Scalability via Sparsity in Stackelberg Security Games: An Augmented Decision Space Approach

ADAM ŻYCHOWSKI, Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
ABHISHEK GUPTA, School of Mechanical Sciences, Indian Institute of Technology Goa, India
YEW-SOON ONG, College of Computing and Data Science, Nanyang Technological University, Singapore
Centre for Frontier AI Research, Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore
JACEK MAŃDZIUK, Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
Faculty of Computer Science, AGH University of Krakow, Poland

This paper proposes the Augmented Decision Space Optimization (ADSO) framework for sparsity-driven optimization of mixed strategies in Stackelberg Security Games (SSGs). Unlike conventional evolutionary search processes that gradually converge toward sparse solutions, ADSO integrates binary variables to explicitly encode the inclusion or exclusion of pure strategies, while real-valued variables fine-tune their selection probabilities. This dual representation directly enforces sparsity and facilitates computational efficiency in large-scale problem instances. Extensive empirical studies on three benchmark games demonstrate that ADSO consistently produces compact strategies with minimal computational cost, achieving performance comparable to exact methods where they are feasible, and delivering state-of-the-art results in settings beyond the reach of such methods. Apart from SSGs, the framework exhibits strong potential for broader application to other game-theoretic and combinatorial optimization problems characterized by sparse solutions.

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

Additional Key Words and Phrases: Stackelberg Security Games, Evolutionary Computation, Game Theory, Augmented Decision Space

ACM Reference Format:
Adam Żychowski, Abhishek Gupta, Yew-Soon Ong, and Jacek Mańdziuk. 2026. Scalability via Sparsity in Stackelberg Security Games: An Augmented Decision Space Approach. *ACM Trans. Evol. Learn. Optim.* 0, 0, Article 000 (2026), 25 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Optimization of mixed strategies in game theory has been a pivotal research area, particularly in applications involving decision-making under uncertainty [Kochenderfer 2015] or adversarial conditions [Ho et al. 2022; Zhou et al. 2019].

Authors' Contact Information: 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, Singapore
and 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
and Faculty of Computer Science, AGH University of Krakow, Krakow, Poland.

Manuscript submitted to ACM

Among these, Stackelberg Security Games (SSGs) [Sinha et al. 2018] have emerged as a critical framework, widely used in domains such as infrastructure protection [Jain et al. 2010b; Shieh et al. 2012], cybersecurity [Sinha et al. 2015; Zhang and Malacaria 2021], or wildlife conservation [Fang et al. 2016, 2015]. 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.

SSGs have seen extensive deployment in domains where limited defensive resources must be allocated against strategic adversaries. They have been successfully applied in a variety of real-world scenarios, such as scheduling canine patrols at Los Angeles International Airport using the ARMOR system [Jain et al. 2010b], Federal Air Marshal scheduling (system IRIS) [Tsai et al. 2009], protecting Boston Harbor by the US Coast Guard with PROTECT [Ordónez et al. 2012], and preventing poaching in the Queen Elizabeth National Park in Uganda with PAWS [Fang et al. 2016]. These systems demonstrate the ability of Security Game models to handle large action spaces, incorporate adversary behavior, and produce randomized, deployable strategies that improve operational effectiveness and robustness against adaptive threats.

Existing approaches to mixed-strategy optimization in SSGs leverage mathematical programming [Bošanský and Čermak 2015; Čermák et al. 2016], evolutionary algorithms [Żychowski and Mańdziuk 2021a, 2023], and learning-based techniques [Perrault et al. 2020; Wang et al. 2020]. 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 [Ollivier et al. 2017] 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 a corollary to Danskin's theorem [Danskin 1966] - which provides guarantees about the derivative of the maximum of a differentiable function - allows us to further accelerate the computation of evolutionary updates in the minimax setting, thereby enhancing scalability and robustness of the optimization process. It is worth noting that the guarantees provided by the theoretical

result in the gradient-based setting are reconciled with our (gradient-free) evolutionary algorithm as its probabilistic model-based formulation provides stochastic gradient estimations that can be used in lieu of the true gradient [Al-Dujaili et al. 2019].

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.

This paper is an extension of an earlier conference publication at GECCO 2025 [Żychowski et al. 2025]. The domain of SSGs remains fairly underexplored in the evolutionary computation community. Therefore, compared to its predecessor, this version enhances the introduction to SSGs with a real-world application scenario and a pedagogical numerical example. The Related Work section has been expanded to offer a more comprehensive coverage of methods to tackle various SSG formulations; a section that provides an overview of the limitations of these methods, thereby firmly positioning our contributions with ADSO in the context of existing literature, has also been included. The description of the ADSO algorithm has been refined, with further discussions on the steps leading up to the final evolutionary update equations. New numerical results are presented to highlight the effectiveness of algorithmic innovations in terms of inducing solution sparsity via dual codification and reducing computation time by leveraging Danskin's theorem.

2 Problem Definition

An 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 σ_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 Σ_P, and a mixed strategy $\pi_P \in \Pi_P$ is a probability distribution over Σ_P, where Π_P represents the set of all mixed strategies for P.

Given a pair of mixed strategies (π_L, π_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 $(\pi_L, \mathrm{BR}(\pi_L))$ satisfying the following conditions:

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

$$\mathrm{BR}(\bar{\pi}_L) \in \arg\max_{\bar{\pi}_F \in \Pi_F} U_F(\bar{\pi}_L, \bar{\pi}_F). \tag{2}$$

According to (1) and (2), the Leader's expected payoff is maximized under the assumption that the Follower always responds optimally, in terms of his/her payoff. The definition of the *Strong Stackelberg Equilibrium* (SSE) further prescribes that if the Follower has multiple optimal responses, the one that maximizes the Leader's payoff is selected as the best response $\mathrm{BR}(\cdot)$. 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 [Conitzer and Sandholm 2006] that maximizes their payoff, which narrows the search space for optimal responses.

There exist numerous variants of 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 6.

2.1 A Real-World Example

To illustrate the SSG model, consider a wildlife protection scenario aimed at preventing poaching in a national park. The Leader (park rangers) must allocate limited patrol resources across multiple areas of the park, each containing wildlife of varying ecological and economic value. The Follower (poachers) observes or learns the ranger's patrol strategy over time and then chooses a target area to maximize their own gain, which depends on the likelihood of capturing valuable animals and the risk of being caught. The Leader's payoff decreases when poachers successfully hunt in an area, while the Follower's payoff decreases when caught or deterred. Each pure strategy for the Leader corresponds to a specific patrol schedule, and each pure strategy for the Follower corresponds to selecting a single area to poach. In the SSG framework, the Leader commits to a randomized mixed strategy over patrols, anticipating the Follower's best response. The optimal strategy balances protecting the most valuable and vulnerable areas with the need to remain unpredictable, thereby maximizing the expected security outcome.

2.2 Numerical Example

Consider a one-step game (m=1) with three areas A, B, C. The Leader (L) commits to a mixed strategy $\pi_L \in \Pi_L$ over patrol locations; denote $\pi_L(i)$ as the probability of patrolling area $i \in \{A, B, C\}$. The Follower (F) chooses a pure strategy $e_i \in \Sigma_F$ ("attack i").

Let the poacher's value vector be $v = (12, 8, 4)$ for successfully attacking A, B, and C, respectively, and 0 for an unsuccessful attack. Then $U_F(\pi_L, e_i) = v_i(1 - \pi_L(i))$ and we take the zero-sum case meaning $U_L(\pi_L, \pi_F) = -U_F(\pi_L, \pi_F)$.

