# Augmented Decision Spaces for Stackelberg Security Games: Sparse evolution begets scalability


Adam Żychowski
adam.zychowski@pw.edu.pl
Faculty of Mathematics and Information Science,
Warsaw University of Technology
Warsaw, Poland

Abhishek Gupta
abhishekgupta@iitgoa.ac.in
School of Mechanical Sciences,
Indian Institute of Technology Goa
Goa, India

Yew-Soon Ong
asysong@ntu.edu.sg
College of Computing and Data Science,
Nanyang Technological University
Centre for Frontier AI Research, Institute of High Performance Computing, Agency for Science, Technology and Research
Singapore

Jacek Mańdziuk
mandziuk@mini.pw.edu.pl
Faculty of Mathematics and Information Science,
Warsaw University of Technology
Warsaw, Poland
Faculty of Computer Science, AGH University of Krakow
Krakow, Poland



## Abstract

This paper introduces the Augmented Decision Space Optimization (ADSO) method for sparsity-driven optimization of mixed-strategies in Stackelberg Security Games (SSGs). The proposed method enhances traditional strategy optimization by combining binary variables to represent the presence of pure strategies with real-valued variables to refine their selection probabilities. Specifically, instead of waiting for an evolutionary process to gradually discover sparse solutions, the binary variables in ADS allow the real-valued variables to be switched on or off, thereby directly enforcing sparsity. This dual codification scheme achieves targets such as sparsification and computational efficiency in large-scale games. We demonstrate that ADS outperforms existing heuristic methods, offering superior solution quality, scalability, and stability. Empirical results across three different benchmark games show that ADS generates compact strategies with minimal computational overhead, achieving performance close to the exact methods. Furthermore, state-of-the-art results are obtained for problems where exact methods fail to scale effectively. Our framework promises broad applicability beyond SSGs, encompassing a wide range of game-theoretic and combinatorial optimization problems.



## CCS Concepts

• **Computing methodologies** → **Genetic algorithms**; *Machine learning approaches*; *Search methodologies*.

## Keywords

Stackelberg Security Games, Evolutionary Computation, Game Theory, Augmented Decision Space

**ACM Reference Format:**
Adam Żychowski, Abhishek Gupta, Yew-Soon Ong, and Jacek Mańdziuk. 2025. Augmented Decision Spaces for Stackelberg Security Games: Sparse evolution begets scalability. In *Proceedings of GECCO '25*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3712256.3726440




*GECCO '25, July 14–18, 2025, Málaga, Spain*



ACM ISBN 979-8-4007-1465-8/2025/07
https://doi.org/10.1145/3712256.3726440


## 1 Introduction

Optimization of mixed strategies in game theory has been a pivotal research area, particularly in applications involving decision-making under uncertainty [17] or adversarial conditions [12, 34]. Among these, Stackelberg Security Games (SSGs) [24] have emerged as a critical framework, widely used in domains such as infrastructure protection [13, 23], cybersecurity [25, 33], or wildlife conservation [8, 9]. SSGs involve a Leader-Follower paradigm, where the Leader commits to a strategy, and the Follower responds optimally. The primary challenge in solving these games is the identification of optimal mixed strategies for the Leader, with an emphasis on sparsity to alleviate practical resource constraints.

Existing approaches to mixed-strategy optimization in SSGs leverage mathematical programming [3, 28], evolutionary algorithms [36, 37], and learning-based techniques [22, 29]. Although these methods have demonstrated efficacy, they frequently encounter limitations. Solutions often become computationally intractable as the problem size increases, or they fail to effectively handle sparsity constraints—an essential feature of many real-world applications where a large number of redundant pure strategies have zero probability of being selected.

To address these challenges, we propose a novel evolutionary algorithm with an Augmented Decision Space (ADS) crafted for mixed-strategy SSGs. A mixed-strategy is usually represented as vector of real variables (forming a decision vector), where each variable specifies the probability of enacting the corresponding pure

strategy in a game setting. The core idea behind the ADS Optimization (ADSO) framework is to expand this search space by splitting each decision variable into a real part and a binary part. The real values encode the selection probabilities of the pure strategies, while the binary variables mandate the presence or absence of pure strategies in the mixture. In this way, the binary variables explicitly encode for sparsity despite doubling the dimensionality of the search space, and are therefore found to facilitate the enforcement of sparsity conditions. Interestingly, this dual codification scheme not only enables more efficient exploration of sparse solutions, but also enhances convergence in high-dimensional constrained optimization problems typical of real-world SSGs. While SSGs serve as a compelling use case due to their complexity and real-world relevance, the ADSO framework is expected to be equally suitable for other game-theoretic models and optimization problems with sparse solution characteristics. As a complement to established sparsity-promotion techniques (e.g., $L_1$ regularization) that modify objective functions to guide optimization algorithms towards gradual discovery of sparse solutions, ADSO helps directly impose the desired condition.

In this paper, the evolutionary update equations of the ADSO framework follow the principles of probabilistic model-based search [20] and a rigorous performance evaluation of the methodology is carried out. By leveraging sparsity to achieve scalability, this work contributes to advancing the state-of-the-art in mixed-strategy optimization for SSGs and related game-theoretic problems. In the special case of zero-sum games, the use of Danskin's theorem [1] allows to further accelerate the computation of evolutionary updates in the maximin setting, thereby enhancing scalability and robustness of the optimization process.

The key contributions of the paper are summarized below.

- **Augmented decision spaces** – introduction of a binary-real dual codification scheme that allows real-valued variables to be directly switched on or off during an evolutionary process, thereby enforcing sparsity and compactness upon mixed-strategy solutions;
- **Sparse evolution** – a novel algorithm that simultaneously evolves the binary and real variables for strategy selection and probability fine-tuning, enabling efficient exploration of large solution spaces;
- **Empirical validation** – comprehensive evaluation across three diverse benchmark games (Warehouse Games, Search Games, and FlipIt Games), showcasing consistent performance improvements over state-of-the-art methods;
- **General applicability** – establishing the potential of the ADSO framework for broader use in game-theoretic and optimization problems beyond Stackelberg Security Games.

## 2 Problem definition

A Stackelberg Security Game (SSG) involves two players: the *Leader* (L) and the *Follower* (F), interacting over $m$ time steps. At each time step, both players simultaneously select an action. A pure strategy $\sigma_P$ for player $P \in \{L, F\}$ is defined as a sequence of actions over the time steps: $\sigma_P = (a_1, a_2, \ldots, a_m)$. The set of all possible pure strategies for $P$ is denoted as $\Sigma_P$, and a mixed strategy $\pi_P \in \Pi_P$ is a probability distribution over $\Sigma_P$, where $\Pi_P$ represents the set of all mixed strategies for $P$.

