses less defender resources and still satisfies b_i. To determine if this is the case, we developed a subroutine called MIN-COV, described in detail below, to compute $\mathbf{c}'$ (Line 20). If $\mathbf{c}' = \mathit{null}$, then $\mathbf{c} + \mathrm{addedCov}$ is the minimum coverage which satisfies b_i (Line 24), otherwise $\mathbf{c}'$ is the minimum coverage (Line 22). In either case, $\mathbf{c}$ is updated to the new minimum coverage and then compared against $\mathbf{b}$ to check for violated constraints (Line 2).

If $\mathbf{c} + \mathrm{addedCov}$ does not satisfy b_i, we know that further expansion of the attack set is necessary. Thus, $\mathbf{c}$ is updated to include $\mathrm{addedCov}$ (Line 29), the amount of coverage in $\mathrm{addedCov}$ is deducted from the running total of remaining coverage $\mathrm{covLeft}$ (Line 30), and next is incremented (Line 31). However, if either the $\mathrm{resourcesExceeded}$ or $\mathrm{noninducibleNextTarget}$ flag have been set (Line 26), then further expansion of the attack set is not possible. In this situation, b_i as well as the CSOP are infeasible and ORIGAMI-M terminates. If the attack set is expanded to include all targets (Line 32), i.e., $\mathrm{next} = |T| + 1$, then it may be possible to satisfy b_i if there is still defender resources remaining. Thus, we

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

```
1  c ← empty coverage vector ;
2  while $b_i > U_i^d(\mathbf{c})$ for some bound $b_i$ do
3      sort targets $T$ in decreasing order of value by $U_i^a(c_t, t)$;
4      covLeft ← $m - \sum_{t \in T} c_t$;
5      next ← 2;
6      while next ≤ $|T|$ do
7          addedCov[$t$] ← empty coverage vector;
8          if $\max_{1 \le t < \text{next}} U_i^{c,a}(t) > U_i^a(c_{\text{next}}, t_{\text{next}})$ then
9              $x \leftarrow \max_{1 \le t < \text{next}} U_i^{c,a}(t)$;
10             noninducibleNextTarget ← $true$;
11         else
12             $x \leftarrow U_i^a(c_{\text{next}}, t_{\text{next}})$;
13         for $1 \le t <$ next do
14             addedCov[$t$] ← $\frac{x - U_i^{u,a}(t)}{U_i^{c,a}(t) - U_i^{u,a}(t)} - c_t$;
15         if $\sum_{t \in T}$ addedCov[$t$] > covLeft then
16             resourcesExceeded ← $true$;
17             ratio[$t$] ← $\frac{1}{U_i^{u,a}(t) - U_i^{c,a}(t)}, \forall 1 \le t <$ next;
18             addedCov[$t$] = $\frac{\text{ratio}[t] \cdot \text{covLeft}}{\sum_{1 \le t \le \text{next}} \text{ratio}[t]}, \forall 1 \le t <$ next;
19         if $U_i^d(\mathbf{c} + \text{addedCov}) \ge b_i$ then
20             c′ ← MIN-COV($i$, c, b, next);
21             if c′ ≠ $null$ then
22                 c ← c′;
23             else
24                 c ← c + addedCov;
25             break;
26         else if resourcesExceeded ∨ noninducibleNextTarget then
27             return $infeasible$;
28         else
29             c ← c + addedCov;
30             covLeft −= $\sum_{t \in T}$ addedCov[$t$];
31             next++;
32     if next = $|T| + 1$ then
33         if covLeft > 0 then
34             c ← MIN-COV($i$, c, b, next);
35             if c = $null$ then
36                 return $infeasible$;
37         else
38             return $infeasible$;
39 return c ;
```

update $\mathbf{c}$ to the output generated by calling MIN-COV. If $\mathbf{c} = null$, then b_i is unsatisfiable and ORIGAMI-M returns infeasible, otherwise $\mathbf{c}$ is the minimum coverage.

If $\mathbf{c}^*$ is the coverage vector returned by ORIGAMI-M then Equation (8) of our MILP formulation can be replaced with $c_t^* \le c_t \le 1, \forall t \in T$. If, instead, ORIGAMI-M returns $infeasible$ then there is no feasible solution that satisfies $\mathbf{b}$ and thus there is no need to attempt solving the CSOP with Φ.

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

```
Input: Game index $i$, initial coverage $\mathbf{c}$, lower bound $\mathbf{b}$, size of expanded attack set $\mathrm{next}$;
$\mathbf{c}^* \leftarrow null$;
$\mathrm{minResources} \leftarrow m$;
$\mathrm{baseCov} \leftarrow \sum_{t \in T} c_t$;
for $1 \leq j < \mathrm{next}$ do
    $\mathrm{feasible} \leftarrow true$;
    $\mathbf{c}' \leftarrow \mathbf{c}$ ;
    $c'_j \leftarrow \frac{b_i - U_i^{u,a}(t_j)}{U_i^{c,a}(t_j) - U_i^{u,a}(t_j)}$;
    $c'_j \leftarrow \max(c'_j, c_j)$;
    if $c'_j > 1$ then
        break;
    $\mathrm{covSoFar} \leftarrow \mathrm{baseCov} + c'_j - c_j$;
    for $1 \leq k \leq |T|$ do
        if $j \neq k \wedge U_i^a(c'_{t_k}, t_k) > U_i^a(c'_{t_j}, t_j)$ then
            $c'_k = \frac{U_i^a(c'_{t_j}, t_j) - U_i^{u,a}(t_k)}{U_i^{c,a}(t_k) - U_i^{u,a}(t_k)}$;
            if $c'_k < c_k \vee c'_k > 1$ then
                $\mathrm{feasible} \leftarrow false$;
                break;
            $\mathrm{covSoFar} += c'_k - c_k$;
            if $\mathrm{covSoFar} \geq \mathrm{minResources}$ then
                $\mathrm{feasible} \leftarrow false$;
                break;
    if feasible then
        $\mathbf{c}^* \leftarrow \mathbf{c}'$;
        $\mathrm{minResources} \leftarrow \mathrm{covSoFar}$ ;
return $\mathbf{c}^*$
```

When MIN-COV (Algorithm 4) is called, we know that the coverage $\mathbf{c}$ induces an attack set of size $\mathrm{next}-1$ and does not satisfy b_i, while $\mathbf{c}+\mathrm{addedCov}$ induces an attack set of size next and satisfies b_i. Thus, MIN-COV is designed to determine if there exists a coverage $\mathbf{c}^*$ that uses more coverage than $\mathbf{c}$ and less coverage than $\mathbf{c} + \mathrm{addedCov}$ while still satisfying b_i. This determination can be made by trying to induce a satisfying attack on different targets and comparing the resulting coverage vectors. As $\mathbf{c} + \mathrm{addedCov}$ is the minimum coverage needed to induce an attack set of size next, we only need to consider attacks on the first $\mathrm{next}-1$ targets. Thus, for each target t_j, $1 \leq j < \mathrm{next}$ (Line 5), we generate the coverage vector $\mathbf{c}'$ that induces an attack on t_j and yields a defender payoff of at least b_i. MIN-COV returns $\mathbf{c}^*$ (Line 26), which represents the $\mathbf{c}'$ that uses the least amount of defender resources while satisfying b_i. The variable $\mathrm{minResources}$ denotes the amount of coverage used by the current minimum coverage and is initialized to m, the total number of defender resources.

For each coverage $\mathbf{c}'$, we initialize $\mathbf{c}'$ with $\mathbf{c}$ (Line 7) and then compute the coverage c_j on target t_j needed to yield a defender payoff of b_i (Line 8). We can never remove any coverage that has already been placed, so we ensure that $c'_j \geq c_j$ (Line 9). If $c'_j > 1$, then no valid coverage of t_j could satisfy b_i and thus there is no need to compute $\mathbf{c}'$ for t_j. Otherwise, we update the coverage for every other target t_k, $1 \leq k \leq |T|$ $j \neq k$. Placing c'_j coverage on t_j yields an attacker payoff $U_i^a(c'_j, t_j)$. Since our goal is to induce an attack on t_j, we must ensure that the attacker payoff for every t_k is no greater than for t_j, i.e.,

$U_i^a(c'_j, t_j) \geq U_i^a(c'_k, t_k)$, by placing additional coverage (Line 15). If either $c'_k < c_k$ or $c'_k > 1$ (Line 16) then no feasible coverage $\mathbf{c}'$ exists for t_j. The variable $\mathrm{covSoFar}$ tracks the amount of resources used by $\mathbf{c}'$, if at any point this value exceeds $\mathrm{minResources}$ then $\mathbf{c}'$ for t_j cannot be the minimum defender coverage (Line 20).