Example evaluation. For $\pi_L = (0.5, 0.3, 0.2)$,

$$U_F(\pi_L, e_A) = 12(1 - 0.5) = 6.0, \quad U_F(\pi_L, e_B) = 8(1 - 0.3) = 5.6, \quad U_F(\pi_L, e_C) = 4(1 - 0.2) = 3.2,$$

so $BR(\pi_L) = e_A$ and the expected Leader payoff is $U_L = -6.0$.

Stackelberg solution. In the presented example the Leader solves

$$\min_{\pi_L \in \Pi_L} \max_{i \in \{A,B,C\}} U_F(\pi_L, e_i).$$

The optimum equalizes the two highest terms with $\pi_L(C) = 0$:

$$12(1 - \pi_L(A)) = 8(1 - \pi_L(B)), \quad \pi_L(A) + \pi_L(B) = 1,$$

yielding

$$\pi_L^\star(A) = 0.6, \qquad \pi_L^\star(B) = 0.4, \qquad \pi_L^\star(C) = 0.$$

Therefore the Follower payoff is

$$U_F\big(\pi_L^\star, e_A\big) = U_F\big(\pi_L^\star, e_B\big) = 4.8 \quad (> U_F(\pi_L^\star, e_C) = 4).$$

and the Leader's expected payoff is $U_L(\pi_L^\star, BR(\pi_L^\star)) = -4.8$.

This illustrates the roles of Σ_F (pure responses), Π_L (Leader commitments), $BR(\pi_L)$ (Follower best response), and U_L, U_F (players' payoffs) in equations (1)–(2).

Note that this pedagogical example is intentionally simplified (single-step ($m = 1$), zero-sum, and with only a few pure strategies) to make the concepts clear. In practice, and in our benchmarks, SSGs typically involve multiple time steps, larger action spaces, potentially non-zero-sum payoffs, and more sophisticated Follower models.

3 Related Work

The existing methods for solving 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 [Bošanský and Čermak 2015] extends the DOBBS algorithm [Paruchuri et al. 2008], originally designed for one-step games, to handle extensive-form games [Conitzer and Sandholm 2006] 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 [Čermák et al. 2016] 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 [Karwowski and Mańdziuk 2019b, 2020] uses Upper Confidence Bounds applied to Trees (UCT) [Kocsis and Szepesvári 2006], a Monte Carlo Tree Search-based approach [Świechowski et al. 2023], 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) [Żychowski and Mańdziuk 2020; Żychowski and Mańdziuk 2021a] 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 [Żychowski and Mańdziuk 2023] 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 Other Methods

The approaches described above are general-purpose solvers that can be applied to many different types of SSGs, just like the method proposed in this paper. The shared generality is the main reason we compare our approach against them. However, the literature also contains methods tailored to more specific game structures or modeling assumptions.

One example is *Signaling Games*, where the Leader commits to both a mixed strategy and an information disclosure scheme influencing the Follower's beliefs [Bondi et al. 2020; Żychowski et al. 2022]. The optimization includes an *information design* stage to choose what partial information to reveal, aiming to steer the Follower's behavior. Solvers exploit the signaling policy space and belief-update structure, which differ from standard SSGs.

Another specialized class of methods incorporates *bounded rationality* of the Follower. Instead of assuming a perfectly rational best response, the Follower is modeled using parametric or learned behavioral models such as the Prospect Theory [Yang et al. 2013], Quantal Response [Nguyen et al. 2013], or Anchoring Theory [Pita et al. 2011]. Such methods (e.g. [Karwowski et al. 2023; Perrault et al. 2020; Żychowski and Mańdziuk 2021b]) optimize Leader strategies against an empirically estimated stochastic response, often fitted from historical data or human-subject experiments.

Further domain-specific approaches address settings with *spatio-temporal patrolling* and route constraints [Jain et al. 2010a; Ordónez et al. 2012], *wildlife protection* [Wang et al. 2019], or *cybersecurity control games* [Durkota et al. 2015]. Algorithms in these niches often leverage problem-specific combinatorial structures (e.g., paths, flows, network connectivity).

Although these specialized techniques can be highly effective in their target domains, they are generally less transferable to other SSG formulations. By contrast, the general-purpose methods discussed in the previous subsections, as well as our proposed approach, maintain applicability across a broad range of SSGs, making them natural baselines for comparative evaluation.

4 Limitations of Existing Methods

4.1 Sparsity

Many existing methods do not inherently promote sparsity and often return mixed strategies containing a large number of pure strategies with very low probabilities, which contribute negligibly to the final payoff and unnecessarily inflate solution complexity. In many real-world SSG deployments, the Leader's final mixed strategy must be *sparse*, i.e., composed of a small subset of pure strategies with non-zero probabilities. Sparse strategies offer several practical

benefits such as *interpretability* - decision makers can more easily understand the rationale behind a compact set of actions, facilitating validation and trust in the recommended plan; *ease of implementation* - operational units such as patrol officers or cybersecurity analysts need follow and remember fewer patterns, reducing the risk of implementation errors; and *maintainability* - small sets of active strategies simplify updates in response to changing conditions, making the Leader's policy more adaptable in practice. From a computational perspective, sparsity also reduces the evaluation burden in equilibrium computation: strategies with zero probability need not be considered when computing payoffs or best responses, which is critical in large-scale SSG instances where the number of pure strategies may grow exponentially.

Despite these clear benefits, standard sparsity-promoting techniques widely used in machine learning, such as L_1-regularization, face significant hurdles when applied to SSGs. First, the optimization in SSGs is strictly constrained to a probability simplex (strategies must sum to 1), where standard soft-thresholding operators can easily violate normalization constraints. Second, many regularized approaches rely on the availability of gradients, which are often computationally expensive or impossible to estimate in complex game settings involving discrete spaces or bounded rationality. Consequently, general-purpose solvers typically struggle to enforce sparsity during the optimization phase, necessitating ad-hoc post-processing.

In EASG, for example, sparsity is enforced indirectly through a dedicated *crossover-based pruning* step that probabilistically removes pure strategies from offspring based on their selection probabilities. However, this procedure is inherently *randomized*, meaning that the elimination of low-probability strategies is not always aligned with the globally optimal subset to retain. As a result, the final strategy quality may depend on stochastic pruning events rather than a deliberate, globally guided selection. Moreover, because sparsity in EASG emerges only gradually through the evolutionary process, additional generations are often required before a compact, high-quality subset of pure strategies is discovered, increasing computational cost and introducing variability in convergence speed. *The method we propose overcomes these issues by embedding sparsity directly into its representation from the outset via binary activation variables, ensuring that every candidate solution is inherently sparse and allowing the optimization process to focus immediately on refining both the selected subset of strategies and their probabilities.*

4.2 Solution Encoding and Operators

A significant limitation of existing evolutionary approaches such as EASG and CoEvoSG lies in their reliance on complex, problem-specific solution encodings and specialized evolutionary operators (e.g., tailored mutation and crossover schemes). These operators are intricately designed to exploit the structural properties of particular SSG variants, such as the movement dynamics in Warehouse Games or the trace mechanics in Search Games. While effective within their intended scope, these designs severely limit the portability of the algorithms: adapting them to a different game type (e.g., adapting EASG to Signaling Games in [Żychowski et al. 2022]) required substantial reengineering of the encoding scheme, modification of operators, and careful retuning of algorithmic parameters. This tight coupling to problem-specific representations reduces the generality of the methods and increases the engineering effort required for new domains. *The solution proposed in this paper addresses this limitation by using a generic binary-real dual representation that is independent of game-specific rules, allowing it to be directly applied to different SSG variants and even to other classes of games without redesigning operators or encodings.*