Given a pair of mixed strategies $(\pi_L, \pi_F)$, the players' expected payoffs are $U_L(\pi_L, \pi_F)$ and $U_F(\pi_L, \pi_F)$, respectively. The objective of an SSG is to find the *Stackelberg Equilibrium* (SE), defined as the pair of strategies $(\pi_L, \pi_F)$ satisfying the following conditions:

$$\pi_L = \arg \max_{\bar{\pi}_L \in \Pi_L} U_L(\bar{\pi}_L, \mathrm{BR}(\bar{\pi}_L)), \tag{1}$$

$$\mathrm{BR}(\pi_L) = \arg \max_{\pi_F \in \Pi_F} U_F(\pi_L, \pi_F). \tag{2}$$

According to Eqs. 1 and 2, in SE, the Leader's expected payoff is maximized, under the assumption that the Follower always responds optimally, in terms of his/her payoff. According to the definition of the *Strong Stackelberg Equilibrium* (SSE), if the Follower has multiple optimal responses, the one that maximizes the Leader's payoff is selected. This assumption ensures the existence of an equilibrium, avoiding scenarios where the equilibrium may not exist due to tie-breaking in the Follower's favor.

In this model, both players select their strategies at the game's start (the Leader first, followed by the Follower) and commit to them for the game's duration. This sequential decision-making aligns with real-world scenarios where the Leader commits to a visible strategy that the Follower can observe and react to strategically. It is guaranteed that for any mixed strategy of the Leader, there exists at least one pure strategy for the Follower [6] that maximizes their payoff, which narrows the search space for optimal responses.

There exist numerous variants of Stackelberg Security Games (SSGs), which differ based on the specific scenarios they address, the set of available actions, the rules governing the computation of players' payoffs, and the application domains. The detailed rules and configurations for the three games analyzed in this paper are provided in Section 5.

## 3 Related work

The existing methods for solving sequential Stackelberg Security Games (SSGs) can be broadly classified into two categories: *exact methods* and *approximate approaches*.

### 3.1 Exact Methods

Exact solutions typically leverage Mixed-Integer Linear Programming (MILP) formulations to model SSGs as optimization problems with specific target functions and constraints. These solutions are precise but computationally expensive, especially for large-scale games.

**BC2015:** The BC2015 algorithm [3] extends the DOBBS algorithm [21], originally designed for one-step games, to handle extensive-form games [6] by transforming them into sequence-form representations. It significantly reduces the size of the linear program from exponential to linear relative to the game tree size.

**C2016:** The C2016 approach [28] further improves efficiency by focusing on the Stackelberg Extensive-Form Correlated Equilibrium (SEFCE). It computes SEFCE using linear programming and iteratively converges to the Stackelberg Equilibrium by restricting the Leader's signaling options. Experimental results demonstrate that C2016 is faster than BC2015, making it a preferred method for calculating reference solutions.

### 3.2 Heuristic Approaches

To address the limitations of exact methods in handling large games, heuristic approaches have been developed, which provide near-optimal solutions with better scalability.

**O2UCT:** O2UCT method [15, 16] uses Upper Confidence Bounds applied to Trees (UCT) [18], a Monte Carlo Tree Search-based approach [26], to iteratively sample the Follower's strategy space. The Leader's strategy is optimized under the assumption that the sampled Follower's strategy is optimal. O2UCT offers improved scalability compared to MILP methods while maintaining competitive solution quality.

**EASG:** The Evolutionary Algorithm for Security Games (EASG) [35, 36] employs evolutionary principles to optimize the Leader's strategy. It begins with a randomly generated initial population of solutions (chromosomes), each representing a potential mixed strategy. Through iterative generations, the algorithm applies crossover and mutation operators to diversify the population and improve solutions. The fitness of each chromosome is evaluated based on the Leader's payoff against the Follower's optimal response (which is identified by iterating over all possible Follower's pure strategies).

Crossover merges strategies from selected individuals to enhance exploitation, while mutation introduces new variations by altering actions in selected strategies, encouraging exploration of the search space. Chromosomes are refined to maintain simplicity, reducing the number of strategies with low probabilities. The selection process promotes high-fitness individuals directly to the next generation and uses a binary tournament to fill the population

**CoEvoSG:** The coevolutionary approach proposed in [37] addresses key inefficiency in the evaluation phase of the EASG. EASG exhaustively evaluates the Leader's strategy against all Follower strategies, which can be expensive for larger games or continuous Follower strategy spaces. Additionally, many Follower strategies are either weak or redundant, contributing little to the search for optimal responses. To overcome these challenges, coevolutionary algorithms maintain two populations — Leader and Follower strategies — which evolve in tandem. This reduces the computational burden by narrowing the search space to a subset of representative strategies for both players, while fostering competitive improvements between populations.

### 3.3 Limitations

A primary limitation of the evolutionary approaches (EASG and CoEvoSG) is their reliance on complex solution encodings and the use of specialized evolutionary operators, such as mutation and crossover, which are intricately designed for specific game types and their corresponding rules. Consequently, these methods cannot be seamlessly applied to other game types without substantial modifications to the encoding schemes, operator implementations, and algorithm parameterization. For instance, adapting them to Signaling Games [38] would require extensive reengineering.

Moreover, this dependence on problem-specific encodings and operators hinders the formulation of a general theoretical foundation for the optimization techniques, making it challenging to offer a definitive explanation of their empirical performance. As a result, EA-based methods are relatively weak in their theoretical grounding, a limitation acknowledged by their authors [36].

To alleviate these challenges, we propose a novel *Augmented Decision Space Optimization* (ADSO) method which offers greater generality and adaptability across different game types, and whose update equations follow the mathematical principles of probabilistic model-based evolutionary search.

## 4 Sparse Evolution by ADSO

This section presents our new approach to mixed-strategy optimization by searching ADS with evolutionary algorithms described by population distribution models [4]. The model defines a distribution over the set $\Pi_L$ of the Leader's mixed strategies, where a mixed strategy gives a probability distribution over the set $\Sigma_L$ of pure strategies. In short, the ADSO framework defines and optimizes a distribution over distributions.

Let $x_i$ be a decision variable that assigns a probability measure to the $i$-th pure strategy. Then, assuming $n$ pure strategies, $\bar{\pi}_L = [x_1, x_2, \ldots, x_n]$ where $x_i \geq 0$ and $\sum x_i = 1$ . In the proposed dual codification scheme, we split $x_i$ into two decision variables, namely $\hat{x}_i$ and $\tilde{x}_i$, such that:

$$x_i = \hat{x}_i \cdot \tilde{x}_i,$$

where $\tilde{x}_i$ is real-valued and $\hat{x}_i$ is a binary variable indicating the switching on ($\hat{x}_i = 1$) or switching off ($\hat{x}_i = 0$) of the $i$-th pure strategy in the mixture. Thus the value $\tilde{x}_i$ is interpreted as the probability measure of the $i$-th pure strategy only if $\hat{x}_i = 1$. By reformulating the decision space in this manner, the number of decision variables is effectively doubled, creating an augmented decision vector $\mathbf{x}' = (\hat{\mathbf{x}}, \tilde{\mathbf{x}}) = [\hat{x}_1, \tilde{x}_1, \hat{x}_2, \tilde{x}_2, \ldots, \hat{x}_n, \tilde{x}_n]$ that explicitly encodes for sparsity through the binary variables $\hat{\mathbf{x}}$.