If the coverage for all targets t_k is updated successfully then we know that: (1) $\mathbf{c}'$ satisfies b_i and (2) $\mathbf{c}'$ is the current minimum coverage. For (1), we have ensured t_j is in the attack set Γ_i. By the properties of security games, the attacker will select the target $t \in \Gamma_i$ that yields the highest defender payoff. Thus, in the worst case from the defender's perspective, $t = t_j$ and gives the defender a payoff of at least b_i. Since $\mathrm{covSoFar}$ is compared to $\mathrm{minResources}$ everytime the coverage for a target is updated, (2) is inherently true if all targets have been updated. Having found a new minimum coverage, we update $\mathbf{c}^* \leftarrow \mathbf{c}'$ and $\mathrm{minResources} \leftarrow \mathrm{covSoFar}$.

7.2 Binary Search ORIGAMI-M

The ORIGAMI-M algorithm expands the attack set Γ_i one target at a time until either the current lower bound constraint is satisfied or determined to be infeasible. If the satisfying attack set is large, it may become computationally expensive to incrementally expand and evaluate the satisfiability of Γ_i. Thus, we introduced a modified version of ORIGAMI-M called ORIGAMI-M-BS (Algorithm 5)which uses binary search to find the minimum coverage vector $\mathbf{c}$ which satisfies the lower bound constraints in $\mathbf{b}$. Intuitively, for a violated constraint i, we are performing binary search to find the size of the smallest attack set which satisfies the lower bound constraint b_i. The natural range for the size of Γ_i is between 1 and $|T|$, therefore we use the respective bounds $\mathrm{lower} = 0$ and $\mathrm{upper} = |T| + 1$ for our binary search. The size of the attack set to be evaluated is determined by $\mathrm{next} = (\mathrm{upper}+\mathrm{lower})/2$. We record the size of the smallest satisfying attack set with μ, which is initially set to $|T|+1$. The coverage vector corresponding to the smallest satisfying attack set is $\mathbf{c}^+$ and is initialized to $null$.

For an attack set of a given size, the procedure for placing coverage on targets is identical to the procedure in ORIGAMI-M. The set of targets is sorted in descending order according to attacker payoff, $U_i^a(c_t, t)$ (Line 3). Then it is necessary to compute the vector of additional coverage, $\mathrm{addedCov}$, that must be added to the first $\mathrm{next} - 1$ targets so that Γ_i is expanded to include t_{next}. There are three possible scenarios when evaluating an attack set: (1) An attack set of size next cannot be induced due to either an insufficient amount of defender resources (Line 19) or a noninducible target (Line 12). Therefore, the smallest satisfying attack set must be smaller than size next so we update $\mathrm{upper} = \mathrm{next}$ (Line 24). (2) An attack set of size next can be induced but it does not satisfy the lower bound constraint b_i. Thus, we know that if a satisfying attack set exists it must be larger than size next so we update $\mathrm{lower} = \mathrm{next}$ (Line 31). (3) An attack set of size next can be induced and satisfies the lower bound constraint b_i (Line 25). While the current attack set is a satisfying attack set, it may be possible to find a smaller attack set which also satisfies b_i. Thus, we update $\mathrm{upper} = \mathrm{next}$ (Line 26) and if the current attack set is the smallest satisfying attack set found so far we update $\mathbf{c}^+ = \mathbf{c} + \mathrm{addedCov}$ (Line 27) and $\mu = \mathrm{next}$ (Line 28).

The binary search loop is repeated while $\mathrm{upper} - \mathrm{lower} > 1$ (Line 9). After loop termination, if $\mathbf{c}^+ = null$ and $\mathrm{upper} < |T| + 1$ (Line 32), then the constraint b_i is not satisfiable and the CSOP is infeasible (Line 39). We know this because upper is updated whenever an attack set either satisfies b_i (Line 26), exceeds the available resources (Line 24), and/or contains a noninducible target (Line 24). Thus, $\mathrm{upper} < |T| + 1$ would indicate that at least

```
Algorithm 5: ORIGAMI-M-BS($\mathbf{b}$)
1  $\mathbf{c} \leftarrow$ empty coverage vector ;
2  while $b_i > U_i^d(\mathbf{c})$ for some bound $b_i$ do
3      sort targets $T$ in decreasing order of value by $U_i^a(c_t, t)$;
4      covLeft $\leftarrow m - \sum_{t \in T} c_t$;
5      lower $\leftarrow 0$;
6      upper $\leftarrow |T| + 1$;
7      $\mu \leftarrow |T| + 1$;
8      $\mathbf{c}^+ \leftarrow null$;
9      while upper $-$ lower $> 1$ do
10         next $=$ (upper $+$ lower)$/2$;
11         addedCov$[t] \leftarrow$ empty coverage vector;
12         if $\max_{1 \le t < \text{next}} U_i^{c,a}(t) > U_i^a(c_{\text{next}}, t_{\text{next}})$ then
13             $x \leftarrow \max_{1 \le t < \text{next}} U_i^{c,a}(t)$;
14             noninducibleNextTarget $\leftarrow true$;
15         else
16             $x \leftarrow U_i^a(c_{\text{next}}, t_{\text{next}})$;
17         for $1 \le t <$ next do
18             addedCov$[t] \leftarrow \frac{x - U_i^{u,a}(t)}{U_i^{c,a}(t) - U_i^{u,a}(t)} - c_t$;
19         if $\sum_{t \in T}$ addedCov$[t] >$ covLeft then
20             resourcesExceeded $\leftarrow true$;
21             ratio$[t] \leftarrow \frac{1}{U_i^{u,a}(t) - U_i^{c,a}(t)}, \forall 1 \le t <$ next;
22             addedCov$[t] = \frac{\text{ratio}[t] \cdot \text{covLeft}}{\sum_{1 \le t \le \text{next}} \text{ratio}[t]}, \forall 1 \le t <$ next;
23         if resourcesExceeded $\vee$ noninducibleNextTarget then
24             upper $=$ next;
25         if $U_i^d(\mathbf{c} +$ addedCov$) \ge b_i$ then
26             upper $=$ next;
27             if next $< \mu$ then
28                 $\mathbf{c}^+ \leftarrow \mathbf{c} +$ addedCov;
29                 $\mu \leftarrow$ next;
30         else
31             lower $=$ next;
32     if $\mathbf{c}^+ \ne null \vee$ upper $= |T| + 1$ then
33         $\mathbf{c}' \leftarrow$ MIN-COV$(i, \mathbf{c}, \mathbf{b}, \mu)$;
34         if $\mathbf{c}' \ne null$ then
35             $\mathbf{c} \leftarrow \mathbf{c}'$;
36         else
37             $\mathbf{c} \leftarrow \mathbf{c}^+$;
38     else
39         return $infeasible$;
40 return $\mathbf{c}$;
```

one attack set was found to exceed defender resources or contain a noninducible target, but no satisfying attack set was found given that $\mathbf{c}^+ = null$. However, if $\mathbf{c}^+ = null$ and upper$= |T| + 1$, then it is still possible that a coverage satisfying b_i exists because it means the attack set has been expanded to the full set of targets and there is still remaining coverage. In this situation, as well as when $\mathbf{c}^+ \ne null$, MIN-COV is called to produce a coverage $\mathbf{c}'$ (Line 33). If $\mathbf{c}' \ne null$, then $\mathbf{c}'$ is the minimum coverage which satisfies b_i and we update

$\mathbf{c} \leftarrow \mathbf{c}'$ (Line 35). Otherwise, the coverage $\mathbf{c}^+$ found during the binary search is the minimum coverage and we update $\mathbf{c} \leftarrow \mathbf{c}^+$ (Line 37). The updated $\mathbf{c}$ is then checked for violated constraints (Line 2) and the entire process is repeated until either all constraints are satisfied or $\mathbf{b}$ is determined to be infeasible.

