*ECAI 2024*
*U. Endriss et al. (Eds.)*

*doi:10.3233/FAIA240873*

# Escape Sensing Games: Detection-vs-Evasion in Security Applications

**Niclas Boehmer**[a,*,1], **Minbiao Han**[b,1], **Haifeng Xu**[b] **and Milind Tambe**[a]

[a]Harvard University
[b]University of Chicago

**Abstract.** Traditional game-theoretic research for security applications primarily focuses on the allocation of external protection resources to defend targets. This work puts forward the study of a new class of games centered around strategically *arranging targets* to protect them against a constrained adversary, with motivations from varied domains such as peacekeeping resource transit and cybersecurity. Specifically, we introduce Escape Sensing Games (ESGs). In ESGs, a blue player manages the order in which targets pass through a channel, while her opponent tries to capture the targets using a set of sensors that need some time to recharge after each activation. We present a thorough computational study of ESGs. Among others, we show that it is NP-hard to compute best responses and equilibria. Nevertheless, we propose a variety of effective (heuristic) algorithms whose quality we demonstrate in extensive computational experiments.

## 1 Introduction

The past decade has witnessed an influential line of research in AI, particularly multiagent systems (MAS), that employs computational game theory to tackle critical challenges in security and public safety applications, ranging from protecting national ports [20] to combating smuggling [7] and illegal poaching [14] to defending our cyber systems [26]. At the core of almost all of these problems is to optimize the allocation of (often limited) *external forces* to protect critical targets. In this work, we adopt a similar computational game theory approach but address a fundamentally different type of security challenge that looks to improve security via *optimizing the arrangement of targets* in the face of adversaries. This research contributes a novel perspective to game-theoretic security strategies, emphasizing target arrangement as a defense mechanism against adversaries.

Specifically, we introduce and study *Escape Sensing Games* (ESGs). In these games, a *blue* player aims to securely navigate a set of *targets* through a channel, whereas her opponent, the *red* player, controls a set of *sensors* along the channel and tries to sense (and therefore "steal") as many targets as possible. This model captures strategic interactions arising in various domains. One example is the transportation of peacekeeping resources using a convoy of ships or cars over a fixed route with malicious actors (e.g., pirates or hostile forces) trying to intercept them [24] (see Section 2 for details). In cybersecurity, the blue player could model a network administrator routing sensitive data packets through a network with an attacker trying to intercept them. Our model captures strategic interactions in these settings arising when security measures are either unavailable or have already been allocated and the blue player is only left with scheduling the targets to avoid detection by the attacker.

We study the optimal sequential play in ESGs, where the blue player first commits to an ordering of targets followed by the red player devising an optimal sensing plan. Herein, sensors' capabilities are limited in two ways. First, each sensor is only capable of sensing certain targets, modeling that detection and interception technologies are not uniformly effective across different targets, due to differing characteristics such as size, speed, or defense mechanisms. Second, sensors need a certain time to recharge after sensing a target, modeling limits inherent in detection and interception systems, where permanent action is not feasible.

There are certain challenges integral to our model that make the computation of equilibria highly non-trivial. First, the action space of both players has an exponential size, rendering standard solution approaches such as support enumeration computationally infeasible. Second, also after the strategies of both players have been fixed, the game evolves in a complex, sequential fashion with targets moving one after each other through the channel. Connected to this, third, it turns out that the red player's best response problem of coming up with an optimal sensing plan given a target ordering is already NP-hard. Consequently, this paper also contributes to the algorithmic research on computationally challenging games, a fairly unexplored topic outside of combinatorial game theory [10, 19, 12].

### *1.1 Our Contribution*

We contribute a new perspective to the rich literature on computational game theory for security applications through our study of the previously overlooked problem of target arrangement. Specifically, we introduce and analyze Escape Sensing Games with a focus on the target-controlling blue player. We demonstrate that solving this game is highly complex, as we prove that it is NP-hard for both players to compute their optimal strategies. To nevertheless be able to solve ESGs in practice, we devise algorithms for computing the red player's strategy, which turn out to scale well in our experiments. Computing the blue player's strategy and thereby the game's Stackelberg equilibrium turns out to be a much more intricate task. Our experiments show that our formulation of the problem as a bilevel program is only capable of solving small instances of the game exactly. Motivated by this, we present a heuristic that effectively combines simulated annealing with a greedy heuristic and an Integer Linear Program (ILP) for computing the red player's strategy. We demon-

* Corresponding Author. Email: nboehmer@g.harvard.edu.
[1] Equal contribution.

strate the quality of our heuristic through extensive experiments.

We further this investigation in Section 6 by studying a different variant where sensors are decentralized hence each sensor acts independently according to a simple greedy strategy. We show that it remains NP-hard for the blue player to compute its optimal strategy. While in this setting blue's problem admits an ILP formulation, we demonstrate in experiments that it can only solve up to medium-sized instances. Addressing this, we present heuristics that perform well in our experiments. We also demonstrate that while sensors usually have some gain from coordination, this gain depends decisively on the instance structure and is oftentimes rather small (below 20%).

Full proofs of all results and descriptions of additional experiments can be found in our full version [6].

## 2 Escape Sensing Games (ESGs): The Model


**Figure 1.** A visual representation of an Escape Sensing Game.

In an *Escape Sensing Game* (ESG) a blue player (henceforth `Blue`) tries to route a set of targets through a channel. They compete against a red player (henceforth `Red`) who controls a set of sensors and tries to sense (and therefore "steal") as many of `Blue`'s targets as possible.[2] Formally, an *Escape Sensing Game* is defined by

1. a set of targets $T = \{t_1, \dots, t_n\}$, each target $t_i \in T$ equipped with some utility value $v_i \in \mathbb{R}^+$,
2. a set of sensors $S = \{s_1, \dots, s_k\}$ and a recharging time $\tau \in \mathbb{N} \cup \{\infty\}$ which is the same for all sensors and,
3. a sensing matrix $D \in \{0,1\}^{n \times k}$ where $D_{i,j} = 1$ means that sensor $s_j$ is capable of sensing target $t_i$.

We assume that all of these parts are known to both players at any point in time. Note that in this paper, we consider the constant-sum utility structure and leave the general-sum version for future work. That is, `Blue` seeks to maximize the summed value of *not-sensed* targets, i.e., targets not sensed by any sensor. In contrast, `Red` seeks to minimize this value, or equivalently, maximize the summed value of targets that are sensed by some sensor.

The strategy of `Blue` is an ordering of the targets $\sigma : T \to [n]$ that assigns each target $t \in T$ a unique position $\sigma(t)$, i.e., $\sigma$ is a bijection. The targets move through the channel according to $\sigma$, i.e., the target on position 1 moves first, on position 2 second, and so on. In each *time step* each target moves to the next sensor, leaves the channel (in case it passed all sensors), or enters the channel at the first sensor (in case it is the next target in the ordering $\sigma$).