4.3 Theoretical Generality

Another consequence of this dependence on handcrafted encodings and operators is the difficulty of establishing a general theoretical foundation for these methods. Because the search dynamics are partially dictated by ad hoc design choices embedded in the operators, it becomes challenging to provide rigorous explanations for their observed empirical performance. As acknowledged in [Żychowski and Mańdziuk 2021a], EA-based methods for SSGs remain relatively weak in their theoretical grounding. *In contrast, we propose a solution which builds on the information-geometric optimization (IGO) framework [Ollivier et al. 2017], employing probabilistic model-based evolutionary search with well-defined update equations for both binary and real-valued variables, thus providing a mathematical basis for the optimization process.*

5 Sparse Evolution by ADSO

This section presents our new approach to mixed-strategy optimization by searching a binary-real Augmented Decision Space (ADS) with a principled evolutionary algorithm. The method leverages the probabilistic modeling of candidate solutions [Ceberio et al. 2024], where the model defines a distribution over the set Π_L of the Leader's mixed strategies. A mixed strategy, in turn, gives a probability distribution over the set Σ_L of pure strategies. The ADSO framework thus defines and optimizes a distribution over probability distributions.

Let x_i be a decision variable that assigns a probability of enactment 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 turning on ($\hat{x}_i = 1$) or turning off ($\hat{x}_i = 0$) of the i-th pure strategy in the mixture. Thus the value of $\tilde{x}_i$ is interpreted as the probability of enactment when $\hat{x}_i = 1$. By reformulating the decision space in this manner, the number of decision variables is 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 sampling of the evolving population, each binary variable $\hat{x}_i$ is independently drawn from the Bernoulli distribution:

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

where q_i represents the probability that $\hat{x}_i = 1$. The distribution over binary vectors is then given as $P(\hat{\mathbf{x}}) = P_{q_1}(\hat{x}_1) \times P_{q_2}(\hat{x}_2) \times \ldots \times P_{q_n}(\hat{x}_n)$. 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, $p(\tilde{\mathbf{x}}) = \mathcal{N}(\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}})$, parameterized by mean $\boldsymbol{\mu}_{\tilde{\mathbf{x}}}$ and covariance $\Sigma_{\tilde{\mathbf{x}}}$. The overall probability distribution underlying the population of candidate solutions in the ADS becomes:

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

which is parameterized by $\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}, \mathbf{q}$, where $\mathbf{q} = [q_1, q_2, \ldots, q_n]$.

Let $f(\mathbf{x'})$ be the payoff of the Leader given the mixed strategy encoded by decision vector $\mathbf{x'}$. This payoff is contingent on the Follower's response, i.e., $f(\mathbf{x'}) = U_L(\bar{\pi}_L(\hat{\mathbf{x}}, \tilde{\mathbf{x}}), \mathrm{BR}(\bar{\pi}_L))$, where $\bar{\pi}_L(\hat{\mathbf{x}}, \tilde{\mathbf{x}}) = \frac{1}{\sum \hat{x}_i \cdot \tilde{x}_i}[\hat{x}_1 \cdot \tilde{x}_1, \hat{x}_2 \cdot \tilde{x}_2, \ldots, \hat{x}_n \cdot \tilde{x}_n]$. Following the IGO method - which turns any smooth parametric family of probability distributions into an evolutionary search algorithm [Ollivier et al. 2017] - the Leader's mathematical program can be recast as maximizing the expected

payoff, $F = \mathbb{E}[f(\mathbf{x}')]$, with respect to the parameterized search distribution as:

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

Note that this reformulation does not alter the global optimum of the original optimization problem in (1). If a certain decision vector $(\hat{\mathbf{x}}^\star, \tilde{\mathbf{x}}^\star)$ maximizes the Leader's payoff $U_L(\bar{\pi}_L, \mathrm{BR}(\bar{\pi}_L))$, then a converged population whose distribution has collapsed to a delta function centered at that point, i.e., $p(\mathbf{x}')$ is zero everywhere but at $(\hat{\mathbf{x}}^\star, \tilde{\mathbf{x}}^\star)$, maximizes (3).

Applying the log-likelihood trick, the gradient of F with respect to the parameters of the binary distribution can be derived as:

$$\nabla_{\mathbf{q}} F = \sum_{\forall \hat{\mathbf{x}}} \Big(\int f(\mathbf{x}') \cdot p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \Big) \nabla_{\mathbf{q}} P(\hat{\mathbf{x}}) \tag{4}$$

$$= \sum_{\forall \hat{\mathbf{x}}} \Big(\int f(\mathbf{x}') \cdot p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \Big) \frac{\nabla_{\mathbf{q}} P(\hat{\mathbf{x}})}{P(\hat{\mathbf{x}})} P(\hat{\mathbf{x}}) \tag{5}$$

$$= \sum_{\forall \hat{\mathbf{x}}} \Big(\int f(\mathbf{x}') \cdot p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \Big) \nabla_{\mathbf{q}} (\log(P(\hat{\mathbf{x}}))\, P(\hat{\mathbf{x}}) \tag{6}$$

$$= \sum_{\forall \hat{\mathbf{x}}} \Big(\int f(\mathbf{x}') \cdot p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \Big) \nabla_{\mathbf{q}} \Big(\sum_{i=1}^{n} (\hat{x}_i \log(q_i) + (1 - \hat{x}_i) \log(1 - q_i)) \Big) P(\hat{\mathbf{x}}) \tag{7}$$

Similarly, combining the log-likelihood trick with the Leibniz integral rule for differentiation under the integral sign gives the gradient of F with respect to the normal distribution parameters as:

$$\nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}} F = \sum_{\forall \hat{\mathbf{x}}} \Big(\int f(\mathbf{x}') \cdot \nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}} (\log(\mathcal{N}(\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}))) \cdot p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}} \Big) P(\hat{\mathbf{x}}). \tag{8}$$

5.1 Evolutionary Update Equations

Prior to presenting the mathematical derivation of the evolutionary updates, it is instructive to take a high-level view of the optimization mechanics of ADSO. The framework treats the mixed-strategy optimization as a dual probabilistic search process, maintaining two distinct search distribution models—the binary sparsification model $P(\hat{\mathbf{x}})$ and the multivariate normal distribution $p(\tilde{\mathbf{x}})$ over real-valued enactment probabilities of selected pure strategies—that evolve *cooperatively* to navigate the augmented decision space. To combat the potential curse of dimensionality, ADSO employs an alternating maximization scheme. Its two alternating phases mimic a cooperative coevolutionary process. In one phase, the algorithm fixes the real-valued parameters $\tilde{\mathbf{x}}$, given by the mean $\boldsymbol{\mu}_{\tilde{\mathbf{x}}}$ of the multivariate normal $p(\tilde{\mathbf{x}})$, and updates the model $P(\hat{\mathbf{x}})$ using a population-based stochastic gradient ascent rule derived via the IGO framework. It shall be shown that this update rule coincides with that of the well-known *population-based incremental learning* (PBIL) algorithm [Baluja 1994]. In the other phase, the selection of pure strategies is fixed to the best binary vector found, and the search distribution over enactment probabilities is updated using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Hansen 2016; Hansen and Ostermeier 2001]. This decomposition and iterated switching mechanism enable efficient exploration of the sparse strategy space, without the computational load of a fully coupled joint optimization of $\hat{\mathbf{x}}$ and $\tilde{\mathbf{x}}$.