7.3 Direct MIN-COV

Both ORIGAMI-M and ORIGAMI-M-BS rely on the MIN-COV subroutine which is called when the smallest satisfying attack set is found. However, it is not necessary to first compute the satisfying attack set before calling MIN-COV. The only benefit of precomputing the attack set is to reduce the number of coverage vectors that must be computed in MIN-COV. The minimum coverage for satisfying $\mathbf{b}$ can be computed directly using MIN-COV, if we set the size of the attack set to be $|T| + 1$. In this way, MIN-COV will generate, for every target t, the coverage necessary to induce a satisfying attack on t. These coverages will be compared and the smallest, feasible, satisfying coverage will be selected. Thus, we introduced DIRECT-MIN-COV (Algorithm 6) which bypasses computing the smallest satisfying attack set and uses MIN-COV to compute the minimum coverage $\mathbf{c}$ needed to satisfy $\mathbf{b}$. Additionally, due to every target being considered for an attack there is no need to sort the targets by $U_i^a(c_t, t)$, as in ORIGAMI-M and ORIGAMI-M-BS. In all three algorithms (ORIGAMI-M, ORIGAMI-M-BS, and DIRECT-MIN-COV), MIN-COV is called only once for each violated constraint, the only difference being the number of coverage vectors computed. Despite DIRECT-MIN-COV having to generate more coverages via MIN-COV than either ORIGAMI-M or ORIGAMI-M-BS, the intuition is that there could be potential computational savings in not having to first compute Γ_i. As we show in Section 9, the fastest algorithm for computing lower bounds on the defender coverage depends on the specific properties of the MOSG such as the number of resources and targets.

Algorithm 6: DIRECT-MIN-COV($\mathbf{b}$)

```
1 c ← empty coverage vector ;
2 while $b_i > U_i^d(\mathbf{c})$ for some bound $b_i$ do
3     c ←MIN-COV($i, \mathbf{c}, \mathbf{b}, |T| + 1$);
4     if c = $null$ then
5         return $infeasible$;
6 return c ;
```

8 Approximate Approach

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 (Algorithm 7), an extension to these heuristics 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 any one of the three algorithms described in Section 7. We will use Ψ to refer to whichever algorithm (ORIGAMI-M, ORIGAMI-M-BS, or DIRECT-MIN-COV) is used as the subroutine in ORIGAMI-A. A series of feasibility problems is generated using binary search in order to approximate the optimal solution to the CSOP. This decomposition of the CSOP provides computational savings as we have developed efficient algorithms for solving the individual feasibility problems. Each of the three algorithms that can be used as a subroutine (Ψ) in ORIGAMI-A are polynomial in the number of targets, while the number of calls to Ψ by ORIGAMI-A is bounded by $\mathcal{O}(n \log r)$, where r denotes the length of the range formed by the objective values. Thus, ORIGAMI-A is polynomial in the size of the MOSG, while solving even a single iteration of lexicographic maximization for the exact MILP formulation is NP-hard, based on the result from [11] which proved the computational complexity of Bayesian security games. As a result, this algorithmic approach is much more efficient and the level of approximation between the computed solution and the Pareto optimal solution can be bounded.

Algorithm 7: ORIGAMI-A$(\mathbf{b}, \alpha)$

```
1  $\mathbf{c} \leftarrow$ empty coverage vector;
2  $b_1^+ \leftarrow \min_{t \in T} U_1^{u,d}(t)$;
3  $\mathbf{b}^+ \leftarrow \{b_1^+\} \cup \mathbf{b}$ ;
4  for $1 \leq i \leq n$ do
5      $lower \leftarrow b_i^+$;
6      $upper \leftarrow \max_{t \in T} U_i^{c,d}(t)$;
7      while $upper - lower > \alpha$ do
8          $b_i^+ \leftarrow \frac{upper + lower}{2}$;
9          $\mathbf{c}' \leftarrow \Psi(\mathbf{b}^+)$;
10         if $\mathbf{c}' = violated$ then
11             $upper \leftarrow b_i^+$;
12         else
13             $\mathbf{c} \leftarrow \mathbf{c}'$;
14             $lower \leftarrow b_i^+$;
15     $b_i^+ \leftarrow U_i^d(\mathbf{c})$;
16 return $\mathbf{c}$ ;
```

The subroutine Ψ is used to compute 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 use Ψ to determine if there exists a coverage vector $\mathbf{c}$ such that $\mathbf{b}^+$ is satisfied.

ORIGAMI-A finds an approximately optimal coverage vector $\mathbf{c}$ by using Ψ 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 Ψ. 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 Ψ, 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 $\rho(\epsilon, \alpha) \leq \max\{\epsilon, \alpha\}$. ■

9 Evaluation

The purpose of this section is to analyze how the choice of approach and properties of MOSGs impact both the runtime and solution quality of Iterative-ϵ-Constraints. We perform this 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 $U_i^{c,a}(t)$ are uniformly distributed integers between -1 and -10, for all targets. Unless otherwise mentioned, the default 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. ORIGAMI-M is the default subroutine used in ORIGAMI-A. 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 and include error bars showing standard error.

9.1 Runtime Analysis

This section evaluates how different factors (e.g., the number of targets) impact the time needed to generate the Pareto frontier using five different 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 analyzing the effect of the number of targets on runtime, we evaluate all five formulations for solving CSOPs. We then select ORIGAMI-A and the fastest MILP formulation, MILP-PM, to evaluate the effect of the remaining factors.

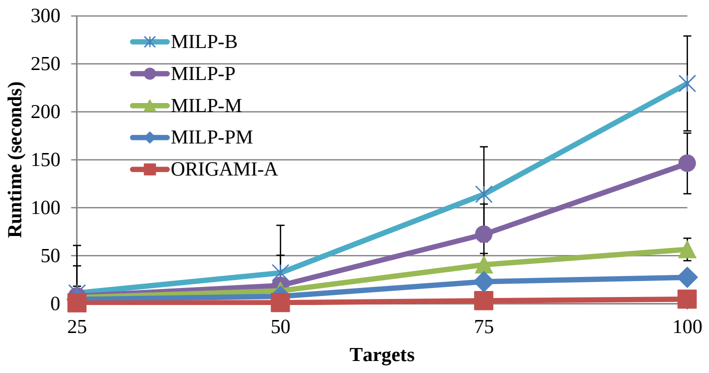

Fig. 8 Effect of target scale up on the runtime of Iterative-ϵ-Constraints with different CSOP solvers.

9.1.1 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 8, 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 greater than 50 times. Even compared to fastest MILP formulation, MILP-PM at 27.36 seconds, ORIGAMI-A still achieves a 6 times speedup. Additionally, since a small α value is used (0.001), there is only negligible loss in solution quality. A more detailed analysis of solution quality is presented in Section 9.3. T-test yields p-value < 0.001 for all comparisons of different formulations when there are 75 or 100 targets.

We conducted an additional set of experiments to determine how both 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 Figure 9, we can conclude 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.

9.1.2 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 exper-

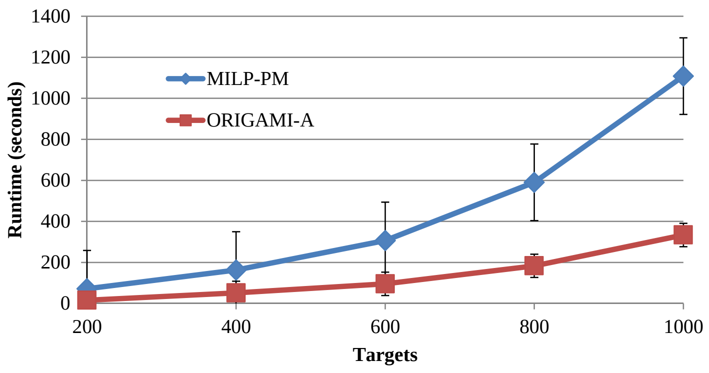

Fig. 9 Effect of additional target scale up on the runtime of Iterative-ϵ-Constraints with the most efficient exact CSOP solver (MILP-PM) and the approximate CSOP solver (ORIGAMI-A).

iments, we fixed the number of targets at 10. Figure 10 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. At 4 objectives, the runtime increases to 126 seconds for MILP-PM and 28 seconds for ORIGAMI-A. With 5 objectives, the separation between the two algorithm increases with respective runtimes of 917 and 669 seconds, with 7 trials with MILP-PM and 6 trials with ORIGAMI-A timing out after 1800 seconds. Whereas, with 6 objectives neither approach is able to generate the Pareto frontier before the runtime cap of 1800 seconds. The reason for this exponential runtime increase is two-fold. First, there is an increase in the number of generated solutions because the Pareto frontier now exists in a higher dimensional space. Second, each solution on the Pareto frontier takes longer to generate because the lexicographic maximization needed to solve a CSOP requires additional iterations. These results show that the number of objectives, and not the number of targets, is the key limiting factor in solving MOSGs.