The strategy of `Red` is a sensing plan $\psi : S \to 2^T$ that maps each sensor to a subset of targets $T' \subseteq T$ sensed by the sensor, where each sensor senses different targets, i.e., $\psi(s) \cap \psi(s') = \emptyset$ for each $s \neq s' \in S$. `Red` cannot play arbitrary sensing plans but only those which are *valid*. A sensing plan is *valid* (with respect to a senor ordering $\sigma$) if (i) a sensor only senses targets it has the capabilities to sense, i.e., for each $s_j \in S$ and $t_i \in \psi(s_j)$ we have $D_{i,j} = 1$, and (ii) a sensor pauses for at least $\tau$ time steps after sensing a target, i.e., for each $s \in S$ and $t, t' \in \psi(s)$ we have $|\sigma(t) - \sigma(t')| > \tau$. Given a sensing plan, we can immediately calculate the value of not-sensed targets as $v(\psi) := \sum_{t_i \in T \setminus \cup_{s \in S} \psi(s)} v_i$, which quantifies `Blue`'s utility.

**Objectives and equilibrium** Due to the motivating applications of our interest, this work adopts `Blue`'s perspective and analyzes sequential play in this game by assuming `Blue` moves first.[3] Our analysis consists of two parts. First, we will analyze the best response problem for `Red` called BEST RED RESPONSE: Given an ordering $\sigma$ of the targets, output the sensing plan $\psi$ that is valid with respect to $\sigma$ and minimizes $v(\psi)$ among such plans. Second, to compute the optimal strategy of `Blue`, we analyze the game's Stackelberg equilibrium[4], which can be written as the following bilevel optimization problem: $\max_\sigma \min_{\psi : \psi \text{ is valid wrt. } \sigma} v(\psi)$. We term the corresponding computational problem BLUE LEADER STACKELBERG EQUILIBRIUM.

**A motivating application** One major motivation of our work is the secure transit of peacekeeping resources in the presence of adversarial actors such as pirates, which has critical importance due to past incidents, e.g., to the United Nations [24]. Citing the UN's peacekeeping mission manual [25], "*protecting shipping in transit ensures the safety and security of vessels as they pass through waters threatened by piracy on the high seas...*" In these applications, UN plays `Blue`'s role whereas pirates correspond to `Red`, who can observe the ordering of targets and then act second. The UN commands a fleet of ships (i.e., targets in our model) that often carry resources of different importance and that can be arranged strategically. Protecting shipping is overall a complex, multi-facet, task and our model captures one of the phases after potential (often scarce) security measures have already been allocated to the ships and the pirates look to identify targets to attack. According to Winn and Govern [27], pirates often use a set of boats (i.e., sensors in our model) to probe different passing targets, usually by following them to observe their speed, crew amount, firearm, etc. to judge based on this whether they are capable of capturing the ship. Such probing takes time, which is modeled by the recharging time $\tau$.

**Sensor and target types** We develop some customized algorithms for instances with only a few different target or sensor models: We say that two sensors $s_i, s_j \in S$ are of the same type if they are capable of sensing the same targets, i.e., the $i$-th and $j$-th column of the sensing matrix are identical. We say that two targets $t_i, t_j \in T$ are of the same type if they have the same utility value and can be sensed by the same sensors, i.e., $v_i = v_j$ and the $i$-th and $j$-th row of the sensing matrix are identical. We denote as $\Gamma = \{\gamma_1, \dots, \gamma_{n_\chi}\}$ and $\Theta = \{\theta_1, \dots, \theta_{k_\chi}\}$ the set of target and sensor types, respectively. It is easy to see that $k_\chi \leq 2^{n_\chi}$, as a sensor's sensing capabilities are defined by the set of target types it can sense. Similarly, assuming that all targets have the same value, it holds that $n_\chi \leq 2^{k_\chi}$.

## 3 Related Work

While the escape sensing game model is new, it is closely related to a few lines of AI research, as detailed below.

**Computational game theory for security** Conceptually, our work subscribes to the extensive MAS literature on computational game theory for tackling security challenges. The Stackelberg security

[2] Note that the terms "sensor" and "sensing" are only part of our terminology and do not limit the applications of our model. For instance, instead of "sensing" the targets, `Red` might also aim to intercept them.

[3] Note that this is already reflected in our game definition, since the validity of `Red`'s sensing plan depends on the strategy of `Blue`. Thus, `Red` cannot move before or simultaneous to `Blue`.

[4] We assume that ties in the strategies are broken according to some predefined lexicographic ordering of the strategies.

game [23] is one widely studied example. Other game-theoretic models include the hide-and-seek game [8], blotto games [4], auditting games [5] and catcher-evader games [18]. Most of these games study the optimal usage of security forces under different game structures. In contrast, our ESG model is motivated by detection-vs-evasion situations in which security forces have already been allocated.

**Scheduling** On a formal level, our problem is to schedule/order targets in an adversarial environment, which shares similarities with the classic problem of *scheduling* that looks to assign tasks to different machines to optimize certain criteria [17]. There is a rich body of AI research on scheduling, ranging from solving varied problems using AI techniques such as satisfiability [11] and distributed constraint optimization [21], to developing new models of scheduling problems under uncertainty [3] or in multi-agent setups [28].

In fact, the BEST RED RESPONSE problem can be formulated as the following slightly non-standard scheduling problem: There are $k$ machines (modeling sensors). In each step, a job (modeling a target) arrives. The job can be processed (modeling sensing) by a given subset of machines and if executed successfully generates a given reward value. The job has a processing time of $\tau$ and needs to be processed within the next $\tau$ steps. This implies that the job needs to be processed (i.e., sensed) either now or its reward is lost.

## 4 The Algorithmics of Escape Sensing Games

We analyze the computational complexity of ESGs starting with `Red`'s best response problem, followed by computing equilibria.

### 4.1 Computing `Red`'s Best Response Strategy

We analyze `Red`'s best response problem that `Red` needs to solve in each game after `Blue` has committed to a target ordering. This problem turns out to be NP-hard, even if `Red` is only interested in determining whether it can sense all targets. This intractability result is the first strong indicator of the intricate game dynamics in ESGs.

**Theorem 1.** BEST RED RESPONSE *is NP-complete, even when asked to decide whether* `Red` *can sense all targets or not.*

*Proof.* We reduce from HITTING SET where we are given a universe $U$, a collection of sets $\mathcal{Z} = \{Z_1, \dots, Z_m\}$ and an integer $t$, and the question is whether there a size-$t$ subset $U' \subseteq U$ containing at least one element from each set in $\mathcal{Z}$ (we assume that $t \geq 2$ and $|U| > t$).