5.1.1 Stochastic gradient approximations. An evolution strategy integrating the alternating phases of ADSO can be crafted by sampling and evaluating 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 arrive at Monte Carlo estimates of the gradients in (7) and (8). These stochastic gradient approximations could then be used in lieu of the true

gradient to iteratively update the search distribution models towards regions of ADS containing performant mixed strategies for the Leader. To this end, recall that ADSO maintains two distribution models:

- 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 $q_1, ..., q_n$ for the binary vector $\hat{\mathbf{x}}$.

A complete 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 payoff against the Follower's best response.

Monte Carlo estimates of (7) and (8) can, however, depict high variance due to the dimensionality of the ADS. To combat this curse of dimensionality, we make certain simplifying assumptions while deriving the update equations of ADSO. Specifically, while approximating (7), 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 (8), 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 $\hat{\mathbf{x}}^{gen\_best}$. This reduces (8) to:

$$\nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}} F \approx \int f(\hat{\mathbf{x}}^{gen\_best}, \tilde{\mathbf{x}}) \cdot \nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}} (\log(\mathcal{N}(\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}}))) \cdot p(\tilde{\mathbf{x}})\, d\tilde{\mathbf{x}}. \tag{9}$$

ADSO thus alternates between updating the normal distribution using (9) and the binary distribution by simplifying (7) to:

$$\nabla_{\mathbf{q}} F \approx \sum_{\forall \hat{\mathbf{x}}} f(\hat{\mathbf{x}}, \boldsymbol{\mu}_{\tilde{\mathbf{x}}})\, \nabla_{\mathbf{q}} \Big(\sum_{i=1}^{n} (\hat{x}_i \log(q_i) + (1 - \hat{x}_i) \log(1 - q_i)) \Big) P(\hat{\mathbf{x}}) \tag{10}$$

5.1.2 Binary distribution updates. In the gradient on the right of (10), 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 rule from generation t to $t + 1$ of our evolutionary algorithm:

$$q_i^{(t+1)} \leftarrow q_i^{(t)} + \eta_i \nabla_{q_i^{(t)}} F \approx q_i^{(t)} + \frac{\eta_i}{N} \sum_{j=1}^{N} f(\hat{\mathbf{x}}^j, \boldsymbol{\mu}_{\tilde{\mathbf{x}}}) \frac{\hat{x}_i^j - q_i^{(t)}}{q_i^{(t)} (1 - q_i^{(t)})}, \tag{11}$$

where η_i is the learning rate. The fittest binary vector in the population is cached as $\hat{\mathbf{x}}^{gen\_best}$.

One of the key considerations in ensuring the stability of gradient-based updates is having small step sizes; i.e., the distance between $\mathbf{q}^{(t+1)}$ and $\mathbf{q}^{(t)}$ should be bounded from above. In this regard, notice that as q_i converges to either 0 or 1, the update in (11) blows up as its denominator tends to 0. In order to avoid the resulting instability, we propose to substitute $\frac{\eta_i}{N}$ by $\eta \times q_i(1 - q_i)$, for some constant η, which modifies the update equation to:

$$q_i^{(t+1)} \leftarrow q_i^{(t)} + \eta \sum_{j=1}^{N} f(\hat{\mathbf{x}}^j, \boldsymbol{\mu}_{\tilde{\mathbf{x}}}) (\hat{x}_i^j - q_i^{(t)}). \tag{12}$$

The same rule can also be derived by strictly following the steps of the IGO method which uses information-theoretic arguments to constrain the *distance* between the pair of binary distributions represented by $\mathbf{q}^{(t+1)}$ and $\mathbf{q}^{(t)}$. This distance is commonly defined using the *Kullback-Leibler divergence* [Kullback 1951], which leads to the inverse of the Fisher matrix for rescaling the gradient. We leave out the complete IGO derivation as we find the exploding-gradient argument to provide sufficient intuition for the adaptation in (12).

Yet another consideration in IGO is to ensure the invariance of the updates under all order-preserving transformations of the fitness function. In other words, the behavior of the algorithm should not change drastically under monotone transformations of f. To this end, we apply rank-based fitness shaping by replacing the true objective function f with a non-negative utility value u. This further adapts the update of the binary distribution to its final form:

$$q_i^{(t+1)} \leftarrow q_i^{(t)} + \eta \sum_{j=1}^{N} u(\hat{\mathbf{x}}^j, \boldsymbol{\mu}_{\tilde{\mathbf{x}}})(\hat{x}_i^j - q_i^{(t)}). \tag{13}$$

The obtained algorithm coincides with different versions of the so-called PBIL algorithm [Baluja 1994] depending on the choice of utility values. The original form of PBIL is recovered by setting $u(\hat{\mathbf{x}}^{gen\_best}, \boldsymbol{\mu}_{\tilde{\mathbf{x}}})$ to 1 and all other $u(\hat{\mathbf{x}}^j, \boldsymbol{\mu}_{\tilde{\mathbf{x}}})$ to 0. This equivalence is confirmed by Proposition 14 in [Ollivier et al. 2017].

5.1.3 Real-valued distribution updates. The stochastic gradient update of normal distributions governed by Monte Carlo estimates of (9) is at the core of a family of natural evolution strategies [Wierstra et al. 2014]. In our implementation, we directly adopt the CMA-ES [Hansen 2016], a prominent member of this family, for real-valued distribution updates. The connection between (9) and the CMA-ES update equations is established by Proposition 15 of [Ollivier et al. 2017]. Algorithm 1 outlines the alternating binary-real ADSO procedure that switches between binary distribution updates via (13) and the CMA-ES.

5.2 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. The discussion below demonstrates how Danskin's theorem, often used in the context of minimax problems [Danskin 1966], can help alleviate this bottleneck in the case of zero-sum SSGs.

We explicitly emphasize that this acceleration relies on the properties of minimax functions and is therefore strictly applied to zero-sum game formulations only; for general-sum games, the standard evaluation procedure must be retained.

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

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

Recall from Section 2 that the Follower's mathematical program, $\arg\max_{\bar{\pi}_F \in \Pi_F} U_F(\bar{\pi}_L, \bar{\pi}_F)$, may have multiple optimal responses, with ties broken in favor of the Leader under the SSE definition. In this regard, we rephrase the proven statement C.2 from [Madry et al. 2018] in the context of zero-sum SSGs.