During evolutionary search, we consider sampling the binary variables $\hat{x}_i$ independently from the Bernoulli distribution:

$$P(\hat{x}_i) = q_i^{\hat{x}_i} \cdot (1 - q_i)^{1-\hat{x}_i},$$

where $q_i$ represents the probability that $\hat{x}_i = 1$. Simultaneously, the vector $\tilde{\mathbf{x}}$ of real-valued decision variables $[\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n]$ is assumed to follow a multivariate normal distribution $p(\tilde{\mathbf{x}}) = \mathcal{N}(\mu_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}})$, parameterized by mean $\mu_{\tilde{\mathbf{x}}}$ and covariance $\Sigma_{\tilde{\mathbf{x}}}$. The overall probability distribution underlying the population of solutions in the ADS can then be expressed as:

$$p(\mathbf{x}') = \mathcal{N}(\mu_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}) \cdot \prod_{i=1}^{n} q_i^{\hat{x}_i} \cdot (1 - q_i)^{1-\hat{x}_i},$$

which is parameterized by $\mu_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}, q_1, q_2, \ldots, q_n$.

Let $f(\mathbf{x}')$ be the pay-off of the Leader given the mixed strategy encoded by the decision vector $\mathbf{x}'$. That is, $f(\mathbf{x}') = U_L(\bar{\pi}_L, \mathrm{BR}(\bar{\pi}_L))$, where $\bar{\pi}_L(\hat{\mathbf{x}}, \tilde{\mathbf{x}}) = \frac{1}{\sum \hat{x}_i \cdot \tilde{x}_i}[\hat{x}_1 \cdot \tilde{x}_1, \hat{x}_2 \cdot \tilde{x}_2, \ldots, \hat{x}_n \cdot \tilde{x}_n]$. Based on the unifying picture of information-geometric optimization [20], the objective function can be recast as maximizing the expected payoff for the Leader as:

$$\arg \max_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}, q_1, q_2, \ldots, q_n} F = \sum_{\forall \hat{\mathbf{x}}} \Big( \int f(\mathbf{x}') \cdot p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \Big) P(\hat{\mathbf{x}}). \qquad (3)$$

Following the Leibniz integral rule and the log-likelihood trick, the gradients of $F$ with respect to the parameters of the binary and the

normal distributions are:

$$\nabla_{q_1,\ldots,q_n} F = \sum_{\forall \hat{\mathbf{x}}} \Big( \int f(\mathbf{x}') \cdot p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \Big) \nabla_{q_1,\ldots,q_n} \Big( \sum_{i=1}^{n} (\hat{x}_i \log(q_i) + (1-\hat{x}_i)\log(1-q_i)) \Big) P(\hat{\mathbf{x}}) \quad (4)$$

and:

$$\nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}},\Sigma_{\tilde{\mathbf{x}}}} F = \sum_{\forall \hat{\mathbf{x}}} \Big( \int f(\mathbf{x}') \cdot \nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}},\Sigma_{\tilde{\mathbf{x}}}} (\log(\mathcal{N}(\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}))) \cdot p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \Big) P(\hat{\mathbf{x}}). \quad (5)$$

## 4.1 Evolutionary update equations

An evolution strategy employing the aforementioned gradients can be crafted by sampling and evaluating populations of candidate solutions from $\mathcal{N}(\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}) \prod_{i=1}^{n} q_i^{x_i} \cdot (1-q_i)^{1-x_i}$ to compute Monte Carlo estimates of (4) and (5). These estimates would then be used to iteratively update the population distribution model. To this end, the ADSO algorithm maintains two distributions:

- a multivariate normal distribution $\mathcal{N}(\mu_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}})$ for the real-valued variables $\tilde{\mathbf{x}}$, and
- independent Bernoulli distributions with parameters $q_1, \ldots, q_n$ for the binary vector $\hat{\mathbf{x}}$.

A new solution is generated by concatenating points sampled from each distribution. The concatenated solution corresponds to a mixed strategy $\bar{\pi}_L$ that is evaluated based on the Leader's expected payoff against the Follower's best response (see Eq. 1).

Monte Carlo estimates of (4) and (5) can, however, depict high variance due to the dimensionality of the ADS. To combat this curse of dimensionality, we make certain simplifying assumptions to arrive at the update equations of ADSO. Specifically, while approximating (4), the distribution $p(\tilde{\mathbf{x}})$ is assumed to be concentrated at its mean with $\Sigma_{\tilde{\mathbf{x}}} \to 0$, such that $\int f(\mathbf{x}') \cdot p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \to f(\hat{\mathbf{x}}, \boldsymbol{\mu}_{\tilde{\mathbf{x}}})$. Likewise, while approximating (5), the variance of the Bernoulli distribution is deemed sufficiently small such that $P(\hat{\mathbf{x}})$ may be lumped at the current best binary vector $\hat{\mathbf{x}}^*$. This reduces (5) to:

$$\nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}},\Sigma_{\tilde{\mathbf{x}}}} F = \int f(\hat{\mathbf{x}}^*, \tilde{\mathbf{x}}) \cdot \nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}},\Sigma_{\tilde{\mathbf{x}}}} (\log(\mathcal{N}(\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}))) \cdot p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}}. \quad (6)$$

ADSO iteratively switches between updating the binary distribution by (4) and the normal distribution by (6).

*4.1.1 Binary distribution updates.* In the gradient on the right of (4), the partial derivative with respect to probability $q_i$ is:

$$\frac{\partial}{\partial q_i} \Big( \sum_{i=1}^{n} (\hat{x}_i \log(q_i) + (1-\hat{x}_i)\log(1-q_i)) \Big) = \frac{\hat{x}_i - q_i}{q_i(1-q_i)}.$$

Given a population of $N$ binary vectors drawn from $P(\hat{\mathbf{x}})$, we therefore get the following stochastic gradient update:

$$q_i \leftarrow q_i + \eta_i \nabla_{q_i} F \approx q_i + \frac{\eta_i}{N} \sum_{j=1}^{N} f_s(\hat{\mathbf{x}}^j, \boldsymbol{\mu}_{\tilde{\mathbf{x}}}) \frac{\hat{x}_i^j - q_i}{q_i(1-q_i)},$$

where $f_s$ is a rank-based shaping of the original fitness function $f$ that provides invariance under all monotone transformations of $f$ [20]. The fittest binary vector in the population is denoted $\hat{\mathbf{x}}^*$. The fitness shaping employed in our implementation coincides with the so-called population-based incremental learning algorithm [2]. Moreover, $\eta_i = \eta\, q_i(1-q_i)$ to avoid instability due to singularities in the update when $q_i$ approaches 1 or 0. The final update equation is then:

$$q_i \leftarrow q_i + \frac{\eta}{N} \sum_{j=1}^{N} f_s(\hat{\mathbf{x}}^j, \boldsymbol{\mu}_{\tilde{\mathbf{x}}})(\hat{x}_i^j - q_i).$$

with $\eta$ being a constant learning rate.