In the construction, all targets have a value of 1 and the question is whether `Red` can sense all targets. As the core of the construction we add *element sensors* $\{a_u \mid u \in U\}$, *set targets* $\{\alpha_Z \mid Z \in \mathcal{Z}\}$, and *selection targets* $\{\beta_{i,j} \mid i \in [|U|-t], j \in [m]\}$. Each element sensor can sense all selection targets and all set targets corresponding to sets in which the element appears. Regarding the ordering of targets, it is easiest to think of the targets as being arranged in "rounds". In each round $j \in [m]$, first the selection targets $\{\beta_{i,j} \mid i \in [|U|-t]\}$ move through the channel followed by the the set target $\alpha_{Z_j}$. The idea is that the same $|U| - t$ element sensors sense the selection targets in every round, which correspond to the elements that are not part of the hitting set (we extend the construction in the following paragraph to ensure that this holds). Then, the remaining $t$ element sensors need to form a hitting set to be able to sense the set target in each round.

We extend the construction as follows. We add *filling targets* $\gamma_{i,j}$ for all $i \in [t-1]$ and $j \in [m]$, which all element sensors can sense. Moreover, we add *dummy sensors* $d_{i,j}$ for each $i \in [2|U|]$ and $j \in [m]$ and *dummy targets* $\delta_{i,j}$ for each $i \in [2|U|]$ and $j \in [m]$. For each $i \in [2|U|]$ and $j \in [m]$, dummy sensor $d_{i,j}$ can sense dummy target $\delta_{i,j}$. We set $\tau := 2|U| + 1$. Formally, the target ordering $\sigma$ is constructed—in multiple "rounds"—as follows. In each round $j \in [m]$, we first move the selection targets $\{\beta_{i,j} \mid i \in [|U|-t]\}$ through the channel, then the dummy targets $\{\delta_{i,j} \mid i \in [|U|]\}$, then the set target $\alpha_{Z_j}$, then the filling targets $\{\gamma_{i,j} \mid i \in [t-1]\}$ and then the dummy targets $\{\delta_{i,j} \mid i \in [|U|+1, 2|U|]\}$ (the ordering of targets in each of the groups is arbitrary).

**Proof of correctness: forward direction** Assume that $U' \subseteq U$ is a size-$t$ hitting set of $\mathcal{Z}$. For each $i \in [2|U|]$ and $j \in [m]$, we let $d_{i,j}$ sense $\delta_{i,j}$. We construct the sensing plan for the element sensors iteratively as follows. In each round $j \in [m]$, we let each of the $|U| - t$ element sensors $\{a_u \mid u \in U \setminus U'\}$ sense exactly one of the selection targets $\{\beta_{i,j} \mid i \in [|U| - t]\}$. Now, let $u^*$ be an element from $U'$ that is contained in $Z_j$ (such an element needs to exist because $U'$ is a hitting set). We let $a_{u^*}$ sense $\alpha_{Z_j}$ and we let each of the $t-1$ element sensors $\{a_u \mid u \in U' \setminus \{u^*\}\}$ sense exactly one of the filling targets $\{\gamma_{i,j} \mid i \in [t-1]\}$.

The constructed sensing plan senses all targets and clearly respects the sensing matrix. It remains to be argued that the recharging times of all element sensors are respected (dummy sensors only sense one target). For each $u \in U \setminus U'$, we have that $a_u$ senses one selection target in each round. Between two selection targets in two different rounds there are at least $2|U|$ dummy targets and one set target, so recharging times are respected. For each $u \in U'$, the sensor $a_u$ senses either a set or filling target in each round. There are $2|U|$ dummy targets and $|U| - t \geq 1$ selection targets between each two sets and filling targets from different rounds, so recharging times are respected.

**Proof of correctness: backward direction** Assume that $\psi$ is a valid sensing plan that senses all targets. Consequently, in each round, the $|U|$ element sensors need to sense $|U| - t$ selection, $t-1$ filling, and one set target. As $\psi$ is valid and there are only $|U|-t-1+|U|+1+t-2 = 2|U|-2$ targets between the first selection and last filling target in each round, this means that each element sensor needs to sense exactly one of these targets in each round. Note that an element sensor that senses a non-selection (i.e., either a set or filling) target in round $j \in [m]$ cannot sense a selection target in round $j+1$, as there are only $t-1+|U|+|U|-t-1 = 2|U|-2$ targets between the first non-selection target in round $j$ and the last selection target in round $j+1$. Consequently, as each element sensor needs to sense one target in each round, it follows that there is a set $U'' \subseteq U$ of $|U| - t$ elements so that the corresponding element sensors sense a selection target in every round. Consequently, the remaining $t$ element sensors need to sense all set targets. As an element sensor is only capable of sensing a set target if the element appears in the set, it follows that $U \setminus U''$ is a size-$t$ hitting set of $\mathcal{Z}$. □

Despite this intractability result, it is still possible to construct exact combinatorial algorithms for BEST RED RESPONSE. In particular, we present a dynamic programming-based algorithm empowered by some structural observations on ESGs that runs in $\mathcal{O}(n \cdot (k_\chi + 1)^{\tau+2})$ (recall that $k_\chi \leq k$). This algorithm in particular implies that the problem becomes polynomial-time solvable if the recharging time, which we expect to be rather small in comparison to the number of targets, is a constant.

**Proposition 2.** *There is a* $\mathcal{O}(n \cdot (k_\chi + 1)^{\tau+2})$*-time algorithm for* BEST RED RESPONSE.

*Proof Sketch.* Our idea is to construct a valid sensing plan iteratively by going through the arriving targets one by one (we assume that the

ordering of targets is $t_1, \dots, t_n$). For each target, we either decide that it will not be sensed or assign it to one of the sensors so that the resulting plan is still valid. Our key observation to bring down the time and space complexity of the dynamic program is that we do not need to store the full sensing plan to ensure the validity of the plan after updating it. Instead, it is sufficient to know for each sensor whether it has sensed a target in the last $\tau$ steps. More formally, given a valid sensing plan $\psi$ that has been constructed by iterating over the first $i \in [n]$ targets, we only store the following information: (i) the value of all targets $\{t_1, \dots, t_i\}$ that have not been assigned to a sensor in $\psi$, i.e., $\sum_{t_j \in \{t_1,\dots,t_i\} \setminus \cup_{s \in S} \psi(s)} v_j$, (ii) the sensors the last $\tau + 1$ targets have been assigned to. It is possible to store this information in a table of size $\mathcal{O}(n \cdot (k+1)^{\tau+1})$ where each cell can be computed in $\mathcal{O}(k)$-time. To extend the algorithm to sensor types, we prove that we can collapse sensors of one type into a "meta" sensor, making it sufficient to bookmark the types of sensors that have sensed the last $\tau + 1$ targets. □

We conclude by giving a clean ILP formulation of BEST RED RESPONSE, which turns out to scale very favorably in our experiments allowing us to solve instances with up to 10000 targets within one minute.