Theorem 1. *Let $BR(\bar{\pi}_L)$ be such that $BR(\bar{\pi}_L) \in \Pi_F$ and is a maximizer of $\max_{\bar{\pi}_F \in \Pi_F} U_F(\bar{\pi}_L, \bar{\pi}_F)$. Then, as long as it is non-zero, $-\nabla_{\tilde{\mathbf{x}}} U_F(\bar{\pi}_L(\hat{\mathbf{x}}^{gen\_best}, \tilde{\mathbf{x}}), BR(\bar{\pi}_L))$ is a descent direction for the maximum function $\max_{\bar{\pi}_F \in \Pi_F} U_F(\bar{\pi}_L(\hat{\mathbf{x}}^{gen\_best}, \tilde{\mathbf{x}}), \bar{\pi}_F)$.*

The statement follows as a corollary to Danskin's result about the property of directional derivatives of the maximum function [Danskin 1966], and implies that $-\nabla_{\tilde{\mathbf{x}}} U_F(\bar{\pi}_L(\hat{\mathbf{x}}^{gen\_best}, \tilde{\mathbf{x}}), \mathrm{BR}(\bar{\pi}_L))$ is an ascent direction for $f(\hat{\mathbf{x}}^{gen\_best}, \tilde{\mathbf{x}}) = -\max_{\bar{\pi}_F \in \Pi_F} U_F(\bar{\pi}_L(\hat{\mathbf{x}}^{gen\_best}, \tilde{\mathbf{x}}), \bar{\pi}_F)$. Moreover, in cases where the set of Follower's optimal responses is a singleton, $-\nabla_{\tilde{\mathbf{x}}} U_F(\bar{\pi}_L(\hat{\mathbf{x}}^{gen\_best}, \tilde{\mathbf{x}}), \mathrm{BR}(\bar{\pi}_L))$ is the *steepest* ascent direction for $f(\hat{\mathbf{x}}^{gen\_best}, \tilde{\mathbf{x}})$.

The intuition behind this result is that since the maximum function and $U_F(\bar{\pi}_L(\hat{\mathbf{x}}^{gen\_best}, \tilde{\mathbf{x}}), \mathrm{BR}(\bar{\pi}_L))$ are locally the same, their gradients will also be the same. In order to leverage this insight, we first note that in the limit of a normal

Algorithm 1: ADSO: Augmented Decision Space Optimization for SSGs

Input: Leader payoff oracle (U_L); Follower payoff oracle for computing best response (BR); population size (n)
Output: Mixed Leader's strategy

Init: Set $q_i \leftarrow q_0$, $\boldsymbol{\mu}_{\tilde{\mathbf{x}}} \leftarrow \mu_0$, $\Sigma_{\tilde{\mathbf{x}}} \leftarrow \Sigma_0$, $U_L^{\text{best}} \leftarrow -\infty$, $\hat{\mathbf{x}}^{best} \leftarrow \emptyset$, $\tilde{\mathbf{x}}^{best} \leftarrow \emptyset$
while *termination criterion not met* **do**
  `// Binary distribution update phase`
  for $j = 1$ *to* N **do**
    Sample binary vector $\hat{\mathbf{x}}^{(j)} \sim \text{Bernoulli}(\mathbf{q})$
    Evaluate $U_L^{(j)} := U_L(\bar{\pi}_L(\hat{\mathbf{x}}^{(j)}, \boldsymbol{\mu}_{\tilde{\mathbf{x}}}), \text{BR}(\bar{\pi}_L))$
  end
  Compute utilities u_j from $\{U_L^{(j)}\}$ (rank-based fitness shaping)
  $\mathbf{q} \leftarrow \mathbf{q} + \eta \sum_{j=1}^{N} u_j \cdot \left(\hat{\mathbf{x}}^{(j)} - \mathbf{q}\right)$
  $\hat{\mathbf{x}}^{gen\_best} \leftarrow \arg\max_j U_L^{(j)}$
  `// Real-valued distribution update phase`
  for $j = 1$ *to* N **do**
    Sample real-value vector $\tilde{\mathbf{x}}^{(j)} \sim \mathcal{N}(\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}})$
    Evaluate $U_L^{(j)} := U_L(\bar{\pi}_L(\hat{\mathbf{x}}^{gen\_best}, \tilde{\mathbf{x}}^{(j)}), \text{BR}(\bar{\pi}_L))$
  end
  Update $(\boldsymbol{\mu}_{\tilde{\mathbf{x}}}, \Sigma_{\tilde{\mathbf{x}}})$ using $\{U_L^{(j)}\}$ based on CMA-ES update equations
  `// Update global best for reporting`
  if $\max_j U_L^{(j)} > U_L^{best}$ **then**
    $U_L^{\text{best}} \leftarrow \max_j U_L^{(j)}$
    $(\hat{\mathbf{x}}^{best}, \tilde{\mathbf{x}}^{best}) \leftarrow \arg\max_j U_L^{(j)}$
  end
end
return $(\hat{\boldsymbol{x}}^{best}, \tilde{\boldsymbol{x}}^{best})$

search distribution $p(\tilde{\mathbf{x}})$ with small variance, (9) simplifies to:

$$\lim_{\Sigma_{\tilde{\mathbf{x}}} \to 0} \nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}}} F = \nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}}} f(\hat{\mathbf{x}}^{gen\_best}, \boldsymbol{\mu}_{\tilde{\mathbf{x}}}), \tag{14}$$

which, according to Danskin, either coincides with or forms an acute angle with the vector $\nabla_{\boldsymbol{\mu}_{\tilde{\mathbf{x}}}} U_F(\bar{\pi}_L(\hat{\mathbf{x}}^{gen\_best}, \mu_{\tilde{\mathbf{x}}}), \text{BR}(\bar{\pi}_L))$. Consequently, for the CMA-ES updates that correspond to (9), it suffices to compute the Follower's best response only once per generation with respect to the distribution mean, i.e., Leader's strategy $\bar{\pi}_L(\hat{\mathbf{x}}^{gen\_best}, \mu_{\tilde{\mathbf{x}}})$. Optimal responses need not be recomputed for every population member sampled from the normal distribution, under the assumption that the distribution is sufficiently narrow. This reduces the computational cost per generation by a factor of N given population size N. We shall demonstrate in Section 7.5 that computational savings are consistently achieved in practice across all our experiments, even though the narrowness assumption of the normal distribution may not be guaranteed.

6 Experimental Setup

6.1 Benchmark Games

6.1.1 Zero-sum Warehouse Games. (WHGs), as introduced in [Karwowski and Mańdziuk 2019a], 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 an example of a WHG scenario.

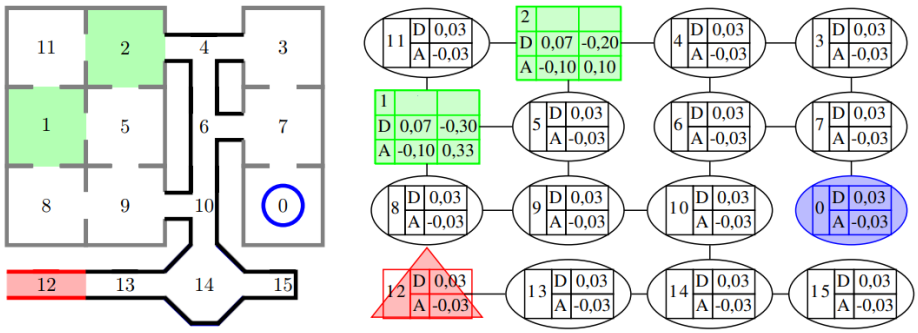

Fig. 1. Example WHG scenario [Karwowski and Mańdziuk 2019a]: the warehouse layout (left) and its corresponding graph representation (right) depict the payoffs associated with the players (D - Defender/Leader, A - Attacker/Follower) for the respective game outcomes. Green rectangular vertices represent targets, while the red triangular vertex and blue circular vertex denote the starting positions of the Follower and Leader, respectively.

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 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 [Watts and Strogatz 1998] with an average vertex degree $d_{\text{avg}} = 3$.