*4.1.2 Real-valued distribution updates.* The stochastic gradient update of normal distributions governed by (6) is at the core of a family of natural evolution strategies [31]. In our implementation, we directly adopt the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [10, 11], a well-established member of this family, for real-valued distribution updates. Details of the algorithm are omitted for the sake of brevity.

## 4.2 Leveraging Danskin's theorem

A computational bottleneck in solving SSGs stems from the need to determine the Follower's best response corresponding to all candidate mixed strategies (encoded by $\mathbf{x}'$) of the Leader. Danskin's theorem, often used in the context of maximin problems [1], helps alleviate this bottleneck in the case of zero-sum SSGs.

A zero-sum problem is attained when $U_F = -U_L$, which transforms the SSG to:

$$\max_{\bar{\pi}_L \in \Pi_L} \min_{\bar{\pi}_F \in \Pi_F} U_L(\bar{\pi}_L, \bar{\pi}_F).$$

Considering $f(\hat{\mathbf{x}}^*, \tilde{\mathbf{x}}) = \min_{\bar{\pi}_F \in \Pi_F} U_L(\bar{\pi}_L(\hat{\mathbf{x}}^*, \tilde{\mathbf{x}}), \bar{\pi}_F)$, Danskin's theorem states [7]:

$$\frac{\partial f}{\partial \tilde{\mathbf{x}}} = \frac{\partial U_L(\bar{\pi}_L(\hat{\mathbf{x}}^*, \tilde{\mathbf{x}}), \mathrm{BR}(\bar{\pi}_L))}{\partial \tilde{\mathbf{x}}}, \quad (7)$$

as long as $U_L$ is a continuous function and $\mathrm{BR}(\bar{\pi}_L)$ is unique. In other words, changes to $\mathrm{BR}(\bar{\pi}_L)$ in a neighborhood of small variations to $\tilde{\mathbf{x}}$ can be ignored. Leveraging this insight, we first note that in the limit of small variations, (6) simplifies as:

$$\lim_{\Sigma_{\tilde{\mathbf{x}}} \to 0} \nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}}} F = \frac{\partial f(\hat{\mathbf{x}}^*, \boldsymbol{\mu}_{\tilde{\mathbf{x}}})}{\partial \boldsymbol{\mu}_{\tilde{\mathbf{x}}}} = \frac{\partial U_L(\bar{\pi}_L(\hat{\mathbf{x}}^*, \mu_{\tilde{\mathbf{x}}}), \mathrm{BR}(\bar{\pi}_L))}{\partial \boldsymbol{\mu}_{\tilde{\mathbf{x}}}}. \quad (8)$$

Consequently, while running CMA-ES, best responses need not be recomputed for every candidate solution sampled from a (narrow) normal distribution. In each iteration, the Follower's best response shall be derived only once corresponding to $\mathbf{x}' = (\hat{\mathbf{x}}^*, \boldsymbol{\mu}_{\tilde{\mathbf{x}}})$, reducing computational cost by a factor of $N$ given population size $N$.

# 5 Experimental setup

## 5.1 Benchmark games

*5.1.1 Zero-sum Warehouse Games.* (WHGs), as introduced in [14], are inspired by scenarios involving protection of real estate assets, such as warehouses or residential buildings. These games are formalized on undirected graphs consisting of $n$ vertices and evolve over $m$ discrete time steps. A subset of these vertices is designated as targets ($T$), representing critical points to be defended. In this representation, graph edges correspond to corridors, while vertices symbolize rooms. Figure 1 presents example of a WHG scenario.

At the beginning of the game, the Leader and the Follower are positioned at predetermined starting vertices. During each time step, a player can either move to a neighboring vertex (connected

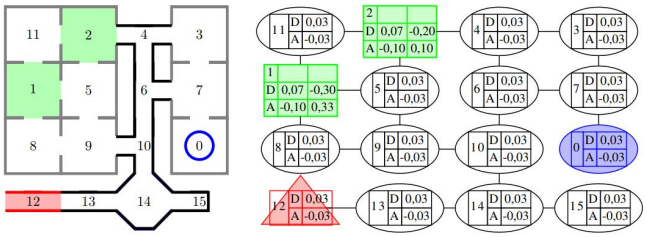


**Figure 1: Example WHG scenario [14]: the warehouse layout (left) and its corresponding graph representation (right) depict the payoffs associated with the players for the respective game outcomes. Green rectangular vertices represent targets, while the red triangular vertex and blue circular vertex denote the starting positions of the Follower and Leader, respectively.**

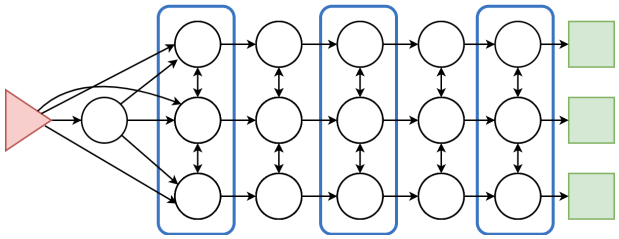

**Figure 2: Example of Search Game. A red triangle denotes the Follower's starting vertex, green rectangles are targets. Three rounded groups of vertices represent restricted subsets of nodes within which the Leader's units can freely move.**

by an edge) or remain in the currently occupied vertex. The game concludes under one of the following conditions:

(1) **Capture**: Both players occupy the same vertex $v$ at the same time step, signifying that the Follower is "caught." Payoffs are then assigned as $U^v_{D+} > 0$ for the Leader and $U^v_{A-} < 0$ for the Follower.
(2) **Successful Attack**: The Follower reaches a target vertex $t \in T$ without the Leader also occupying it. This outcome indicates a successful attack, and payoffs are given as $U^t_{D-} < 0$ for the Leader and $U^t_{A+} > 0$ for the Follower.
(3) **No Event**: If neither of the above occurs, both players receive a payoff of 0.

In our experiments we considered only zero-sum WHGs which means $U^v_{D+} = -U^v_{A-}$ and $U^t_{D-} = -U^t_{A+}$. To evaluate algorithms performance, 150 WHG instances were generated using various combinations of $m$ and $n$: $m \in \{3, 4, 5, 6, 8, 10\}$ and $n \in \{15, 20, 25, 30, 40\}$, with 5 games created for each $(m, n)$ pair. Player payoffs were sampled uniformly from the interval $[-1, 1]$. The number of target vertices was determined by the graph size as $|T| = \lceil \frac{n}{5} \rceil$. The underlying graph structures were generated using the Watts–Strogatz random graph model [30] with an average vertex degree $d_{\text{avg}} = 3$.

*5.1.2 Search Games.* (SEGs), introduced in [3], are played on directed graphs where the Follower's objective is to navigate from a fixed initial vertex to one from the set of designated target vertices. Unlike in Warehouse Games (WHG), SEGs feature a Leader possessing multiple defending units, each constrained to operate within a specific subset of the graph's vertices. Figure 2 presents an example of SEG scenario.