**Proposition 3.** BEST RED RESPONSE *admits an ILP formulation with $\mathcal{O}(n \cdot k)$ binary variables and $\mathcal{O}(n \cdot k)$ constraints.*

*Proof.* We model an instance $\mathcal{I}$ of BEST RED RESPONSE as an ILP as follows. We assume that the targets are ordered as $t_1, \dots, t_n$. We create a binary variable $x_{i,j}$ for each $i \in [n]$ and $j \in [k+1]$. Setting $x_{i,j}$ to one corresponds to letting sensor $s_j$ sense target $t_i$ if $j \in [k]$, and letting $t_i$ not be sensed by any sensor if $j = k+1$.

To ensure that `Red` minimizes the value of not-sensed targets, the optimization criterion becomes: $\min \sum_{i \in [n]} v_i \cdot x_{i,k+1}$. To ensure the validity of the sensing plan $\psi$, for each $i \in [n]$, we enforce that: $\sum_{j \in [k+1]} x_{i,j} = 1$. Moreover, to ensure that sensor capabilities are respected, we impose for each $i \in [n]$ and $j \in [k]$ that: $x_{i,j} \leq D_{i,j}$. Lastly, to enforce that recharging times are respected, for each $j \in [k]$ and $i \in [n - \tau]$ we add the constraint: $\sum_{\ell=i}^{i+\tau} x_{\ell,j} \leq 1$. □

## 4.2 Solving for the Stackelberg Equilibrium

We now study the problem of computing `Blue`'s optimal strategy, i.e., to solve BLUE LEADER STACKELBERG EQUILIBRIUM. Theorem 1 already shows the NP-hardness of BEST RED RESPONSE. While this does not imply the hardness of computing Stackelberg equilibria[5], a convincing intractability result for `Blue`'s optimal strategy shall ideally "disentangle" its complexity from `Red`'s best response problem. With this in mind, we prove the NP-hardness of BLUE LEADER STACKELBERG EQUILIBRIUM even in situations where `Red`'s best response problem is linear-time solvable. This demonstrates that the complexity in our reduction does not come from finding `Red`'s strategy but from the problem of whether `Blue` can arrange the targets in an optimal way.

**Theorem 4.** BLUE LEADER STACKELBERG EQUILIBRIUM *is NP-hard, even on instances where* BEST RED RESPONSE *is linear-time computable and the recharging time is* 3.

[5] Note that the fact that it is NP-hard for `Red` to best respond to certain `Blue` strategies (as constructed in the reduction of Theorem 1) does not imply that is also hard for `Red` to best respond to the particular Stackelberg equilibrium strategy of `Blue` (as these strategies might admit some structure that makes it easier to best respond).

Note that the NP-hardness upholds even if sensors' recharging time is constant, a case in which `Red`'s best response problem is polynomial-time solvable (see Proposition 2). Our hardness result indicates that computing `Blue`'s optimal strategy is a generally much harder problem than computing `Red`'s optimal strategy. In fact, it remains open whether BLUE LEADER STACKELBERG EQUILIBRIUM is contained in NP or whether it is complete for complexity classes beyond NP. We suspect the latter to hold.

### 4.2.1 Bilevel Optimization

In light of this, it is unclear (and from our perspective rather unlikely) that BLUE LEADER STACKELBERG EQUILIBRIUM admits an ILP formulation. Naive brute-force approaches are also computationally infeasible, as we would need to enumerate all $n!$ possible target orderings and solve the NP-hard BEST RED RESPONSE problem as a subroutine for each of them.

Thus, we turn to a formulation as a bilevel optimization problem [9] as one way to solve the problem exactly. In such formulations, constraints are still linear, but there exist two connected levels of the problem, i.e., an outer and an inner level. The inner level controls certain variables that it sets to minimize an objective subject to linear constraints that also involve variables controlled by the outer level, while the outer level sets these variables to maximize the objective. In our problem, we can model `Red`'s best response problem as the inner level loosely following the ILP from Proposition 3. The outer-level models `Blue`'s problem. The key parts of the outer level are variables for each target that encode the position in which the target appears in the final ordering and that are used in the inner level to ensure the validity of the sensing plan.

**Proposition 5.** BLUE LEADER STACKELBERG EQUILIBRIUM *admits a bilevel optimization formulation with $\mathcal{O}(n^2 + n \cdot k)$ binary variables, $\mathcal{O}(n)$ integer variables, and $\mathcal{O}(n^2 \cdot k)$ constraints.*

Note that standard techniques to convert this bilevel program into an (integer) linear program, e.g., by exploiting KTT-optimality conditions [2, 13], are not applicable in our setting, as we are solving an *integer* bilevel program within which the inner-level program is already non-convex.

### 4.2.2 Heuristic

We will see later that the running time for the bilevel formulation of the problem becomes already infeasible on small-sized instances. Therefore, we experimented with different heuristics to solve the problem.[6] In the following, we present two variants of simulated annealing-based heuristics that performed best. For a target ordering $\sigma$, we denote as $N(\sigma)$ its neighbors, i.e., all $\binom{n}{2}$ orderings that arise from $\sigma$ by swapping the position of any two different targets. The *relaxed version* of our simulated annealing (SA_Relax) is presented in Algorithm 1. The idea is to find an optimal ordering through repeated local rearrangements. We store the current ordering as $\sigma$ and compute its value for `Blue` by solving `Red`'s best response problem using Proposition 3. Then, we pick a random neighbor of $\sigma$, compute

[6] Note that the heuristic double-oracle approach that has been successfully employed for other large combinatorial games [1, 16] is not applicable to ESGs. Traditionally, the approach successively expands the strategy spaces of both players by letting them best respond to each other. However, in ESGs, we face a bilevel problem in which there is no best response of the leader to the follower. The approach also fails here because the valid strategies of the follower heavily depend on the strategy picked by the leader.

**Algorithm 1** SA_RELAX

**Input**: Target ordering $\sigma$ and temperature $T = 100$

1: Compute optimal sensing plan $\psi$ wrt. $\sigma$.
2: **while** $T > 0.00001$ **do**
3: Select a random neighbor $\hat{\sigma} \in N(\sigma)$.
4: Compute optimal sensing plan $\psi'$ wrt. $\hat{\sigma}$.
5: **if** $e^{\frac{v(\hat{\psi})-v(\psi)}{T}} >$ random[0, 1] **then**
6: $\sigma := \hat{\sigma}$, $\psi := \hat{\psi}$.
7: $T := T \cdot 0.9$.
8: **Return** $\sigma$.

its value, and update the ordering based on this according to standard simulated annealing rules.