6.1.2 Search Games. (SEGs), introduced in [Bošanský and Čermak 2015], 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.

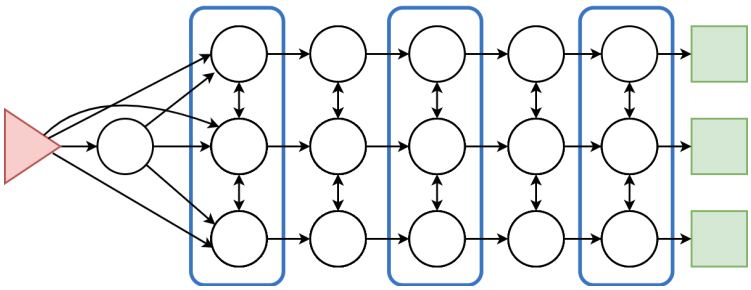

Fig. 2. Example of Search Game. A red triangle denotes the Follower's starting vertex, green rectangles are targets. Three rounded groups of vertices represent restricted subsets of nodes within which the Leader's units can freely move.

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 [Żychowski and Mańdziuk 2021a] 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.

6.1.3 FlipIt Games. (FIGs), as introduced in [Van Dijk et al. 2013], 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.

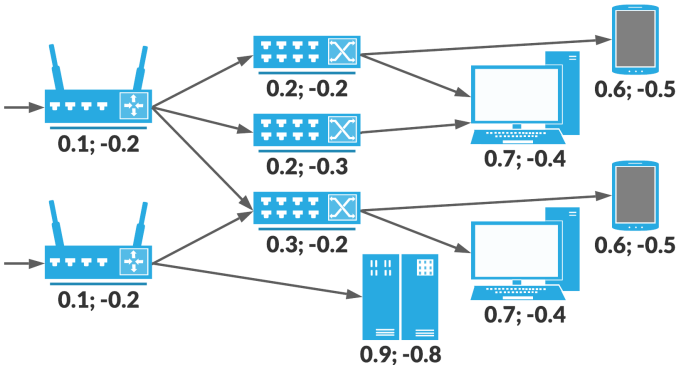

Fig. 3. Example FIG scenario [Żychowski and Mańdziuk 2023] with two entry nodes (routers) on the left. Numbers below each component denote a reward for controlling the node (left) and a cost of a flip attempt (right).

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 [Watts and Strogatz 1998] with an average vertex degree $d_{\text{avg}} = 3$.

The experiments were conducted under the *No-Info* variant [Černỳ et al. 2018], 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.

6.2 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#. The source code is publicly accessible at github.com/zychowskia/ADSO. 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. To facilitate reproducibility and provide a clear overview of the experimental configuration, the specific parameter setting employed for the ADSO framework is presented in Table 1.

Table 1. Summary of ADSO parameter settings.

Parameter	Value
Population Size	200
Max. Fitness Evaluations	10^5
Stagnation Limit (Generations)	20
Learning Rate (η)	10^{-4}
Optimizer (Real-valued part)	CMA-ES
Fitness Shaping	Rank-based

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} 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.

7 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. To ensure full reproducibility of the experiments, a fixed sequence of random seeds was used to initialize the pseudo-random number generators across all independent runs for all evaluated algorithms.

7.1 Payoffs

Tables 2, 3, and 4 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.

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. The proposed ADSO method

Table 2. 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**. Values are presented as mean ± std dev.

n	C2016	O2UCT	EASG	CoEvoSG	CMA-ES	ADSO
15	**0.052** ± 0.000	0.051 ± 0.004	0.051 ± 0.004	0.050 ± 0.005	0.049 ± 0.006	0.051 ± 0.003
20	**0.054** ± 0.000	0.053 ± 0.004	0.052 ± 0.004	0.051 ± 0.005	0.050 ± 0.006	0.053 ± 0.004
25	**0.048** ± 0.000	0.046 ± 0.005	0.045 ± 0.005	0.044 ± 0.006	0.044 ± 0.007	0.047 ± 0.004
30	-	0.044 ± 0.005	0.042 ± 0.006	0.040 ± 0.006	0.040 ± 0.008	**0.045** ± 0.005
40	-	-	0.040 ± 0.007	0.038 ± 0.007	0.038 ± 0.009	**0.041** ± 0.006
m	C2016	O2UCT	EASG	CoEvoSG	CMA-ES	ADSO
3	**0.043** ± 0.000	**0.043** ± 0.001	**0.043** ± 0.001	**0.043** ± 0.001	**0.043** ± 0.002	**0.043** ± 0.001
4	**0.052** ± 0.000	0.050 ± 0.003	0.050 ± 0.003	0.049 ± 0.004	0.048 ± 0.005	0.051 ± 0.003
5	**0.055** ± 0.000	0.054 ± 0.004	0.053 ± 0.004	0.052 ± 0.005	0.051 ± 0.006	0.054 ± 0.004
6	**0.058** ± 0.000	0.056 ± 0.005	0.054 ± 0.005	0.052 ± 0.006	0.052 ± 0.007	0.055 ± 0.005
8	-	**0.053** ± 0.006	0.051 ± 0.006	0.049 ± 0.007	0.048 ± 0.008	0.052 ± 0.005
10	-	-	**0.048** ± 0.007	0.046 ± 0.008	0.046 ± 0.009	**0.048** ± 0.006

Table 3. 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**. Values are presented as mean ± std dev.

n	C2016	O2UCT	EASG	CoEvoSG	CMA-ES	ADSO
15	**0.122** ± 0.000	0.116 ± 0.009	0.115 ± 0.008	0.115 ± 0.010	0.114 ± 0.009	0.119 ± 0.008
20	**0.117** ± 0.000	0.112 ± 0.009	0.106 ± 0.008	0.104 ± 0.011	0.104 ± 0.010	0.114 ± 0.009
25	-	0.123 ± 0.010	0.117 ± 0.009	0.116 ± 0.012	0.115 ± 0.011	**0.124** ± 0.009
30	-	-	0.136 ± 0.011	0.135 ± 0.014	0.134 ± 0.013	**0.137** ± 0.010
40	-	-	-	0.152 ± 0.016	0.151 ± 0.015	**0.154** ± 0.012
m	C2016	O2UCT	EASG	CoEvoSG	CMA-ES	ADSO
3	**0.137** ± 0.000	0.126 ± 0.007	0.118 ± 0.006	0.118 ± 0.008	0.117 ± 0.007	0.128 ± 0.006
4	**0.124** ± 0.000	0.113 ± 0.008	0.110 ± 0.007	0.109 ± 0.009	0.108 ± 0.008	0.117 ± 0.007
5	**0.106** ± 0.000	0.093 ± 0.009	0.090 ± 0.008	0.087 ± 0.010	0.087 ± 0.009	0.101 ± 0.008
6	-	0.129 ± 0.010	0.123 ± 0.009	0.123 ± 0.012	0.123 ± 0.011	**0.134** ± 0.010
8	-	-	0.112 ± 0.011	0.111 ± 0.013	0.110 ± 0.012	**0.117** ± 0.011
10	-	-	-	0.144 ± 0.015	0.144 ± 0.014	**0.149** ± 0.012

addresses this issue by incorporating an augmented binary component, which improves scalability by promoting sparsity in the resulting mixed strategies. More detailed results and discussion about the sparsity of returned results can be found in Section 7.3.