A key distinction between SEG and WHG is the property of partial observability. In SEGs, the Follower leaves detectable traces at the vertices they visit. These traces can be uncovered by a Leader's unit if it visits the same vertex in subsequent time steps. However, the Follower has the ability to erase traces by remaining in the vertex for an additional time step (i.e., staying in the same vertex for two or more consecutive time steps).

The conditions for terminating the game are similar to WHG: the Leader receives a positive payoff for apprehending the Follower, the Follower gains a reward for successfully reaching a target vertex without interception, or the game concludes with neutral payoffs if neither condition is met within the allotted time steps.

The inclusion of trace dynamics and multiple Leader units fundamentally differentiates SEGs from WHGs. Consequently, the Leader's strategy in SEGs is more complex, requiring not only a sequence of moves but also a reactive component to respond to the detection of the Follower's traces. This strategic extension is elaborated in [36] and adopted in this study. Additionally, unlike WHGs, SEGs are not zero-sum games.

For evaluation, 150 SEG instances were generated, with parameters including time steps $m \in \{3, 4, 5, 6, 8, 10\}$ and the number of graph vertices $n \in \{15, 20, 25, 30, 40\}$. The number of target vertices $|T|$ varied between 2 and 6.

*5.1.3 FlipIt Games.* (FIGs), as introduced in [27], are inspired by cybersecurity scenarios where the Follower seeks to gain control over elements of the network infrastructure (e.g., computers, routers, mobile devices) while the Leader attempts to reclaim control of compromised units.

These games are played on directed graphs with $n$ vertices over $m$ discrete time steps. In each time step, both players simultaneously choose one vertex to attempt to take control of (referred to as a *flip* action). Initially, all vertices are under the Leader's control, and only a subset of vertices, known as entry nodes, is accessible to the Follower. This setup reflects real-world scenarios where certain parts of a network (e.g., publicly accessible interfaces) are exposed to external threats. The Follower begins their intrusion from one of these entry nodes.

A flip attempt is successful if two conditions are simultaneously satisfied:

- The player already controls at least one of the predecessor vertices of the node to be flipped (unless the node is an entry node).
- The current controller of this node does not simultaneously attempt to flip that same node.

Each vertex is associated with two values: a positive reward for controlling the node and a negative cost for attempting a flip action. The final payoff for a player is calculated as the sum of the rewards from all nodes controlled by that player over all time steps, minus the cumulative costs of all flip attempts (whether successful or not) made during the game. Figure 3 presents a sample FIG scenario.

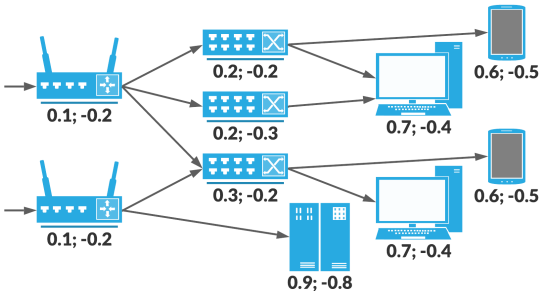


**Figure 3: Example FIG scenario [37] with two entry nodes (routers) on the left. Numbers below each component denote a reward for controlling the node (left) and a cost of a flip attempt (right).**

In the experimental setup, 150 FIG instances were generated with parameters $m \in \{3, 4, 5, 6, 8, 10\}$ and $n \in \{5, 10, 15, 20, 25\}$. For each $(m, n)$ pair, 5 games were tested with random payoffs, where rewards were sampled from the interval $(0, 1)$ and costs from $(-1, 0)$. Graphs were constructed using the Watts–Strogatz model [30] with an average vertex degree $d_{\text{avg}} = 3$.

The experiments were conducted under the *No-Info* variant [5], meaning players are unaware of whether their flip attempts succeed. Consequently, their strategies are independent of the opponent's actions, reflecting a high level of uncertainty in decision-making.

## 5.2 Algorithm setup

To explore the real-valued part of the augmented decision space, we employed the well-established Covariance Matrix Adaptation Evolution Strategy (CMA-ES). CMA-ES was selected due to its popularity, practical effectiveness and robustness to hyperparameter settings. We utilized the CMA-ES implementation available at github.com/yn-cloud/CMAES.NET. Our overall implementation of ADSO with dual codification scheme was developed in C#, and the source code will be made publicly accessible upon the publication of the paper. All computational experiments were conducted on an Intel Xeon Silver 4116 processor with a clock speed of 2.10 GHz.

For all tested evolutionary algorithms, the population size was fixed at 200. The stop condition included two cases: either a maximum of $10^5$ fitness function evaluations (corresponding to the Leader's expected payoff calculations) were performed or 20 consecutive generations without improvement in the best solution took place. For the ADSO algorithm, the learning rates were set to $10^{-4}$. We conducted additional experiments varying the key hyperparameters and observed that ADSO consistently maintains its performance with only minor variations in convergence speed.

An additional important optimization is the solution representation scheme. Since all the games we consider are played on graphs, we optimized the probabilities assigned to graph edges rather than optimizing the probabilities of each pure strategy. Specifically, this means assigning probabilities to the selection of edge $e$ in time step $m_i$. This approach significantly reduced the size of the search space ($m \times |E|$ instead of $(d_{avg})^m$, where $m$ is the number of game time steps, $|E|$ is the number of graph edges and $(d_{avg})^m$ is the average degree of graph nodes).

No extensive parameter optimization was conducted to avoid biasing the algorithm toward any specific game setup. Instead, the parameters were chosen based on a set of 5 randomly generated WHGs and insights derived from the literature.

# 6 Results

The proposed Augmented Decision Space Optimization (ADSO) algorithm was evaluated against five other methods: C2016, O2UCT, EASG, and CoEvoSG, described in Section 3, and CMA-ES. CMA-ES was directly applied to optimize the Leader's decision space, represented as a probability vector over all possible pure strategies.

The methods were compared based on three primary criteria: solution quality (Leader's expected payoff), computational scalability, and stability. The evaluation utilized 450 game instances (150 instances for each of WHG, SEG, and FIG), as described in the previous section. All reported results are averaged over 30 independent runs per game instance and further aggregated by game type.

## 6.1 Payoffs

Tables 1, 2, and 3 summarize the average Leader's payoffs for WHG, SEG, and FIG games, respectively. The results are presented as a function of the number of graph nodes ($n$) and time steps ($m$). A dash ("-") indicates that the corresponding algorithm failed to solve certain test game instances within the computational limit of 100 hours per instance.