In the *full version* of our simulated annealing (SA), instead of picking a random neighbor $\hat{\sigma}$ from $N(\sigma)$ in Line 3 of Algorithm 1, we first run a heuristic for BEST RED RESPONSE on all orderings from $N(\sigma)$.[7] Then, on the $\mu$ fraction of neighbors with the highest returned value, we execute the ILP from Proposition 3 to compute the optimal sensing plan. Of the examined neighbors, we pick the one with the highest returned value as $\sigma'$. As a hyper-parameter tuning process, we tested the performance of our heuristic algorithm with respect to the choice of $\mu$ (see our full version [6] for results). It turns out that $\mu = 0.1$ provides a good trade-off between the algorithm's running time and `Blue`'s utility. Thus, we fix $\mu = 0.1$ throughout the paper. For both heuristics, we always run the heuristic three times with three different initial randomly generated target orderings and return the best computed ordering.

## 5 Experimental Evaluations

We analyze the quality and performance of our algorithms to compute the Stackelberg equilibrium.[8] We consider three simulated game settings for generating ESGs. For each setting, we determine the value of a target by drawing a number uniformly within $[0, 1]$[9]:

1. **DEFAULT (DEF):** For each $i \in [n]$ and $j \in [k]$, we set $D_{i,j} = 1$ with probability 0.2.
2. **EUCLIDEAN (EUC):** Each target $t_i \in T$ and each sensor $s_j \in S$ are uniformly sampled points in $[0, 1] \times [0, 1]$. A sensor can sense a target (i.e., $D_{i,j} = 1$) if the Euclidean distance between their points is below 0.3.
3. **RANDOMLEVEL (RAND):** Each target $t_i \in T$ has a difficulty level $d_i$ uniformly sampled from $[0, 1]$, and each sensor $s_j \in S$ has a skill level $s_j$ uniformly sampled from $[0, 1]$. For each $i \in [n]$ and $j \in [k]$, we set $D_{i,j} = 1$ with probability $(1 - d_i) \cdot s_j$.

In all our experiments, if not stated otherwise, we average over 50 instances generated according to one of the models. We present our experimental results as tables where each entry contains `Blue`'s average utility (i.e., the summed value of not-sensed targets) from the computed target ordering assuming `Red` best responds and the average running time in seconds in *italics*, both followed by their respective standard deviations. Note that standard deviations are calculated across the different sampled instances, implying that independent of the solution method some non-trivial standard deviation is to be expected, as certain instances are more favorable for `Blue` than others.

We analyze the maximum size of instances that we can solve exactly using the bilevel program, which we denote as OPT. We present results for the **DEFAULT** game setting in Table 1 (results for other simulated game settings are similar). It turns out that while instances with 5 targets can be solved within a second by OPT, instances with 9 targets take already around 9 hours to solve. This demonstrates that the bilevel program is only usable for quite small instances. Moreover, we observe the to-be-expected trend that `Blue`'s utility increases when `Blue` has more targets or `Red` has less sensors. However, we do not find any consistent trend regarding whether it is more advantageous for `Blue`: more targets or fewer sensors.

Motivated by the high computational cost of the bilevel program, we now turn to analyzing the quality of our heuristics. We also include the *Random* method here as a baseline where `Blue` simply picks an arbitrary ordering of targets (and `Red` best responds to it). In addition, we compare our heuristics against a naive random strategy of comparable computational cost. For this, we include the *Random2* method which generates 3000 random orderings for Table 2 and $3 \cdot 10^7$ random orderings for Table 3. The sampled ordering that achieves the highest utility for `Blue` assuming that `Red` best responds is returned.

In Table 2, we show the algorithms' performance for small instances where we can still compute `Blue`'s maximum utility (OPT) via the bilevel program. In Table 3, we consider larger instances where the optimum value is unknown. Note that higher values correspond to a better performance of the algorithm, as we always report `Blue`'s utility for `Red`'s best response.

From the results in Table 2, we can see that all heuristics perform well on small instances. In particular, SA_Relax, SA, and Random2 find the optimal solution in all (but one) cases. However, SA_Relax proves advantageous because it only needs a sixth of the running time of the other two methods.

While our two heuristics SA and SA_Relax show a similar approximation quality for small instances, for larger instances (Table 3) SA clearly outperforms SA_Relax. For the **DEFAULT** game setting, using SA compared to SA_Relax even regularly leads to a doubled utility for `Blue`. While this is a strong argument for using SA, SA's downside is its higher computational cost, needing over 7 hours to solve instances with 75 targets.

Finally, we observe that both methods clearly outperform the Random baseline, with SA consistently preserving an average of approximately 20 more targets for the larger instances. This highlights that the solution quality of the target ordering clearly increases throughout the simulated annealing. Considering Random2, we find that repeatedly sampling orders (instead of only once) leads to a noticeable utility increase. However, on the larger instances, Random2 performs even worse than SA_Relax while running as long as SA, thereby combining the disadvantages of SA and SA_Relax. Overall, our experiments highlight that `Blue` benefits from ordering the targets strategically instead of randomly.

## 6 Escape from Non-Coordinated Sensing

ESGs assume that the different sensors are controlled by a central authority that computes the sensing plan. We now investigate the situation where these sensors are *non-coordinated* and each one acts independently based on a natural greedy algorithm. This happens when sensors cannot easily exchange information and coordinate with each

[7] In our (greedy) heuristic, we consider the targets in decreasing order of their value and construct the sensing plan $\psi$ iteratively. Let $S' \subseteq S$ be the sensors $s$ so that $\psi$ remains a valid sensing plan after adding the current target to $\psi(s)$. We let the target be sensed by a randomly selected sensor from $S$ (or by no sensor if $S$ is empty). For a formal description, see our full version [6].

[8] We use Gurobi [15] to solve the ILP from Proposition 3 and MIBS [22] to solve the bilevel program from Proposition 5. Both are among the most popular off-the-shelf tools for solving the respective problem.

[9] In our full version [6], we analyze supplementary scenarios, reinforcing similar conclusions to those presented here.

| #targets \ #sensors | 2 | 3 | 5 |
|---|---|---|---|
| 5 | 1.79 ± 0.71, *0.61 ± 0.21* | 1.55 ± 0.65, *0.77 ± 0.02* | 1.02± 0.62, *0.98 ± 0.04* |
| 7 | 2.41 ± 0.78, *102 ± 36* | 2.29 ± 0.80, *116 ± 31* | 1.62± 0.72, *140 ± 29* |
| 8 | 2.96 ± 0.81, *1501 ± 354* | 2.33 ± 0.74, *1760 ± 38* | 1.7 ± 0.69, *1814 ± 23* |
| 9 | n/a, *31358* | n/a, *32541* | n/a, 35376 |

**Table 1.** Scalability test of bilevel-program (OPT) for **DEFAULT** game setting with $\tau = 2$. For $n = 9$, we report running time for one instance. For all tables: each entry shows `Blue`'s average utility (top) and running time in seconds (bottom).