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

Table 4. 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**. Values are presented as mean ± std dev.

n	C2016	O2UCT	EASG	CoEvoSG	CMA-ES	ADSO
5	**0.890** ± 0.000	0.887 ± 0.052	0.886 ± 0.058	0.886 ± 0.062	0.880 ± 0.055	0.889 ± 0.051
10	**0.854** ± 0.000	0.851 ± 0.055	0.847 ± 0.063	0.846 ± 0.068	0.839 ± 0.059	0.852 ± 0.055
15	**0.811** ± 0.000	0.807 ± 0.059	0.802 ± 0.068	0.800 ± 0.073	0.795 ± 0.063	0.809 ± 0.058
20	-	0.784 ± 0.062	0.780 ± 0.071	0.775 ± 0.076	0.774 ± 0.067	**0.786** ± 0.061
25	-	-	-	0.748 ± 0.079	0.745 ± 0.070	**0.757** ± 0.064
m	C2016	O2UCT	EASG	CoEvoSG	CMA-ES	ADSO
3	**0.823** ± 0.000	0.821 ± 0.053	0.820 ± 0.060	0.817 ± 0.065	0.816 ± 0.056	0.822 ± 0.053
4	**0.817** ± 0.000	0.812 ± 0.056	0.808 ± 0.064	0.805 ± 0.069	0.802 ± 0.059	0.814 ± 0.055
5	**0.810** ± 0.000	0.801 ± 0.058	0.798 ± 0.067	0.791 ± 0.072	0.787 ± 0.062	0.806 ± 0.058
6	-	0.792 ± 0.060	0.792 ± 0.070	0.791 ± 0.075	0.789 ± 0.065	**0.800** ± 0.060
8	-	0.785 ± 0.063	0.784 ± 0.072	0.781 ± 0.078	0.776 ± 0.068	**0.788** ± 0.062
10	-	-	0.780 ± 0.075	0.778 ± 0.080	0.777 ± 0.070	**0.780** ± 0.065

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 5 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 5. 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%)**

7.2 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

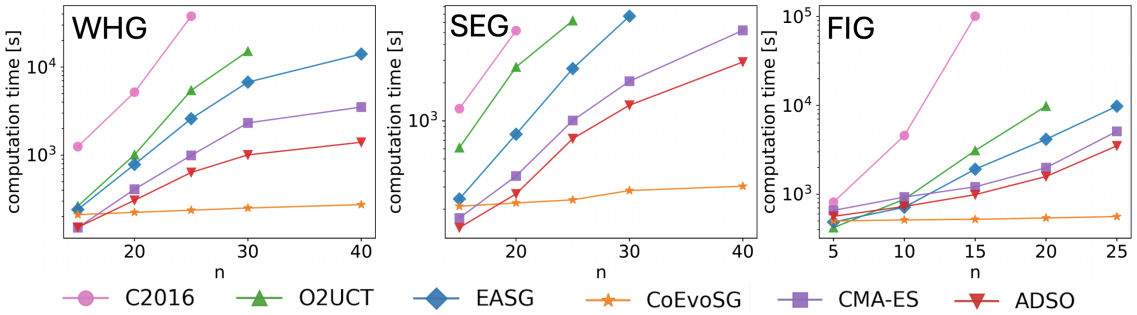

Fig. 4. Scalability of tested methods: computation time (logarithmic scale) with respect to the number of graph nodes (n).

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.

As detailed in Section 7.5, we can only use Danskin's theorem to speed up zero-sum WHGs. We cannot apply it to general-sum SEGs and FIGs. Nevertheless, ADSO overcomes this limitation by producing sparse strategies that need fewer evaluations. As Figure 4 indicates, this built-in efficiency keeps the algorithm highly scalable across all tested games.

7.3 Sparsity of the Leader's Strategy

Besides payoff quality and scalability, another critical property of the compared methods is the sparsity of the resulting Leader's mixed strategy. Sparsity is measured as the number of pure strategies with non-zero probability in the final solution. Sparse solutions are particularly desirable in practical deployments, since they are easier to interpret, communicate, and implement in operational settings.

Table 6 presents the average number of pure strategies (together with the corresponding standard deviations) for all tested methods across the three benchmark families. The results clearly indicate that ADSO consistently produces compact strategies of similar sparsity to exact methods, while substantially outperforming CMA-ES, which tends to maintain a large number of nearly redundant pure strategies. Compared to EASG and CoEvoSG, which enforce sparsity only indirectly via crossover-based pruning, ADSO achieves a more stable and direct sparsification effect through its binary activation variables.

Table 6. Average number of pure strategies (sparsity) in the final Leader's strategy with standard deviation (mean ± std dev.). Lower values indicate sparser strategies. The best results (excluding C2016) are bolded.

Game	C2016	O2UCT	EASG	CoEvoSG	CMA-ES	ADSO
WHG	5.74 ± 0.7	12.8 ± 3.5	7.83 ± 1.5	8.12 ± 1.7	32.73 ± 6.4	**6.76** ± 1.4
SEG	5.91 ± 0.9	13.5 ± 3.8	8.01 ± 1.6	8.44 ± 1.9	34.12 ± 6.8	**6.93** ± 1.5
FIG	5.48 ± 0.6	11.9 ± 3.1	7.65 ± 1.4	7.80 ± 1.6	31.34 ± 6.0	**6.59** ± 1.3
Average	5.71	12.7	7.83	8.12	32.73	**6.76**

Table 7. 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**

7.4 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 7 presents the mean and maximum standard deviations for all tested methods.

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.

7.5 Effectiveness of Danskin's Theorem in Zero-Sum Games

In Section 5.2, we described how a corollary to Danskin's theorem could be leveraged to reduce computational costs in solving zero-sum SSGs with ADSO. The theoretical conclusions drawn therein were for the case of CMA-ES updates of a "narrow" normal distribution, with such narrowness being difficult to guarantee in practice. Therefore, to assess the practical impact of the proposed algorithmic innovation, we conducted an ablation study on zero-sum WHGs evaluating ADSO with and without the use of Danskin's theorem. Recall the implication of the theorem that the Follower's best

Table 8. ADSO ablation in zero-sum WHG across graph sizes n in terms of computation time and Leader's payoff (mean ± std dev.).

n	Computation Time [s]		Leader's Payoff	
	Without Danskin	With Danskin	Without Danskin	With Danskin
15	53 ± 16	48 ± 13	0.051 ± 0.0035	0.051 ± 0.0043
20	255 ± 43	212 ± 41	0.052 ± 0.0039	0.053 ± 0.0048
25	551 ± 87	409 ± 81	0.046 ± 0.0032	0.047 ± 0.0041
30	804 ± 127	706 ± 118	0.045 ± 0.0040	0.045 ± 0.0049
40	1354 ± 205	1128 ± 182	0.040 ± 0.0049	0.041 ± 0.0057

response need only be computed once per generation corresponding to the distribution mean, instead of for every candidate solution sampled under CMA-ES. This reduces the number of payoff evaluations and therefore lowers runtime. Although the overall saving may appear modest due to the presence of other computational overheads, it can still prove significant for many practical applications.