9.1.3 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 11, results are shown for ϵ values of 0.1, 0.25, 0.5, 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 consists of 49 points, whereas for $\epsilon = 0.1$ that number increases 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.

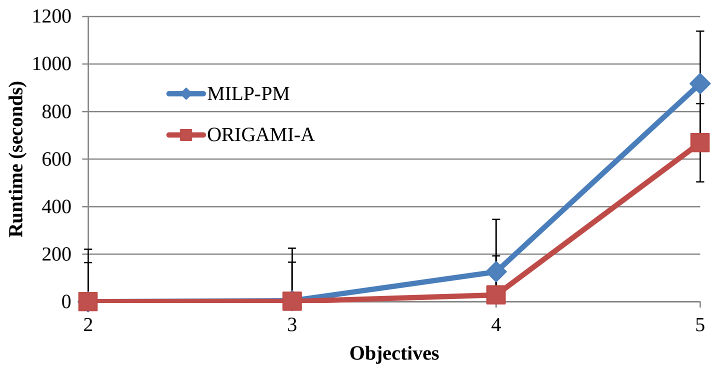

Fig. 10 Effective of objective scale up on the runtime of Iterative-ϵ-Constraints.

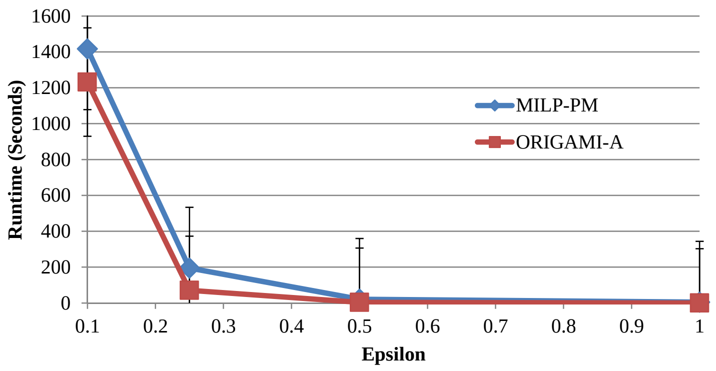

Fig. 11 Effect of epsilon on the runtime of Iterative-ϵ-Constraints.

9.2 Objective Similarity Analysis

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.

9.2.1 Effect of Objective Distribution

As the objective payoffs become similar, there is less conflict between the objectives. Less conflict means there is a reduction in the possible tradeoff between objectives, as it becomes increasingly likely that multiple objectives will be maximized simultaneously. As a result, the Pareto frontier is made up of fewer solutions, which means it can be generated more efficiently by Iterative-ϵ-Constraints.

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 12 shows the results for using ORIGAMI-A to solve MOSGs

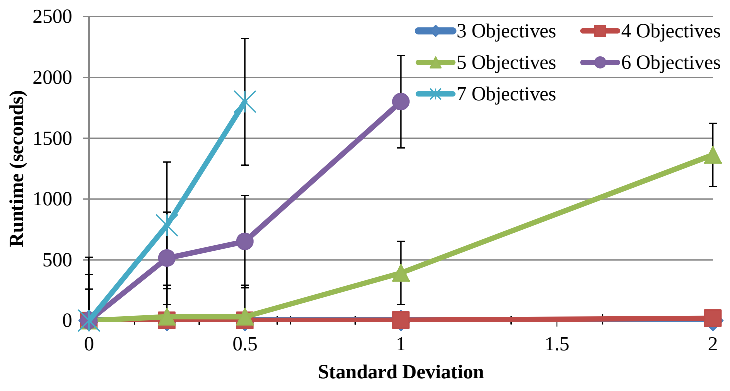

Fig. 12 Effect of objective similarity on the runtime of Iterative-ϵ-Constraints using ORIGAMI-A for a varying number of objectives.

with between 3 and 7 objectives using σ values of 0, 0.25, 0.5, 1.0, and 2.0. 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. With 5 objectives, a significant runtime increase is observed, going from an average of 32 seconds at $\sigma = 0.25$ to 1363 seconds at $\sigma = 2.0$. This effect is further amplified as the number of objectives is increased. At 6 objectives, Iterative-ϵ-Constraints is unable to finish within the 1800 second time limit with $\sigma > 1.0$, while the same is true for 7 objectives with $\sigma > 0.5$. We conclude that it is possible to scale to larger number of objectives if there is similarity, as defined in this section, between the attacker types.

9.2.2 Effect of Objective Clustering

In Section 9.2.1, the payoffs for each objective function are sampled from the same Gaussian distribution. This implies that all of the objective functions are related in their structure. However, there could be situations where one or more objectives are similar but other objectives are independently distributed. In this model, the set of related objectives can be viewed as forming a cluster while the remaining objectives are divergent from this cluster. A cluster is defined by two parameters. The first parameter is the number of objectives in the cluster as compared to the number of divergent objectives. A cluster size of 4 means that all of the objectives are in the cluster and thus all similar. In contrast, a cluster size of 1 implies that all objective functions are independently distributed. The second parameter is the value of σ which is the standard deviation defining the Gaussian distribution from which the objectives in the cluster are drawn, i.e., the degree of similarity between the related objectives. In Figure 13, we show the runtime results for MOSGs with 4 objectives for different cluster sizes and values of σ. We observe a trend in which the average runtime rises as the value of σ is increased. This is a logical result as larger values of σ mean that there is greater dissimilarity between the objectives within the cluster. When the cluster size is between 2 and 4, increasing σ always results in an increase in the runtime. When the cluster

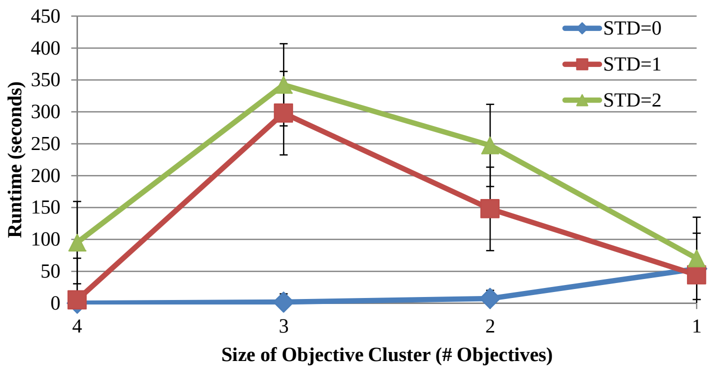

Fig. 13 Effect of objective clustering size on the runtime of Iterative-ϵ-Constraints using ORIGAMI-A for varying levels of intra-cluster Gaussian distribution.

contains only 1 objective, the runtimes for all values of σ are similar because all objectives are independently distributed.

Another trend we would expect to observe is that as the size of the cluster decreases, the runtime would increase as fewer objectives are similar and more are independently distributed. However, this trend only holds for $\sigma = 0$, when all of the objectives within the cluster are exactly identical. For $\sigma > 0$, we observe a substantially different runtime trend. With $\sigma = 1$ and $\sigma = 2$, the runtime starts low for clusters of size 4 and then increases dramatically when the size of the cluster is reduced to 3. Beyond 3 objectives, the runtime begins to decrease along with the cluster size until the runtime becomes similar for all values of σ at cluster size 1. It is counterintuitive that the worst runtimes are achieved with three similar objectives and one independently distributed objective. Upon close analysis of the experiment output files, the increase in runtime is the result of solving more CSOPs and having a larger Pareto frontier. In Figure 14, we can see that a comparison of the number of solutions in the Pareto frontier closely resembles the trends seen in the comparison of runtimes. Thus, one possible hypothesis could be that having three somewhat related objectives and one independently distributed objective allows for greater tradeoff between the objective payoffs than four independently distributed objectives.

9.3 Solution Quality Analysis

9.3.1 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 would be the most preferred solution, were it presented to the end user. In Section 5.3, 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

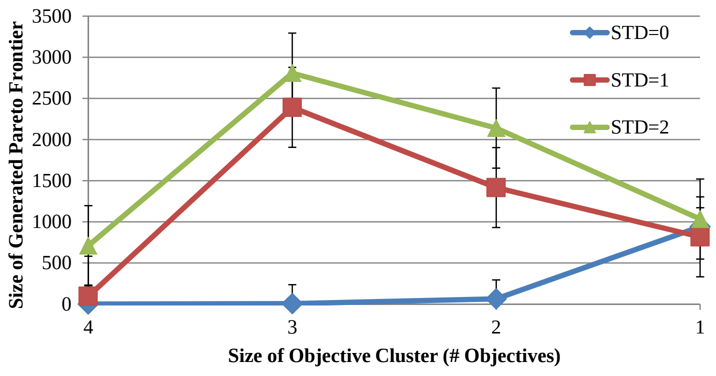

Fig. 14 Effect of objective clustering on size of the Pareto frontier generated by Iterative-ϵ-Constraints using ORIGAMI-A for varying levels of intra-cluster Gaussian distribution.