**Table 1: Average Leader's payoffs with respect to the number of graph nodes (top) and time steps (bottom) for Warehouse Games. The best results are bolded.**

| $n$ | C2016 | O2UCT | EASG | CoEvoSG | CMA-ES | ADSO |
|---|---|---|---|---|---|---|
| 15 | **0.052** | 0.051 | 0.051 | 0.050 | 0.049 | 0.051 |
| 20 | **0.054** | 0.053 | 0.052 | 0.051 | 0.050 | 0.053 |
| 25 | **0.048** | 0.046 | 0.045 | 0.044 | 0.044 | 0.047 |
| 30 | - | 0.044 | 0.042 | 0.040 | 0.040 | **0.045** |
| 40 | - | - | 0.040 | 0.038 | 0.038 | **0.041** |
| $m$ | C2016 | O2UCT | EASG | CoEvoSG | CMA-ES | ADSO |
| 3 | **0.043** | **0.043** | **0.043** | **0.043** | **0.043** | **0.043** |
| 4 | **0.052** | 0.050 | 0.050 | 0.049 | 0.048 | 0.051 |
| 5 | **0.055** | 0.054 | 0.053 | 0.052 | 0.051 | 0.054 |
| 6 | **0.058** | 0.056 | 0.054 | 0.052 | 0.052 | 0.055 |
| 8 | - | **0.053** | 0.051 | 0.049 | 0.048 | 0.052 |
| 10 | - | - | **0.048** | 0.046 | 0.046 | **0.048** |

The results demonstrate that direct application of CMA-ES to the Leader's decision space is suboptimal. A detailed analysis indicates that the mixed strategies produced by this approach often include a large number of pure strategies with negligible probabilities. In contrast, the optimal mixed strategies computed by C2016 typically include no more than 10 pure strategies, with an average of 5.74. Many of the CMA-ES pure strategies with minimal probabilities are redundant and should ideally be excluded from the final solution. However, the standard CMA-ES approach, without augmentation of the decision space, is unable to effectively eliminate these superfluous strategies. This limitation is a key factor contributing to the inferior performance of CMA-ES compared to ADSO.

**Table 2: Average Leader's payoffs with respect to the number of graph nodes (top) and time steps (bottom) for Search Games. The best results are bolded.**

| $n$ | C2016 | O2UCT | EASG | CoEvoSG | CMA-ES | ADSO |
|---|---|---|---|---|---|---|
| 15 | **0.122** | 0.116 | 0.115 | 0.115 | 0.114 | 0.119 |
| 20 | **0.117** | 0.112 | 0.106 | 0.104 | 0.104 | 0.114 |
| 25 | - | 0.123 | 0.117 | 0.116 | 0.115 | **0.124** |
| 30 | - | - | 0.136 | 0.135 | 0.134 | **0.137** |
| 40 | - | - | - | 0.152 | 0.151 | **0.154** |
| $m$ | C2016 | O2UCT | EASG | CoEvoSG | CMA-ES | ADSO |
| 3 | **0.137** | 0.126 | 0.118 | 0.118 | 0.117 | 0.128 |
| 4 | **0.124** | 0.113 | 0.110 | 0.109 | 0.108 | 0.117 |
| 5 | **0.106** | 0.093 | 0.090 | 0.087 | 0.087 | 0.101 |
| 6 | - | 0.129 | 0.123 | 0.123 | 0.123 | **0.134** |
| 8 | - | - | 0.112 | 0.111 | 0.110 | **0.117** |
| 10 | - | - | - | 0.144 | 0.144 | **0.149** |

**Table 3: Average Leader's payoffs with respect to the number of graph nodes (top) and time steps (bottom) for FlipIt Games. The best results are bolded.**

| $n$ | C2016 | O2UCT | EASG | CoEvoSG | CMA-ES | ADSO |
|---|---|---|---|---|---|---|
| 5 | **0.890** | 0.887 | 0.886 | 0.886 | 0.880 | 0.889 |
| 10 | **0.854** | 0.851 | 0.847 | 0.846 | 0.839 | 0.852 |
| 15 | **0.811** | 0.807 | 0.802 | 0.800 | 0.795 | 0.809 |
| 20 | - | 0.784 | 0.78 | 0.775 | 0.774 | **0.786** |
| 25 | - | - | - | 0.748 | 0.745 | **0.757** |
| $m$ | C2016 | O2UCT | EASG | CoEvoSG | CMA-ES | ADSO |
| 3 | **0.823** | 0.821 | 0.820 | 0.817 | 0.816 | 0.822 |
| 4 | **0.817** | 0.812 | 0.808 | 0.805 | 0.802 | 0.814 |
| 5 | **0.810** | 0.801 | 0.798 | 0.791 | 0.787 | 0.806 |
| 6 | - | 0.792 | 0.792 | 0.791 | 0.789 | **0.800** |
| 8 | - | 0.785 | 0.784 | 0.781 | 0.776 | **0.788** |
| 10 | - | - | 0.780 | 0.778 | 0.777 | **0.780** |

The proposed ADSO method addresses this issue by incorporating an augmented binary component, which improves scalability by promoting sparsity in the resulting mixed strategies. On average, ADSO-generated strategies contain 6.76 pure strategies, a significant improvement in sparsity compared to standard CMA-ES (32.73 pure strategies in average). Similarly, EASG and CoEvoSG achieve increased sparsity through more complex mechanisms involving crossover operations, in which pure strategies are probabilistically removed based on their likelihood of being selected (inversely proportional to their probability). These mechanisms result in final strategies with an average of 6.83 pure strategies for EASG and 7.12 for CoEvoSG.

Among the heuristic methods, ADSO achieved the best performance across all benchmarks, producing results close to those of the exact C2016 algorithm. For WHG, SEG, and FIG games, ADSO's superiority over other heuristic methods was statistically significant in 97/150 (64.7%), 114/150 (76.0%), and 127/150 (84.7%) instances, respectively. Statistical significance was established using a one-tailed paired $t$-test at a significance level of 0.05, with normality verified using the Shapiro-Wilk test.

The exact MILP method (C2016) was able to solve 60 WHG, 60 SEG, and 30 FIG test instances within the computational time limit. Table 4 reports the number of instances in which each method achieved the same optimal strategy as C2016 (i.e., where the difference in the Leader's payoff was less than $\varepsilon = 0.0001$). ADSO produced optimal solutions in 49/60 (81.7%) WHG, 41/60 (68.3%) SEG, and 21/30 (70.0%) FIG cases. The average deviations between ADSO's outcomes and the optimal results were 0.0018 for WHG, 0.0087 for SEG, and 0.0107 for FIG.

**Table 4: The number of games in which each method successfully identified the optimal strategy (achieved a Leader's payoff difference of less than $\varepsilon = 0.0001$ compared to the C2016 solution). The best results (excluding C2016) are bolded.**

| | C2016 | O2UCT | EASG | CoEvoSG | CMA-ES | ADSO |
|---|---|---|---|---|---|---|
| WHG | 60 (100%) | 39 (65.0%) | 43 (71.7%) | 38 (63.3%) | 15 (25.0%) | **49 (81.7%)** |
| SEG | 60 (100%) | 36 (60.0%) | 26 (43.3%) | 17 (28.3%) | 12 (20.0%) | **41 (68.3%)** |
| FIG | 30 (100%) | 17 (56.7%) | 19 (63.3%) | 16 (53.3%) | 5 (16.7%) | **21 (70.0%)** |

## 6.2 Scalability analysis

Figure 4 illustrates the scalability analysis of the tested methods as a function of the number of graph nodes. Among them, CoEvoSG demonstrates near-constant computation time, regardless of game size. In contrast, other heuristic methods (O2UCT, EASG, CMA-ES, ADSO) scale approximately linearly, while the exact method (C2016) exhibits exponential scaling.