| Algo. \ Setting | DEF | EUC | RAND |
|---|---|---|---|
| OPT | **2.29** ± 0.80, *116 ± 31* | **1.952** ± 0.74, *126± 2.29* | **2.09** ± 0.87, *120 ± 25* |
| SA | **2.29** ± 0.80, *4.78 ± 0.38* | 1.951 ± 0.75, *4.96 ± 0.47* | **2.09** ± 0.87, *5.03 ± 0.69* |
| SA_Relax | **2.29** ± 0.80, *0.81 ± 0.03* | **1.952** ± 0.74, *0.84 ± 0.05* | **2.09** ± 0.87, *0.85 ± 0.08* |
| Random | 2.16 ± 0.84, *0.001* | 1.71 ± 0.83, *0.001* | 1.93 ± 0.92, *0.001* |
| Random2 | **2.29** ± 0.80, *5.13 ± 0.16* | **1.952** ± 0.74, *5.25 ± 0.33* | **2.09** ± 0.87, *5.3 ± 0.46* |

**Table 2.** Comparison of algorithms to compute `Blue`'s utility for different simulated game settings, where $n = 7$, $k = 3$, and $\tau = 2$.

| Algo. \ Setting | DEF | EUC | RAND |
|---|---|---|---|
| SA | **15.96** ± 1.1, *28101 ± 563* | **18.4** ± 2.1, *27755 ± 928* | **17.3** ± 4.2, *27970 ± 1136* |
| SA_Relax | 8.76 ± 0.9, *49.6 ± 2.57* | 10.53 ± 2.32, *47.5 ± 0.86* | 12.3 ± 3.6, *49.7 ± 1.6* |
| Random | 6.19 ± 1.26, *0.001* | 6.86 ± 2.25, *0.001* | 9.68 ± 2.78, *0.001* |
| Random2 | 8.27 ± 0.73, *25036 ± 311* | 9.54 ± 2.45, *24333 ± 211* | 11.68 ± 3.58, *26810 ± 295* |

**Table 3.** Comparison of algorithms to compute `Blue`'s utility for different simulated game settings, where $n = 75$, $k = 10$, and $\tau = 5$. We generate 9 instances per method.

other. Another motivation is when sensors are controlled by different adversaries, each serving only their own interests and being unlikely to coordinate their actions and share their reward. Both of these scenarios can occur in our motivating domain of piracy at large open seas, as coordination between different groups is likely to be challenging. Different pirate groups might even refuse to coordinate at all and instead directly compete with each other.

We model these situations by assuming that sensors have a predefined ordering as $s_1, \dots, s_k$ (as induced by fixed locations of the sensors); for each sensor $s \in S$, as soon as a not previously sensed target that $s$ can sense passes $s$ (i.e., $s$ has the capabilities and is currently not recharging), $s$ senses it, thereby greedily maximizing its number of sensed targets.

Formally, given a target ordering $\sigma$, we construct a sensing plan $\psi_\sigma$ sequentially as follows. For each step $\ell \in [n + k - 1]$, if target $t_i$ passes sensor $s_j$ in step $\ell$, then we add $t_i$ to $\psi_\sigma(s_j)$ if the resulting sensing plan remains valid with respect to $\sigma$ (formally, for $i \in [n]$ and $j \in [k]$ target $\sigma^{-1}(i)$ passes sensor $s_j$ in step $i + j - 1$). As the strategy of `Red` is fixed, the problem BEST BLUE RESPONSE `Blue` faces is to pick a target ordering $\sigma$ so that $v(\psi_\sigma)$ gets maximized. In the following, we study the computational complexity of this problem and solve it in computational experiments. By comparing the answer of BEST BLUE RESPONSE to the value of the Stackelberg equilibrium in the corresponding ESG we can ultimately answer how much `Red` gains from being able to centrally control its sensors.

## 6.1 Algorithmic Analysis

Unfortunately, it turns out that computing `Blue`'s strategy is NP-hard, even in restricted cases where each sensor can only sense one target. Due to the sequential construction of `Red`'s sensing plan, this reduction is our most intricate one:

**Theorem 6.** BEST BLUE RESPONSE *is NP-complete, even if the recharging time is $\infty$, i.e., each sensor can sense only one target, each target has value* 1*, and the sum of each row and column in the sensing matrix is at most four.*

*Proof Sketch.* We focus on the variant where each sensor can only sense one target. Interestingly, as discussed in more detail in our full version [6] this problem shares some similarities with the NP-hard MINIMUM MAXIMAL MATCHING problem, as we can view the sensors and targets as two sides of a bipartite graph with sensor-target pairs where the sensor senses the target corresponding to maximal matchings in this graph. However, the ordering of the sensors makes only certain maximal matchings in these graphs realizable, which is why we instead show NP-hardness by reducing from a variant of 3-SAT where each variable appears only twice positive and once negative. The core idea of our construction is the following: We add a *literal target* for each literal. Moreover, for each clause, we add a *clause sensor* and a *clause target*. The clause sensor is capable of sensing the corresponding clause target as well as targets corresponding to the three literals appearing in the clause. We add further targets and sensors to the instance so that all clause targets need to make it unsensed through the channel. This implies that each clause sensor needs to sense a literal target as it will otherwise sense the corresponding clause target in passing, i.e., we need to "cover" each clause with a literal appearing in the clause. Now for each variable, we add a slightly intricate gadget that ensures that we can either use the targets corresponding to positive literals to cover clause sensors (which corresponds to setting the variable to true) or the one target corresponding to a negative literal (which corresponds to setting the variable to false). Because we need to "cover" each clause, the induced assignment is satisfying. □

We can adopt a similar view as in Proposition 2 to solve the problem via dynamic programming. However, this time the dynamic programming iteratively constructs the optimal target ordering and we need to keep track of the previously used targets together with the sensors used in the last $\tau + 1$ timesteps. This results in a naive running time of $\mathcal{O}(n \cdot 2^n \cdot (k+1)^{\tau+2})$, which can be improved to $\mathcal{O}\left(n_\chi \cdot \left(\prod_{i=1}^{n_\chi}(\ell_i + 1)\right) \cdot (k+1)^{\tau+2}\right)$ if we incorporate types:

**Proposition 7.** BEST BLUE RESPONSE *is solvable in $\mathcal{O}\left(n_\chi \cdot \left(\prod_{i=1}^{n_\chi}(\ell_i + 1)\right) \cdot (k+1)^{\tau+2}\right)$, where $\ell_i$ is the number of targets of type $\gamma_i$.*

### 6.1.1 ILP Formulation

Constructing an ILP for BEST BLUE RESPONSE turns out to be slightly more challenging, as we need to encode `Red`'s greedy sequential behavior:

**Proposition 8.** *There is an ILP formulation for* BEST BLUE RESPONSE *with $\mathcal{O}(n^2 \cdot k)$ binary variables, $\mathcal{O}(n)$ integer variables, and $\mathcal{O}(n^2 \cdot k)$ constraints.*

*Proof Sketch.* We introduce for each target $i \in [n]$ an integer variable $z_i$ encoding the position in which the target appears. Moreover, similar to Proposition 3, for each $i \in [n]$ and $j \in [k+1]$, we add a binary variable $x_{i,j}$, which encodes whether $t_i$ is sensed by sensor $s_j$ or whether the target makes it unsensed through the channel (for $j = k+1$). We can add mostly straightforward constraints to ensure that $x_{i,j}$ respects recharging times. The main challenge is to encode the greedy behavior of the sensors (i.e., the ILP cannot have the freedom to pick the $x_{i,j}$ values arbitrarily to optimize `Blue`'s utility but they are set according to sensors' greedy behavior). For this, for each

| #targets \ #sensors | 2 | 3 | 5 |
|---|---|---|---|
| 5 | 2 ± 0.68, *0.007 ± 0.005* | 1.71 ± 0.57, *0.009 ± 0.005* | 1.32± 0.52, *0.01 ± 0.003* |
| 15 | 6.6 ± 1, *0.14 ± 0.35* | 6.29 ± 0.9, *0.32 ± 0.88* | 5.46 ± 0.87, *6.85 ± 22.9* |
| 20 | 9.05 ± 1.13, *0.52 ± 2.94* | 8.49 ± 1.01, *1.58 ± 4.4* | 7.56 ± 1.01, *229 ± 1261* |
| 25 | 11.38 ± 1.26, *6.8 ± 30.6* | 10.96 ± 1.29, *283 ± 1771* | n/a, *22537* |

**Table 4.** Scalability test of ILP (OPT) for **DEFAULT** game setting with $\tau = 2$. For all tables: each entry shows `Blue`'s average utility (top) and running time in seconds (bottom).

| Algo. \ Setting | DEF | EUC | RAND |
|---|---|---|---|
| OPT | **3.1**± 0.86, *0.15 ± 0.37* | **3.25**± 0.79, *0.25 ± 0.71* | **3.04** ± 0.78, *3.26 ± 8.77* |
| SA | 2.83 ± 0.81, *0.7± 0.02* | 2.92 ± 0.81, *0.71 ± 0.02* | 2.72 ± 0.8, *0.71 ± 0.018* |
| SA_Relax | 3.06 ± 0.88, *0.02* | 3.17 ± 0.81, *0.02* | 2.89 ± 0.82, *0.02* |
| Random | 1.9 ± 0.72, *0.001* | 2.15 ± 0.9, *0.001* | 1.9 ± 0.9, *0.001* |
| Random2 | 2.91 ± 0.87, *0.81 ± 0.0004* | 3.11 ± 0.79, *0.81 ± 0.0004* | 2.89 ± 0.78, *0.83 ± 0.0004* |

**Table 5.** Comparison of algorithms for BEST BLUE RESPONSE for different game settings, where $n = 10$, $k = 5$, and $\tau = 2$.

| Algo. \ Setting | DEF | EUC | RAND |
|---|---|---|---|
| SA | **16.57** ± 1.64, *485 ± 20* | **17.2** ± 2.4, *470 ± 9.6* | **17.54** ± 3.27, *503 ± 20* |
| SA_Relax | 12.3 ± 1.58, *3.14 ± 0.35* | 13.11 ± 2.62, *2.84 ± 0.2* | 14.47 ± 2.96, *2.95 ± 0.2* |
| Random | 9.2 ± 1.99, *0.001* | 10.02 ± 2.77, *0.001* | 11.67 ± 3.07, *0.001* |
| Random2 | 12.75 ± 1.03, *458 ± 15* | 12.98 ± 2.37, *437 ± 13* | 14.67 ± 3.18, *496 ± 19* |

**Table 6.** Comparison of algorithms for BEST BLUE RESPONSE for different game settings, where $n = 75$, $k = 10$, and $\tau = 5$.

**Algorithm 2** SA_Relax for BEST BLUE RESPONSE

**Input**: Initial target ordering $\sigma$ and temperature $T = 100$

1: **while** $T > 0.00001$ **do**
2: Select a random neighbor $\hat{\sigma} \in N(\sigma)$.
3: **if** $e^{\frac{v(\psi_{\hat{\sigma}})-v(\psi_\sigma)}{T}} >$ random[0, 1] **then**
4: $\sigma := \hat{\sigma}$.
5: $T := T \cdot 0.9$.
6: **Return** $\sigma$.

$i, i' \in [n]$ and $j \in [k]$, we add a binary variable $y_{i,i',j}$ and add constraints so that $y_{i,i',j}$ is equal to one if target $i$ is sensed by sensor $j$ and because of this $j$ recharges when $i'$ is passing, i.e., $i$ "covers" $i'$.

To encode sensors' greedy behavior, we want to add a constraint that makes sure that in case $x_{i,j} = 1$, the target needs to be covered by other targets for all sensors that are capable of sensing it placed before $j$. Note that this together with another constraint ($\sum_{j\in[k+1]} x_{i,j} = 1$) in particular implies that each target is sensed by the first sensor it passes which is not recharging, thereby encoding the greedy behavior of sensors. Specifically, for each $i \in [n]$ and $j \in [k+1]$, we add:

$$\sum_{t\in[j-1]:D_{i,t}=0} 1 + \sum_{t\in[j-1]:D_{i,t}=1} \sum_{i'\in[n]} y_{i',i,t} - (j-1) \quad (1)$$
$$\geq -n(1 - \sum_{t=j}^{k+1} x_{i,t}).$$

□

### 6.1.2 Heuristic

Since it will turn out that the ILP formulation cannot quickly solve medium-to-large instances, we explore various simulated annealing-based heuristics, similar to the approach discussed in Section 5. We present the variant SA_Relax where a random neighbor is picked in Algorithm 2. The other variant SA computes `Blue`'s utility $v(\psi_{\hat{\sigma}})$ for all neighbors and picks the one with the highest utility.

## 6.2 Experiments

We reuse the general setup described in Section 5, but naturally now report `Blue`'s computed utility assuming that sensors act greedily. Here, we let the Random2 method generate 1000 random orderings in Table 5 and $5 \cdot 10^5$ random orderings in Table 6.

First of all, we evaluate the scalability of our ILP for BEST BLUE RESPONSE (OPT) in Table 4. The ILP can solve the problem for medium-sized instances with up to 25 targets in a few minutes. However, due to the complexity of the ILP modeling, already for 25 targets as soon as the number of sensors reaches 5, instances can take more than 5 hours to solve. This is why the last line of the table only reports the running time for one instance.