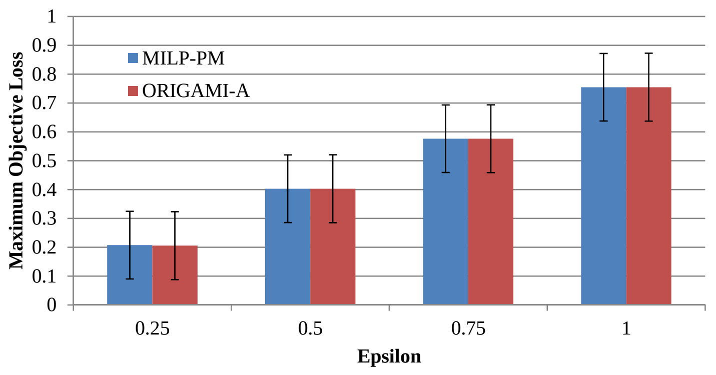

Fig. 15 Effect of epsilon on solution quality of the Pareto frontier generated by Iterative-ϵ-Constraints using MILP-PM and ORIGAMI-A compared against a Pareto frontier generated by MILP-PM using $\epsilon = 0.001$.

us to generate, thus we simulate the true frontier by using $\epsilon = 0.001$. Due to the computational cost associated with such a value of ϵ, we fix the number of objectives to 2. Figure 15 shows the results for ϵ values of 0.25, 0.5, 0.75, 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 0.75 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.

9.3.2 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

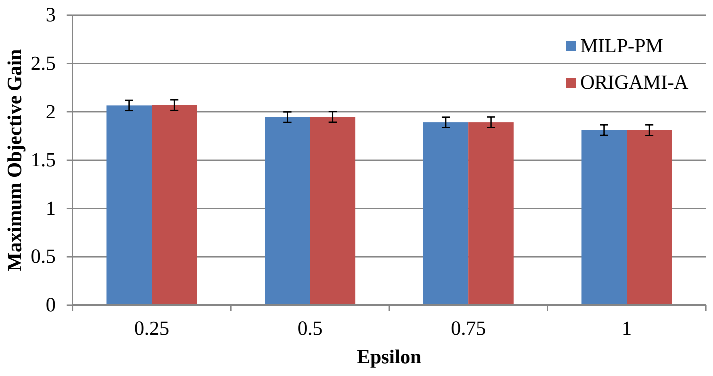

Fig. 16 Effect of epsilon on the benefit of the Pareto frontier generated by Iterative-ϵ-Constraints using MILP-PM and ORIGAMI-A over the single solution generated by a uniformly weighted Bayesian security game.

distribution. We conducted experiments to show the potential benefit of using MOSG over Bayesian security games in such cases. We computed the maximum objective gain produced by using a point in the Pareto frontier generated by Iterative-ϵ-Constraints as opposed to the Bayesian solution. 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 16 shows the results for ϵ values of 0.25, 0.5, 0.75, and 1.0. At $\epsilon = 1.0$, the maximum objective gain was 1.81 for both MILP-PM and ORIGAMI-A. Decreasing ϵ all the way to 0.25 increases the maximum objective gain by less than 15% for both algorithms. These results suggests that ϵ has limited impact on maximum objective gain, 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.

9.4 Constraint Computation Analysis

A key property of the ORIGAMI-M algorithm is that it computes the minimum coverage satisfying a vector $\mathbf{b}$ of lower bound constraints by attempting to satisfy one constraint at a time until no violated constraints remain. In the process of computing the additional coverage needed to satisfy the current constraint it is possible that previously satisfied constraints could become violated. It is important to understand the frequency with which this phenomenon occurs as it can have serious implications for the efficiency of the algorithm. Thus, we performed experiments which recorded the number of constraints that had to be satisfied for each call to ORIGAMI-M. The number of constraints is inherently linked to the number of objectives, thus we tested how the number of constraints computed was affected when scaling up the number of objectives. Figure 17 shows the average number of computed constraints for MOSGs with between 2 and 5 objectives and 10 targets. With 2 objectives, the number of constraints computed is 1.78, implying that on average ORIGAMI-M finds the minimal coverage with one pass through the constraints. Additionally, it means that there are situations where solving the first constraint results in a coverage which also satisfies the second constraint. For MOSGs with 5 objectives, the average number of computed constraints is 5.3 which again implies that ORIGAMI-M mostly requires just one pass through

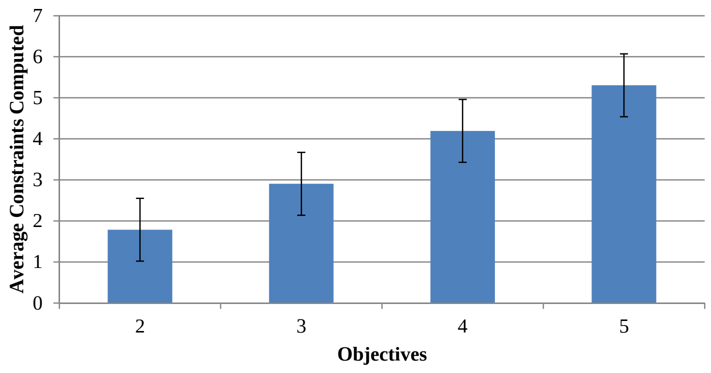

Fig. 17 Effect of objective scale up on the number of constraints computed per call to ORIGAMI-M for Iterative-ϵ-Constraints using ORIGAMI-A.

the constraints. However, it also indicates that there are instances where previously satisfied constraints become violated and must be recomputed. Fortunately, these violated constraints appear to be infrequent and do not seem to produce a cascading effect of additional violated constraints. These results suggest that ORIGAMI-M is able to efficiently compute the minimum coverage and is capable of scaling up to larger number of objectives.

9.5 Improved Pruning

In Section 5.2, we introduced two sets of pruning rules to improve the efficiency of Iterative-ϵ-Constraints. As shown in Section 9.1.2, the number of objectives is one of the key contributors to runtime when solving MOSGs. Thus, in order to perform a comparison, we evaluated each set of pruning heuristics as the number of objectives is increased. In Figure 18, we show results which demonstrate the impact of the improved pruning heuristic. The x-axis represents the number of objectives in the MOSG, while the y-axis represents the average runtime for Iterative-ϵ-Constraints to compute the Pareto frontier. For MOSGs with 2 or 3 objectives, there is little difference in the average runtimes between the original and improved pruning heuristics. When the number of objectives is increased to 4, the benefit of the improved pruning heuristic emerges, reducing the average runtime from 34.5 to 23.1 seconds. At 5 objectives the improved pruning heursitic results in significant computational savings, reducing the average runtime by almost 28% (813.8 versus 588.7 seconds). Even with the improved set of pruning heuristics, Iterative-ϵ-Constraintsis still not able to finish in under the 1800 second time limit. These results indicate that by further exploiting the concept of Pareto dominance, it is possible obtain modest runtime improvements.

9.6 ORIGAMI-A Subroutine Analysis

The ORIGAMI-A algorithm relies on ORIGAMI-M to compute the minimum coverage necessary to satisfy a set of lower bound constraints. ORIGAMI-M is a critical subroutine which is called multiple times for each CSOP, thus making efficiency paramount. In Figure 9, we showed the ability of ORIGAMI-M to scale up to large number of targets. However,

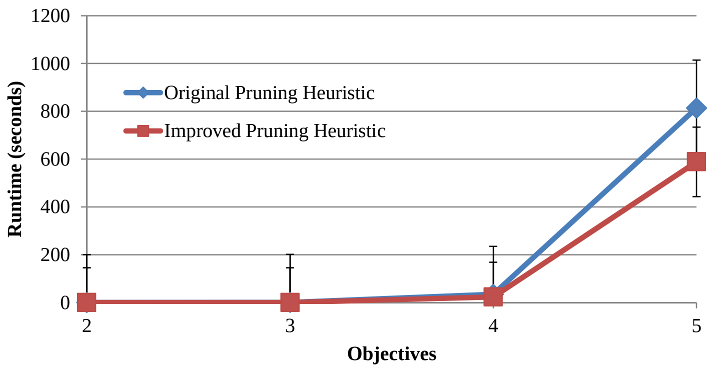

Fig. 18 Effect of pruning heuristic on the runtime of Iterative-ϵ-Constraints using ORIGAMI-A for a varying number of objectives.

any improvement to the subroutine used by ORIGAMI-A could lead to significant computation savings. Thus, in this section, we describe two approaches that either modify or replace ORIGAMI-M in an attempt to improve the efficiency of ORIGAMI-A.