The primary computational cost for the heuristic methods arises from evaluating the Leader's strategy. This process involves determining the optimal Follower response, which is typically achieved by iterating over all Follower's pure strategies to identify the best one. CoEvoSG improves this by maintaining fixed-size populations for both the Leader and the Follower, irrespective of the game parameters. Leader strategies are evaluated only against those adversarial counterparts in the Follower population, allowing it to avoid full evaluations against all potential adversary strategies and highly reducing the dependence on the game size. This advantage is the key factor behind CoEvoSG's superior scalability in terms of computation time, but at the cost of some deterioration in the quality of the best solutions found.

Among the remaining heuristic methods, ADSO achieves the lowest computation time, outperforming CMA-ES. This result may appear counterintuitive, as CMA-ES maintains a single distribution, whereas ADSO involves an additional binary distribution. A deeper analysis reveals that ADSO's efficiency stems from the reduced time spent on strategy evaluations. As noted earlier, mixed strategies generated by ADSO typically include significantly fewer pure strategies compared to those produced by CMA-ES. By reducing the number of Leader-Follower strategy pairs through sparsity-driven optimization, ADSO minimizes the computational overhead required for payoff calculations, outperforming existing heuristic methods.

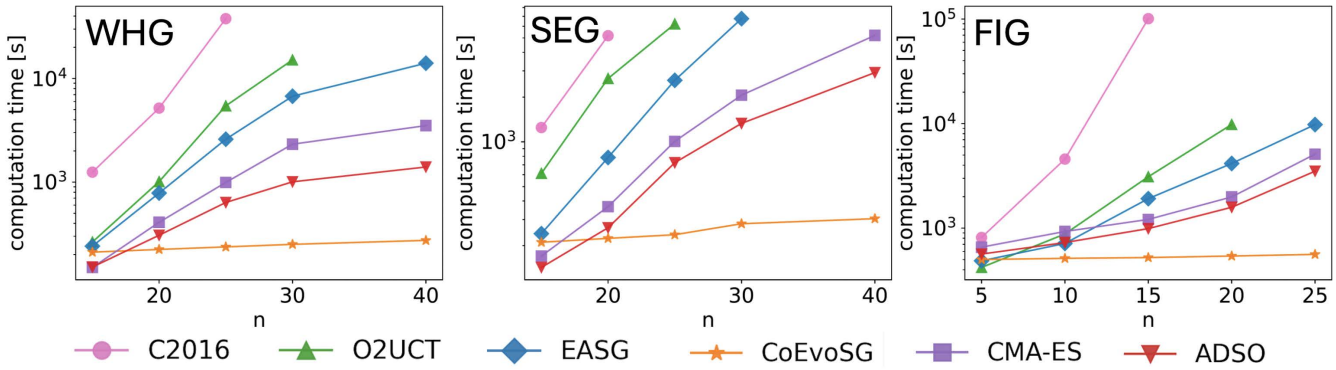


**Figure 4: Scalability of tested methods: computation time (logarithmic scale) with respect to the number of graph nodes ($n$).**

### 6.3 Stability

Since the proposed method is non-deterministic, its ability to achieve optimal solutions (as discussed in the previous subsection) is not sufficient for a comprehensive evaluation. An equally important aspect is the algorithm's consistency in reproducing high-quality results across multiple runs.

To assess the stability of ADSO, we analyzed the standard deviations of the Leader's payoffs across 30 runs per game. In 113 out of 450 tested games (25.1%), the standard deviation was equal to 0, indicating perfect stability. The mean standard deviation across all games was 0.0054, with the highest observed value being 0.1712, which corresponds to 33.1% of the possible payoff range for that particular game.

Table 5 presents the mean and maximum standard deviations for all tested methods.

**Table 5: The mean and maximum standard deviations of the Leader's payoffs across 30 runs, calculated over all tested games. The best results (excluding C2016) are bolded.**

| | | C2016 | O2UCT | EASG | CoEvoSG | CMA-ES | ADSO |
|---|---|---|---|---|---|---|---|
| WHG | mean | 0.000 | **0.046** | 0.045 | 0.051 | 0.051 | 0.047 |
| | max | 0.000 | **0.068** | 0.077 | 0.081 | 0.087 | 0.076 |
| SEG | mean | 0.000 | 0.059 | **0.054** | 0.068 | 0.058 | 0.057 |
| | max | 0.000 | 0.111 | 0.105 | 0.120 | 0.119 | **0.102** |
| FIG | mean | 0.000 | **0.059** | 0.066 | 0.071 | 0.061 | **0.059** |
| | max | 0.000 | 0.175 | 0.184 | 0.189 | 0.192 | **0.171** |

Among the 111 games where ADSO found the optimal solution, the best solution was reproduced in all 30 runs in 84 cases (76.7%). For 97 of these games (87.4%), the optimal solution was achieved in more than 90% of the runs. At the other extreme, in 3 cases (2.4%), the optimal strategy was identified in only one run. These results suggest that, despite the inherent randomness of the method, ADSO demonstrates relatively high stability and consistently reproduces optimal solutions for the majority of tested games.

## 7 Conclusions

In this work, we introduced a novel approach for solving mixed-strategy optimization problems in Stackelberg Security Games (SSGs) - Augmented Decision Space Optimization (ADSO) framework. The core innovation of the proposed methodology lies in the use of an augmented decision space, which incorporates both binary variables to encode the presence of pure strategies and real-valued probabilities to refine their usage. This dual representation leverages sparsity to achieve scalability.

The proposed solution was tested on 3 different SSGs: Warehouse Games, Search Games, and FlipIt Games with 450 game instances overall. The ADSO framework demonstrates substantial advantages over existing methods, consistently achieving the results close to the optimal solutions produced by exact methods, while maintaining superior performance compared to state-of-the-art heuristic approaches. The proposed method effectively reduces redundant strategies in the mixed solution space, generating more compact strategies. By leveraging sparsity, ADSO also reduces the computational overhead associated with strategy evaluations. Despite its probabilistic nature, ADSO displays high consistency across multiple runs, with a low variance in the quality of solutions.

We believe the ADSO framework represents a significant step forward in the optimization of mixed strategies for SSGs. Beyond SSGs, the ADSO framework may prove to be well-suited for other game-theoretic models and combinatorial optimization problems where sparsity and scalability are crucial, such as adversarial multi-agent reinforcement learning [32] and large-scale resource allocation problems [19].

Future work could explore extending ADSO to other game-theoretic models, such as signaling games or dynamic multi-stage interactions, and integrating this framework with reinforcement learning techniques for adaptive decision-making in real-time applications.

## Acknowledgments

Adam Żychowski was funded by the Warsaw University of Technology within the Excellence Initiative: Research University (IDUB) program. This research is also partially supported by the Ramanujan Fellowship from the Science and Engineering Research Board, Government of India (Grant No. RJF/2022/000115), the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-RP-2022-031) and the National Science Centre, Poland, grant number 2023/49/B/ST6/01404.