Next, we analyze the solution quality of our heuristic approaches. On small instances presented in Table 5, our best heuristic algorithm approximates the optimal solution quite well and the error is typically below 10% with the SA_Relax method consistently outperforming SA. Both heuristics outperform Random, while Random2 performs better than SA (yet still worse than SA_Relax, while having a much longer running time). When moving to larger instances in Table 6, the picture flips, as SA is now substantially outperforming SA_Relax. This shows a general trend that the solution quality of SA scales more favorably than that of SA_Relax (while the opposite is naturally true for the running time). The heuristics again clearly outperform Random, with SA sensing approximately 15 more targets. Random 2 performs similarly to the suboptimal heuristic SA_Relax, while being slower by a factor of more than 100.

Finally, we are interested in exploring the power of coordination for `Red`, i.e., the difference between the optimal utility `Blue` gets in the non-coordinated setting explored in this section compared to its utility in the Stackelberg equilibria from Section 5. We find that for the small instances where we can compute the Stackelberg equilibrium exactly `Red` can reduce `Blue`'s utility by 10% to 20% through coordination. For larger instance sizes, we no longer know the optimal solutions, which is why we resort to comparing the results of the respective SA heuristics. We find that for larger instances, the gap decreases with `Red` being only able to decrease `Blue`'s utility by 5% through coordination in the instances from the **DEFAULT** setting underlying Table 3. In our full version [6], we show that when `Red`'s sensors are capable of sensing more targets, coordination is more important sometimes leading to halving `Blue`'s utility.

# 7 Conclusion

By introducing Espace Sensing Games, we initiated the study of a new class of games concerned with target arrangement and motivated by security applications. We showed that while the worst-case computational complexity of ESGs is prohibitive, our presented algorithms still have a good performance in experiments.

There are multiple directions for future work emanating from our work. First, pinpointing the precise complexity of computing Stackelberg equilibria remains a concrete open question. Second, there are other variants of ESGs beyond those studied by us. For instance, it would be possible to merge the settings studied in Sections 4 and 6 into a game where sensors act greedily but `Red` can control the ordering of the sensors. In this game variant where both `Red` and `Blue` need to pick orders, it would also be possible to study simultaneous play or Stackelberg equilibria where `Red` moves first. Lastly, there are various other target arrangement problems to be studied. One example could be a game where `Blue` needs to place targets on a grid and `Red` cannot sense any two targets placed close to each other.

## Acknowledgements

This work was supported by the Office of Naval Research (ONR) under Grant Number N00014-23-1-2802. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Office of Naval Research or the U.S. Government.

## References

[1] L. Adam, R. Horcík, T. Kasl, and T. Kroupa. Double oracle algorithm for computing equilibria in continuous games. In *AAAI*, pages 5070–5077, 2021.

[2] G. B. Allende and G. Still. Solving bilevel programs with the kkt-approach. *Math. Program*, 138:309–332, 2013.

[3] E. Bampis, K. Dogeas, A. V. Kononov, G. Lucarelli, and F. Pascual. Scheduling with untrusted predictions. In *IJCAI*, pages 4581–4587, 2022.

[4] S. Behnezhad, A. Blum, M. Derakhshan, M. HajiAghayi, M. Mahdian, C. H. Papadimitriou, R. L. Rivest, S. Seddighin, and P. B. Stark. From battlefields to elections: Winning strategies of blotto and auditing games. In *SODA*, pages 2291–2310, 2018.

[5] J. Blocki, N. Christin, A. Datta, A. Procaccia, and A. Sinha. Audit games with multiple defender resources. In *AAAI*, pages 791–797, 2015.

[6] N. Boehmer, M. Han, H. Xu, and M. Tambe. Escape sensing games: Detection-vs-evasion in security applications. *CoRR*, abs/2407.20981, 2024.

[7] V. Bucarey, C. Casorrán, Ó. Figueroa, K. Rosas, H. Navarrete, and F. Ordóñez. Building real Stackelberg security games for border patrols. In *GameSec*, pages 193–212, 2017.

[8] M. Chapman, G. Tyson, P. McBurney, M. Luck, and S. Parsons. Playing hide-and-seek: an abstract game for cyber security. In *ACySE*, pages 1–8, 2014.

[9] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of operations research*, 153:235–256, 2007.

[10] J. H. Conway. *On numbers and games, Second Edition*. Academic Press, 2001.

[11] J. M. Crawford and A. B. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *AAAI*, pages 1092–1097, 1994.

[12] E. D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. In *MFCS*, pages 18–32, 2001.

[13] S. Dempe and A. Zemkoho. Bilevel optimization. In *Springer optimization and its applications*, volume 161. Springer, 2020.

[14] F. Fang, T. Nguyen, B. Ford, N. Sintov, and M. Tambe. Introduction to green security games. In *IJCAI*, 2015.

[15] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL https://www.gurobi.com.

[16] M. Jain, D. Korzhyk, O. Vanek, V. Conitzer, M. Pechoucek, and M. Tambe. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, pages 327–334, 2011.

[17] J. K. Lenstra, A. R. Kan, and P. Brucker. Complexity of machine scheduling problems. In *Annals of discrete mathematics*, volume 1, pages 343–362. Elsevier, 1977.

[18] Y. Li, V. Conitzer, and D. Korzhyk. Catcher-evader games. In *IJCAI*, pages 329–337, 2016.

[19] C. Löding and P. Rohde. Solving the sabotage game is PSPACE-hard. In *MFCS*, pages 531–540, 2003.

[20] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer. Protect: A deployed game theoretic system to protect the ports of the united states. In *AAMAS*, pages 13–20, 2012.

[21] E. Sultanik, P. J. Modi, and W. C. Regli. On modeling multiagent task scheduling as a distributed constraint optimization problem. In *IJCAI*, pages 1531–1536, 2007.

[22] S. Tahernejad, T. K. Ralphs, and S. T. DeNegre. A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation. *Math. Program. Comput.*, 12:529–568, 2020.

[23] M. Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge university press, 2011.

[24] UN-News. Somalia: Pirates attack UN aid ship, prompting call for action, June 2007. https://reliefweb.int/report/somalia/somalia-pirates-attack-un-aid-ship-prompting-call-action.

[25] United-Nation. United nations peacekeeping missions military maritime task force manual, Sept 2015.

[26] O. Vaněk, Z. Yin, M. Jain, B. Bošanskỳ, M. Tambe, and M. Pěchouček. Game-theoretic resource allocation for malicious packet detection in computer networks. In *AAMAS*, pages 905–912, 2012.

[27] J. I. Winn and K. H. Govern. Maritime pirates, sea robbers, and terrorists: New approaches to emerging threats. *Homeland Security Rev.*, 2: 131, 2008.

[28] C. Zhang and J. A. Shah. Co-optimizating multi-agent placement with task assignment and scheduling. In *IJCAI*, pages 3308–3314, 2016.