9.6.1 Comparing the Effect of the Number of Targets

In Figure 19, we compare the ability of both ORIGAMI-M-BS and DIRECT-MIN-COV to scale up the number of targets as opposed to ORIGAMI-M. We evaluated the three algorithms for MOSGs with between 200 and 1000 targets. The x-axis indicates the number of targets in the MOSG, whereas the y-axis represents the average time needed to generate the Pareto frontier. The runtime results for ORIGAMI-M-BS are counterintuitive, as the inclusion of binary search fails to provide any improvement over ORIGAMI-M. In fact, for every number of targets tested the runtime for ORIGAMI-M-BS is greater than ORIGAMI-M. The difference in runtime between the two algorithms remains essentially constant at 2 seconds for each number of targets tested. This result suggests that despite having different formulations, ORIGAMI-M and ORIGAMI-M-BS are evaluating a similar number of attack sets. Additionally, the runtimes for DIRECT-MIN-COV are worse than either ORIGAMI-M or ORIGAMI-M-BS for every number of targets tested, except for ORIGAMI-M-BS at 200 targets. As the number of targets is increased, the disparity between the runtimes for the two ORIGAMI-M algorithms and DIRECT-MIN-COV widens.

9.6.2 Comparing the Effect of the Ratio of Defender Resources to Targets

We sought to better understand why neither of the two new proposed algorithms were able to improve upon the performance of ORIGAMI-M. In particular, we wanted to determine why incrementally expanding the attack set (ORIGAMI-M) was faster than performing binary search (ORIGAMI-M-BS), even for MOSGs with 1000 targets.

For all of our experiments, the ratio of defender resources to targets was fixed at $\frac{m}{|T|} = 0.2$. Intuitively, the higher this ratio is, the larger the average size of the attack set will be. With relatively more resources, the defender can place additional coverage so as to induce the attacker into considering a larger number of targets. Thus, the small $\frac{m}{|T|}$ ratio that we

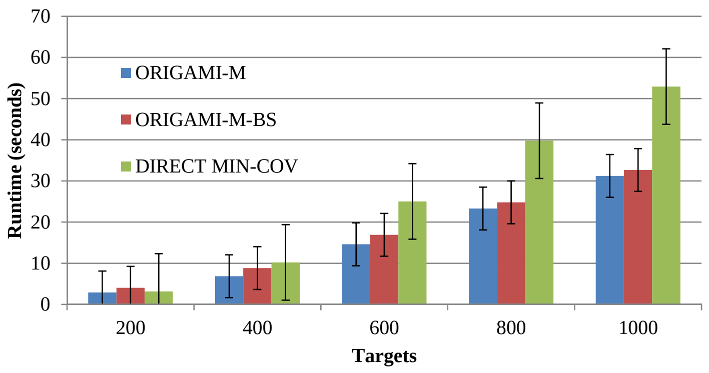

Fig. 19 Effect of ORIGAMI-A subroutine on the runtime of Iterative-ϵ-Constraints for a varying number of targets.

had been using previously meant the average size of the attack set would also be small. This greatly favors ORIGAMI-M which expands the attack set one target at time and returns as soon as it has found a satisfying attack set. In contrast, ORIGAMI-M-BS always evaluates $\log n$ attack sets regardless of the $\frac{m}{|T|}$ ratio. To evaluate the effect of $\frac{m}{|T|}$ on the performance of our three algorithms, we conducted experiments on MOSGs with 400 targets and $\frac{m}{|T|}$ ratios ranging between 0.2 and 0.8. In Figure 20, we show the results for this set of experiments. The x-axis indicates the $\frac{m}{|T|}$ ratio, whereas the y-axis indicates the average time to generate the Pareto frontier. A clear pattern emerges from these results: (1) if $\frac{m}{|T|} < 0.5$ then the ordering of the algorithms from most to least efficient is ORIGAMI-M, ORIGAMI-M-BS, DIRECT-MIN-COV; (2) if $\frac{m}{|T|} \geq 0.5$ then the ordering is reversed to DIRECT-MIN-COV, ORIGAMI-M-BS, ORIGAMI-M. What is interesting is that ORIGAMI-M-BS is never the optimal algorithm. If $\frac{m}{|T|}$ is small then it is better to incrementally expanding the attack set using ORIGAMI-M, whereas when $\frac{m}{|T|}$ is large it is more efficient to not precompute the smallest satisfying attack set as in DIRECT-MIN-COV. This result suggests that the optimal subroutine for ORIGAMI-A is dependent on the underlying properties of the MOSG and thus could vary from domain to domain.

Additionally, there is a discernible trend across all three algorithms as the value of $\frac{m}{|T|}$ is varied. Specifically, the average runtime as a function of $\frac{m}{|T|}$ resembles a bell curve centered at $\frac{m}{|T|} = 0.6$. This is a result of the combinatorial nature of placing coverage on targets. Therefore, when $\frac{m}{|T|} = 0.2$ there are significantly more targets than defender resources and there is only so much that can be done to prevent attacks. Since there are fewer ways to configure the coverage, the Pareto frontier contains fewer solutions. At the other extreme, when $\frac{m}{|T|} = 0.8$ the amount of defender resources is essentially equivalent to the number of targets. It is then possible to generate a coverage which maximizes all objectives simultaneously, leading to a Pareto frontier consisting of a single solution. Then as $\frac{m}{|T|}$ approaches 0.6 from either direction the runtime increases as there are more ways to place coverage and thus more solutions in the Pareto frontier. Due to the large number of possible defender coverages to consider, each individual CSOP also takes longer to solve, which is a phenomenon that has also been observed in single objective security games as described in [19].

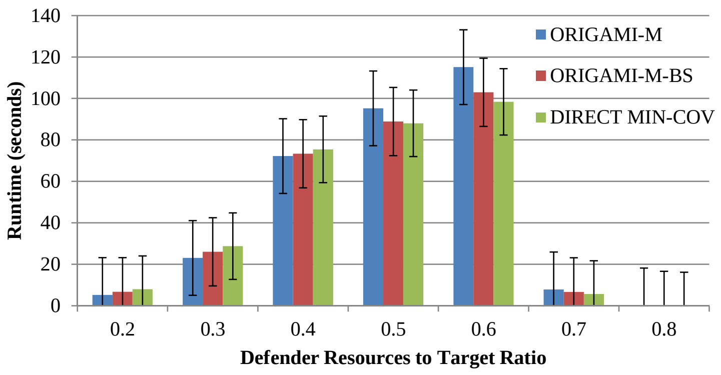

Fig. 20 Effect of ORIGAMI-A subroutine on the runtime of Iterative-ϵ-Constraints for varying resource-target ratios.

10 Visualization

The goal of our research is to provide decision support for decision-makers faced with multi-objective optimization problems. As mentioned previously, solving a multi-objective optimization problem involves generating the Pareto frontier. Once the Pareto frontier has been obtained, it must still be presented to the end user who then selects one of the candidate solutions based on their preferences, background knowledge, etc. One challenge associated with multi-objective optimization is how to present information about the Pareto frontier to the user so as to best facilitate their decision-making process. The most naïve approach is to present the contents of the Pareto frontier in a tabular format. However, this approach suffers from one significant drawback, a lack of visualized spatial information. A table cannot convincingly convey the shape and structure of the Pareto frontier as well as the tradeoff between different objectives and solutions. Thus, visualization is an important component for presenting the Pareto frontier to the user.

In Section 2, we highlighted the Los Angeles rail system as a motivating domain for MOSGs. To recall, the LASD is responsible for protecting 70 stations in the rail system against three potential attacker types: ticketless travelers, criminals, and terrorists. We use the LASD domain as a case study to compare different methods for visualization in security domains, which is only possible using our algorithms for calculating the Pareto frontier.