Table 8 summarizes the results. Computation time is consistently reduced, with an average speedup of around 1.2×, while the Leader's payoff remains statistically indistinguishable across settings. Interestingly, the standard deviation of the payoff is slightly higher when Danskin's theorem is applied. This can be attributed to the stochastic nature of CMA-ES sampling: reusing the Follower's best response at the mean introduces small fluctuations in the fitness landscape approximation, which occasionally lead to marginally different local optima. Nevertheless, these variations remain within statistical noise and do not affect the overall competitiveness of the method. These findings confirm that Danskin's theorem is a valuable enhancement for ADSO in zero-sum games, enabling measurable scalability improvements in WHG while preserving high-quality solutions. This improvement is especially relevant in practice, where even moderate runtime reductions can greatly expand the range of solvable instances.

7.6 Parameter Sensitivity Analysis

To assess the robustness of the ADSO framework and justify the choice of hyperparameters presented in Section 6.2, we conducted a sensitivity analysis focusing on two critical parameters: the population size (N) and the learning rate (η) used for the binary distribution update. The experiments were performed on Zero-Sum Warehouse Games (WHG) with acceleration through the use of Danskin's theorem. All results reported below are averaged over all game instances and 30 independent runs for each instance.

7.6.1 Impact of Population Size. The population size determines the trade-off between the exploration capability of the algorithm and its computational cost per generation. We tested $N \in \{50, 100, 200, 500\}$ while keeping the learning rate fixed at $\eta = 10^{-4}$. Table 9 summarizes the impact of population size on the Leader's payoff, solution sparsity (average number of pure strategies in the final Leader's strategy), and computation time.

As expected, increasing the population size generally improves the quality of the solution (Leader's Payoff) and reduces the variance (lower standard deviation). However, this improvement comes with diminishing returns. The transition from $N = 50$ to $N = 200$ yields a noticeable gain in payoff (from 0.041 to 0.046). Conversely, further increasing N to 500 provides negligible improvement while nearly tripling the computation time. Interestingly, smaller populations ($N = 50$) tend to converge to less sparse solutions (higher number of pure strategies), likely due to insufficient selection pressure to prune suboptimal strategies effectively. The chosen value of $N = 200$ represents a balanced compromise, offering near-optimal solution quality with reasonable computational overhead.

Table 9. Impact of Population Size (N) on algorithm performance (averaged over all WHG instances). The default setting used in the paper is bolded.

Population Size (N)	Leader's Payoff	Sparsity	Computation Time [s]
50	0.041 ± 0.010	8.9 ± 2.3	179 ± 35
100	0.043 ± 0.008	7.1 ± 1.7	293 ± 48
200	**0.046 ± 0.005**	**6.8 ± 1.5**	**504 ± 78**
500	0.047 ± 0.004	6.6 ± 1.4	1494 ± 181

7.6.2 Impact of Learning Rate. The learning rate η controls the magnitude of updates to the binary probability vector q. It directly influences how quickly the algorithm commits to a specific subset of pure strategies (sparsity induction). We evaluated $\eta \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ with a fixed population size of $N = 200$. The results are presented in Table 10.

Table 10. Impact of Learning Rate (η) on algorithm performance (averaged over all WHG instances). The default setting used in the paper is bolded.

Learning Rate (η)	Leader's Payoff	Sparsity	Computation Time [s]
10^{-5}	0.046 ± 0.004	7.1 ± 1.2	972 ± 108
$\mathbf{10^{-4}}$	**0.046 ± 0.005**	**6.8 ± 1.5**	**504 ± 78**
10^{-3}	0.045 ± 0.007	5.7 ± 1.7	396 ± 57
10^{-2}	0.040 ± 0.010	3.8 ± 1.2	327 ± 46

A very low learning rate ($\eta = 10^{-5}$) results in slow convergence. While the final solution quality is high, the computation time increases significantly because the algorithm requires more generations to satisfy the termination criteria. On the other hand, aggressive learning rates ($\eta \geq 10^{-3}$) lead to premature convergence. In these cases, the algorithm drastically prunes the search space too early, often locking into local optima with suboptimal payoffs and excessive sparsity (very few pure strategies). The default value of $\eta = 10^{-4}$ proves to be the most effective, allowing for a gradual discovery of the optimal sparse support without premature loss of diversity.

8 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 (and Danskin's theorem in the case of zero-sum SSGs), 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 [Yu et al. 2019] and large-scale resource allocation problems [Narayanan et al. 2021].

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

Abhishek Gupta was supported by the Anusandhan National Research Foundation (ANRF), Government of India, Ramanujan Fellowship grant RJF/2022/000115.

References

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.

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

Elizabeth Bondi, Hoon Oh, Haifeng Xu, Fei Fang, Bistra Dilkina, and Milind Tambe. 2020. To signal or not to signal: Exploiting uncertain real-time information in signaling games for security and sustainability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1369–1377.

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.

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.

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.

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.

John M Danskin. 1966. The theory of max-min, with applications. *SIAM J. Appl. Math.* 14, 4 (1966), 641–664.

Karel Durkota, Viliam Lisỳ, Branislav Bosanskỳ, and Christopher Kiekintveld. 2015. Optimal Network Security Hardening Using Attack Graph Games.. In *IJCAI*. 526–532.

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.

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.

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

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

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

Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordónez, and Milind Tambe. 2010a. Security games with arbitrary schedules: A branch and price approach. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 24. 792–797.

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

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

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

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.

Jan Karwowski, Jacek Mańdziuk, and Adam Żychowski. 2023. Sequential Stackelberg Games with bounded rationality. *Applied Soft Computing* 132 (2023), 109846.

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

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

Solomon Kullback. 1951. Kullback-leibler divergence. *Tech. Rep.* (1951).

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

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.

Thanh Hong Nguyen, Rong Yang, Amos Azaria, Sarit Kraus, and Milind Tambe. 2013. Analyzing the effectiveness of adversary modeling in security games. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*. 718–724.

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.

Fernando Ordónez, Milind Tambe, Juan F Jara, Manish Jain, Christopher Kiekintveld, and Jason Tsai. 2012. Deployed security games for patrol planning. In *Handbook of operations research for homeland security*. Springer, 45–72.

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.

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 34th AAAI Conference on Artificial Intelligence*. 1378–1386.

James Pita, Manish Jain, Fernando Ordóñez, Milind Tambe, Sarit Kraus, Reuma Magori-Cohen, and Milind Tambe. 2011. Effective Solutions for Real-World Stackelberg Games: When Agents Must Deal with Human Uncertainties. *Security and Game Theory* (2011), 193–212.

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.

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.

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.

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.

Jason Tsai, Shyamsunder Rathi, Christopher Kiekintveld, Fernando Ordonez, and Milind Tambe. 2009. IRIS-a tool for strategic security allocation in transportation networks. *AAMAS (Industry Track)* (2009), 37–44.

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.

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.

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.

Yufei Wang, Zheyuan Ryan Shi, Lantao Yu, Yi Wu, Rohit Singh, Lucas Joppa, and Fei Fang. 2019. Deep reinforcement learning for green security games with real-time information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1401–1408.

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

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.

Rong Yang, Christopher Kiekintveld, Fernando Ordóñez, Milind Tambe, and Richard John. 2013. Improving resource allocation strategies against human adversaries in security games: An extended study. *Artificial Intelligence* 195, 195 (2013), 440–469.

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

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

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.

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 the Genetic and Evolutionary Computation Conference*. 854–862.

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

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

Adam Żychowski and Jacek Mańdziuk. 2021b. Learning attacker's bounded rationality model in security games. In *Proceedings of the 28th ICONIP*, Vol. CCIS 1516. 530–539.

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

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.

Received XX Jan XX; revised XX Jan XX; accepted XX Jan XX