## References


[1] Abdullah Al-Dujaili, Shashank Srikant, Erik Hemberg, and Una-May O'Reilly. 2019. On the application of Danskin's theorem to derivative-free minimax problems. In *AIP Conference Proceedings*, Vol. 2070. AIP Publishing, 020026–1 – 020026–4.

[2] Shummet Baluja. 1994. *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning*. Carnegie Mellon University.

[3] Branislav Bošanský and Jiří Čermak. 2015. Sequence-Form Algorithm for Computing Stackelberg Equilibria in Extensive-Form Games. In *Proceedings of the 29th AAAI Conference*. 805–811.

[4] Josu Ceberio, Alexander Mendiburu, and Jose A Lozano. 2024. A roadmap for solving optimization problems with estimation of distribution algorithms. *Natural Computing* 23, 1 (2024), 99–113.

[5] Jakub Černỳ, Branislav Bošanskỳ, and Christopher Kiekintveld. 2018. Incremental strategy generation for Stackelberg equilibria in extensive-form games. In *Proceedings of the 19th ACM Conference on Economics and Computation*. 151–168.

[6] Vincent Conitzer and Tuomas Sandholm. 2006. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*. 82–90.

[7] John M Danskin. 2012. *The theory of max-min and its application to weapons allocation problems*. Vol. 5. Springer Science & Business Media.

[8] Fei Fang, Thanh H Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Milind Tambe, and Andrew Lemieux. 2016. Deploying PAWS: Field optimization of the protection assistant for wildlife security. In *Proceedings of the 28th Innovative Applications of Artificial Intelligence Conference*. 3966–3973.

[9] Fei Fang, Peter Stone, and Milind Tambe. 2015. When Security Games Go Green: Designing Defender Strategies to Prevent Poaching and Illegal Fishing.. In *IJCAI*, Vol. 15. 2589–2595.

[10] Nikolaus Hansen. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772* (2016).

[11] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation* 9, 2 (2001), 159–195.

[12] Edwin Ho, Arvind Rajagopalan, Alex Skvortsov, Sanjeev Arulampalam, and Mahendra Piraveenan. 2022. Game Theory in defence applications: A review. *Sensors* 22, 3 (2022), 1032.

[13] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordónez. 2010. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces* 40, 4 (2010), 267–290.

[14] Jan Karwowski and Jacek Mańdziuk. 2019. A Monte Carlo Tree Search approach to finding efficient patrolling schemes on graphs. *European Journal of Operational Research* 277 (2019), 255–268.

[15] Jan Karwowski and Jacek Mańdziuk. 2019. Stackelberg Equilibrium Approximation in General-Sum Extensive-Form Games with Double-Oracle Sampling Method. In *Proceedings of the 18th AAMAS conference*. 2045–2047.

[16] Jan Karwowski and Jacek Mańdziuk. 2020. Double-oracle sampling method for Stackelberg Equilibrium approximation in general-sum extensive-form games. In *Proceedings of the 34th AAAI conference*, Vol. 34. 2054–2061.

[17] Mykel J Kochenderfer. 2015. *Decision making under uncertainty: theory and application*. MIT press.

[18] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*. Springer, 282–293.

[19] Deepak Narayanan, Fiodar Kazhamiaka, Firas Abuzaid, Peter Kraft, Akshay Agrawal, Srikanth Kandula, Stephen Boyd, and Matei Zaharia. 2021. Solving large-scale granular resource allocation problems efficiently with pop. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*. 521–537.

[20] Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. 2017. Information-geometric optimization algorithms: A unifying picture via invariance principles. *Journal of Machine Learning Research* 18, 18 (2017), 1–65.

[21] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. 2008. Playing games for security: an efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the 7th AAMAS conference*. 895–902.

[22] Andrew Perrault, Bryan Wilder, Eric Ewing, Aditya Mate, Bistra Dilkina, and Milind Tambe. 2020. End-to-end game-focused learning of adversary behavior in security games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1378–1386.

[23] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. 2012. Protect: A deployed game theoretic system to protect the ports of the united states. In *Proceedings of the 11th AAMAS conference*. 13–20.

[24] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. 2018. Stackelberg Security Games: Looking Beyond a Decade of Success. In *Proceedings of the 27th IJCAI Conference*. 5494–5501.

[25] Arunesh Sinha, Thanh H Nguyen, Debarun Kar, Matthew Brown, Milind Tambe, and Albert Xin Jiang. 2015. From physical security to cybersecurity. *Journal of Cybersecurity* 1, 1 (2015), 19–35.

[26] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. 2023. Monte Carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review* 56, 3 (2023), 2497–2562.

[27] Marten Van Dijk, Ari Juels, Alina Oprea, and Ronald L Rivest. 2013. FlipIt: The game of "stealthy takeover". *Journal of Cryptology* 26, 4 (2013), 655–713.

[28] Jiří Čermák, Branislav Bošanský, Karel Durkota, Viliam Lisý, and Christopher Kiekintveld. 2016. Using Correlated Strategies for Computing Stackelberg Equilibria in Extensive-Form Games. In *Proceedings of the 30th AAAI Conference*. 439–445.

[29] Kai Wang, Andrew Perrault, Aditya Mate, and Milind Tambe. 2020. Scalable Game-Focused Learning of Adversary Models: Data-to-Decisions in Network Security Games.. In *AAMAS*. 1449–1457.

[30] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393, 6684 (1998), 440–442.

[31] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. 2014. Natural evolution strategies. *The Journal of Machine Learning Research* 15, 1 (2014), 949–980.

[32] Lantao Yu, Jiaming Song, and Stefano Ermon. 2019. Multi-agent adversarial inverse reinforcement learning. In *International Conference on Machine Learning*. PMLR, 7194–7201.

[33] Yunxiao Zhang and Pasquale Malacaria. 2021. Bayesian Stackelberg games for cyber-security decision support. *Decision Support Systems* (2021), 113599.

[34] Yan Zhou, Murat Kantarcioglu, and Bowei Xi. 2019. A survey of game theoretic approach for adversarial machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9, 3 (2019), e1259.

[35] Adam Żychowski and Jacek Mańdziuk. 2020. A Generic Metaheuristic Approach to Sequential Security Games. In *Proceedings of the 19th AAMAS*. 2089–2091.

[36] Adam Żychowski and Jacek Mańdziuk. 2021. Evolution of Strategies in Sequential Security Games. In *Proceedings of the 20th AAMAS conference*. 1434–1442.

[37] Adam Żychowski and Jacek Mańdziuk. 2023. Coevolution of players strategies in security games. *Journal of Computational Science* 68 (2023), 101980.

[38] Adam Żychowski, Jacek Mańdziuk, Elizabeth Bondi, Aravind Venugopal, Milind Tambe, and Balaraman Ravindran. 2022. Evolutionary approach to Security Games with signaling. In *Proceedings of the 31st IJCAI Conference*. 620–627.