We model the LASD domain as an MOSG with 3 objectives, 70 targets, and 14 defender resources. Iterative-ϵ-Constraints with $\epsilon = 1.0$ was then used to generate the Pareto frontier which contained 100 solutions. It is this Pareto frontier that we use to compare the different visualization techniques.

10.1 Euclidean Plots

The elements of the Pareto frontier exist in an n-dimensional space, where n is the number of objectives. Visualizing the Pareto frontier for $n = 2$ is intuitive as solutions can be represented in two-dimensional Euclidean space, as shown in Figure 2, by the payoffs obtained for each objective. This approach allows the tradeoff between the two objectives to be directly observed in a comprehensible form. An advantage of using Euclidean plots is

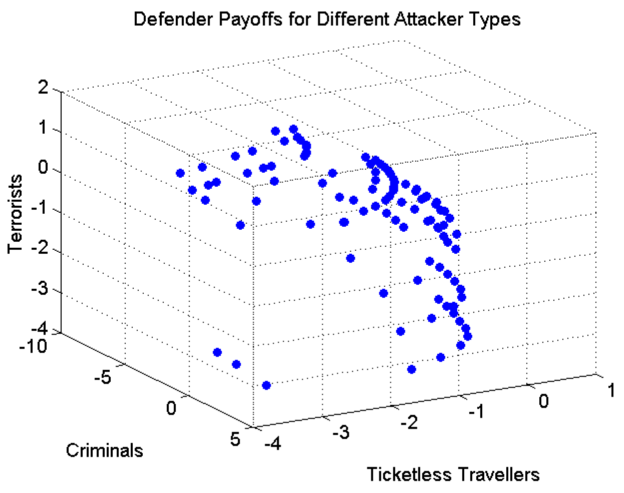

Fig. 21 Euclidean plot for the LASD domain.

that because the solutions are represented as points, the plots can display a large number of solutions without overwhelming the user. For $n = 3$ the Pareto frontier can still be plotted in Euclidean space. In Figure 21, the sample Pareto frontier from the LASD domain is visualized in three-dimensional Euclidean space. This example illustrates one of the drawbacks of using a Euclidean plot for $n = 3$. It is difficult to evaluate the tradeoffs in payoff for defending against ticketless travelers, criminals, and terrorists based on a single figure. Thus, interactive components such as animation or figure manipulation become necessary and present an additional barrier to the user's understanding.

10.2 Scatter Plots

One of the standard methods for visualizing the Pareto frontier is the scatter plot matrix [38], where n dimensions are visualized using $\binom{n}{2}$ two dimensional scatter plots, in which each pair of dimensions has a scatter plot showing their relation. With each scatter plot, the end user is able to gain a fundamental understanding of the tradeoffs between the payoffs for the two objectives. Similar to Euclidean plots, scatter plots are capable of efficiently displaying a large number of solutions. One extension on the standard bi-objective scatter plot is the addition of a third color dimension [29], resulting in $\binom{n}{3}$ possible scatter plots. This color dimension can be represented as either a continuous gradient or as a discrete set of colors mapping to specific segments of the possible objective values. Examples of both bi-objective and tri-objective (with discrete coloring) scatter plots for the LASD domain can be seen in Figures 22 and 23, respectively. For the LASD domain, the tri-objective scatter plot matrix is preferable because the entire Pareto frontier can be visualized in a single figure, rather than the three figures required for the bi-objective scatter plot matrix. This eliminates the need for the end user to synthesize data between multiple scatter plots in order to obtain the global perspective. For both approaches, the decision making process becomes more difficult as the number of objectives is increased due to the polynomial number of scatter plots that must be generated.

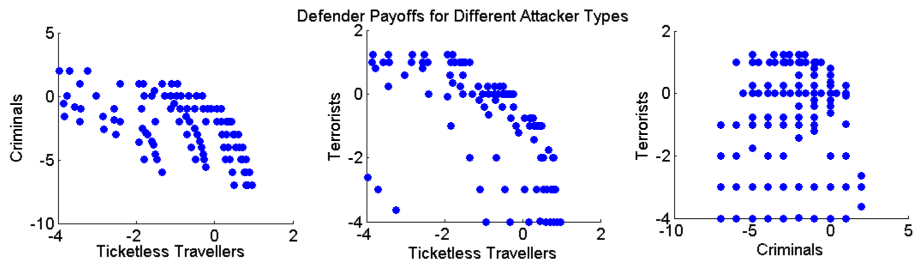

Fig. 22 Bi-objective scatter plot matrix for LASD domain.

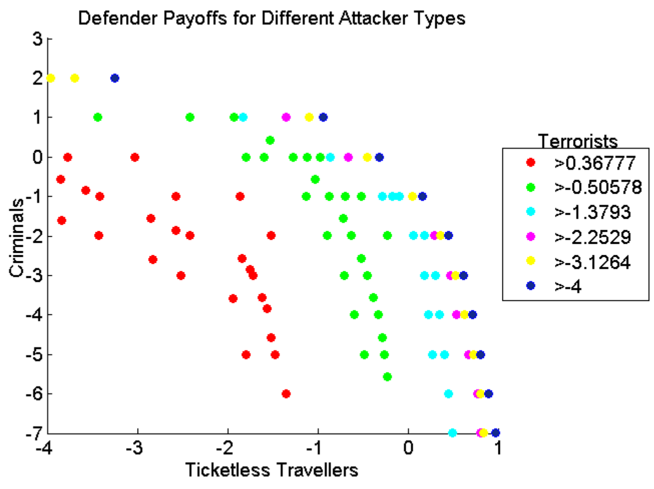

Fig. 23 Tri-objective scatter plot matrix for LASD domain.

10.3 Parallel Coordinates

Parallel Coordinates [17] is another common approach used for visualizing the Pareto frontier. In this approach, n parallel lines are used to represent the range of values for each objective. A Pareto-optimal solution is displayed as a polyline that intersects each parallel line at the point corresponding to the payoff received for that objective. Figure 24 shows the Pareto frontier for the LASD domain using the Parallel Coordinates approach. The main advantage of Parallel Coordinates is that the entire Pareto frontier, regardless of dimensionality, can be presented in a single figure. This eliminates any issues associated with having to process data from multiple sources. However, due to the usage of polylines rather than points, the Pareto frontier can become incomprehensible to the user if the number of solutions in the Pareto frontier is large. This is an issue for the LASD domain because the Pareto frontier consists of 100 candidate solutions, making it difficult to distinguish each individual solution. The number of Pareto optimal solutions can be influenced during processing by adjusting the value of ϵ as well as during post-processing by employing a filter to prevent certain solutions from being displayed. However, the number of solutions may need to be dramatically reduced before the Pareto frontier becomes comprehensible.

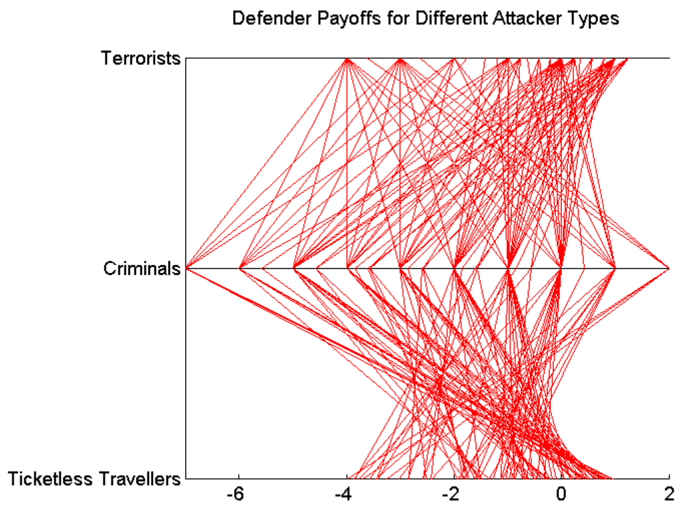

Fig. 24 Parallel coordinates for LASD domain.

10.4 Overall Trends

There is currently no one-size-fits-all visualization approach, the appropriate technique must be determined for each domain based on factors such as the number of objectives and the size of the Pareto frontier. For example, scatter plot matrices are better suited to situations where the dimensionality of the Pareto frontier is low but the number of solutions it contains is high, whereas Parallel Coordinates is better suited to situations with high dimensionality but fewer candidate solutions.

Based on the properties of the domain, we conclude that tri-objective scatter plot is the best approach for visualizing the Pareto frontier of the LASD MOSG because it allows for the most compact and coherent visual representation. It captures the entire Pareto frontier in a single figure which should be intuitive even for non-technical decision makers. By generating and visualizing the Pareto frontier in this way, LASD can gain a significant amount of knowledge about their domain and the tradeoffs that exist between different security strategies. This can be more insightful than finding a single solution, even if it were generated using well thought out weightings for the objectives. Finally, since the tri-objective scatter plot does not rely on animation or manipulation, information about the Pareto frontier can be disseminated easily to large groups and included in printed reports.

We have demonstrated the ability to visualize the Pareto frontier for the LASD domain which has 3 objectives. As the dimensionality of the objective space increases, the Pareto frontier naturally becomes more complex and difficult to understand. However, for most multi-objective optimization problems the total number of objectives is relatively small ($n \leq 5$). Even for domains which require large number of objectives, it may be possible to reduce the dimensionality of the Pareto frontier in order to focus the decision making process only on the most salient objectives. Dimension reduction is possible in two situations: (1) some objectives are insignificant in that their range of Pareto-optimal values is small; (2) there exists a strong correlation between multiple objectives. This reduction is typically performed using machine learning techniques with the most common approach being Principal Component Analysis (PCA) [21]. So if, in the future, LASD requires a higher fidelity model with more attacker types, it may become necessary to use such dimension reduction techniques in order to visualize the Pareto frontier.

11 Conclusion

We draw 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. Instead of a single optimal solution, MOSGs have a set of Pareto-optimal (non-dominated) solutions, known as the Pareto frontier, which represents the space of trade offs between the objectives. A single Pareto optimal solution can be found by solving a CSOP for a given set of constraints $\mathbf{b}$. The Pareto frontier is then generated by solving multiple CSOPs produced by modifying the constraints in $\mathbf{b}$. The contributions presented in this paper 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; (iii) heuristics that achieve speedup by exploiting the structure of security games to further constrain the MILP; (iv) an approximate approach for solving a CSOP built off those same heuristics, increasing the scalability of our approach with quality guarantees. Additional contributions of this paper include proofs on the level of approximation, detailed experimental evaluation of the proposed approaches and heuristics, as well as a discussion on techniques for visualizing the Pareto frontier.

Now that we have demonstrated that generating and analyzing the Pareto frontier is a viable solution concept for multi-objective security games, we plan to further extend our MOSG model in the future. One possible direction to explore is having multiple objectives for the attacker. This could model situations where the attacker explicitly considers multiple criteria when selecting a target, such economic significance, political significance, cost to attack, etc. As a result, the problem becomes even more difficult for the defender, as it is unknown what process the attacker is using to weigh the objectives in order to select a target. Such an extension may require the development of new solution concepts that rely on robust optimization techniques. Another possible direction to investigate is irrational behavior in attackers. In the current MOSG model, full rationality for the defender and all attackers is assumed. However, in practice we know that humans are not fully rational or strictly utility maximizing. Thus, if we wish to build robust model suitable for real world deployment then we must account for this irrationality. Work has been done in this area for single-objective security games [33,40], which we would seek to extend to the multi-objective case. However, one immediate consequence is that ORIGAMI-M, ORIGAMI-M-BS, and DIRECT-MIN-COV all rely on full rationality and thus would either need to be modified or replaced. These extensions will result in a higher fidelity MOSG model that is applicable to an even larger, more diverse set of domains.

12 Acknowledgement

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

References

1. M. Abido. Environmental/economic power dispatch using multiobjective evolutionary algorithms. *IEEE Transactions on Power Systems*, 18(4):1529–1537, 2003.
2. M. J. Alves and J. Clmaco. A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research*, 180(1):99 – 115, 2007.
3. 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.
4. N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 57–64, 2009.
5. W. Brauers, E. Zavadskas, F. Peldschus, and Z. Turskis. Multi-objective decision-making for road design. *Transport*, 23(3):183–193, 2008.
6. K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner. Approximation-guided evolutionary multi-objective optimization. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1198–1203, 2011.
7. M. Brown, B. An, C. Kiekintveld, F. Ordonez, and M. Tambe. Multi-objective optimization for security games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.
8. V. Chankong and Y. Haimes. *Multiobjective decision making: theory and methodology*. North-Holland New York, 1983.
9. C. Coello, G. Lamont, and D. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer-Verlag New York Inc, 2007.
10. V. Conitzer and D. Korzhyk. Commitment to correlated strategies. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 632–637, 2011.
11. V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *ACM Conference on Electronic Commerce*, pages 82–90, 2006.
12. K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, 2001.
13. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.
14. M. Giuliano and M. Johnston. Multi-objective evolutionary algorithms for scheduling the James Webb space telescope. In *International Conference on Automated Planning and Scheduling (ICAPS)*, volume 8, pages 107–115, 2008.
15. Y. Haimes, L. Lasdon, and D. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(3):296–297, 1971.
16. C. Hwang and A. Masud. *Multiple objective decision making, methods and applications: a state-of-the-art survey*. Springer-Verlag Berlin, Heidelberg, 1979.
17. A. Inselberg. Parallel coordinates for visualizing multidimensional geometry. *New Techniques and Technologies for Statistics*, pages 279–288, 1997.
18. H. Iseki, A. Demisch, B. Taylor, and A. Yoh. *Evaluating the Costs and Benefits of Transit Smart Cards*. California PATH Research Report, Institute of Transportation Studies, University of California at Berkeley, 2008.
19. M. Jain, K. Leyton-Brown, and M. Tambe. Where the Hard Security Problems Are? In *AAAI Spring Symposium*, 2012.
20. 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.
21. I. Jolliffe. *Principal Component Analysis*. Springer New York, 2002.
22. C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordonez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 689–696, 2009.
23. I. Kim and O. de Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, 29:149–158, 2005.
24. D. Korzhyk, V. Conitzer, and R. Parr. Security games with multiple attacker resources. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 273–279, 2011.
25. S. Kukkonen and J. Lampinen. Gde3: The third evolution step of generalized differential evolution. In *IEEE Congress on Evolutionary Computation*, volume 1, pages 443–450, 2005.
26. 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.

27. D. Li, J. Yang, and M. Biswal. Quantitative parametric connections between methods for generating noninferior solutions in multiobjective optimization. *European journal of operational research*, 117(1):84–99, 1999.
28. M. Lightner and S. Director. Multiple criterion optimization for the design of electronic circuits. *IEEE Transactions on Circuits and Systems*, 28(3):169 – 179, mar 1981.
29. A. Lotov, V. Bushenkov, and G. Kamenev. *Interactive decision maps: Approximation and visualization of Pareto frontier*. Springer Netherlands, 2004.
30. M. Luque, K. Miettinen, P. Eskelinen, and F. Ruiz. Incorporating preference information in interactive reference point methods for multiobjective optimization. *Omega*, 37(2):450 – 462, 2009.
31. G. Mavrotas. Effective implementation of the ϵ-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455–465, 2009.
32. 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 *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 895–902, 2008.
33. J. Pita, M. Jain, F. Ordez, M. Tambe, S. Kraus, and R. Magori-Cohen. Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2009.
34. S. Pohekar and M. Ramachandran. Application of multi-criteria decision making to sustainable energy planninga review. *Renewable and Sustainable Energy Reviews*, 8(4):365–381, 2004.
35. R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application*. Robert E. Krieger Publishing Company, 1989.
36. R. Tappeta and J. Renaud. Interactive multiobjective optimization procedure. *AIAA Journal*, 37(7):881–889, 1999.
37. A. Toffolo and A. Lazzaretto. Evolutionary algorithms for multi-objective energetic and economic optimization in thermal system design. *Energy*, 27(6):549–567, 2002.
38. J. van Wijk and R. van Liere. Hyperslice: visualization of scalar functions of many variables. In *Visualization*, pages 119–125. IEEE Computer Society, 1993.
39. B. von Stengel and S. Zamir. Leadership with commitment to mixed strategies. Technical Report LSE-CDAM-2004-01, CDAM Research Report, 2004.
40. R. Yang, F. Ordonez, and M. Tambe. Computing optimal strategy against quantal response in security games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.
41. L. Zadeh. Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, 8(1):59–60, 1963.
42. E. Zitzler, M. Laumanns, and L. Thiele. *SPEA2: Improving the strength Pareto evolutionary algorithm*. TIK-Report 103. Swiss Federal Institute of Technology (ETH) Zurich, Computer Engineering and Networks Engineering (TIK), 2001.