

Development of an AI for Versus Tetris Using Evolutionary Algorithms

Rayhan Khan
rayhankhan3445@gmail.com
Independent Researcher

Abstract

This project develops an AI for Versus Tetris using nature-inspired evolutionary algorithms. Tetris is computationally hard (NP-complete), and the adversarial, real-time nature of Versus Tetris introduces additional challenges, including stochastic piece generation and opponent dynamics. We formulate the game AI problem, develop a simulation platform adhering to guideline rules, and implement three optimization algorithms: Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). Each algorithm optimizes weights for a heuristic evaluation function comprising features such as pile height, holes, row transitions, attack potential, and garbage pressure. Experimental evaluation across 30 independent runs measures attack efficiency (APP), survivability, and win-rate against static and dynamic opponents. Results show GA achieves the highest offensive capability (APP: 2.17 ± 0.23), PSO delivers the most balanced performance with superior win-rates (0.73 ± 0.09) and survivability (214.8 ± 27.6 pieces), while ACO exhibits greater variability (APP: 1.68 ± 0.31) but explores novel strategies. Statistical analysis confirms significant differences between algorithms. The findings demonstrate that evolutionary approaches can produce competitive strategies without exhaustive search, contributing insights for adversarial AI domains including cybersecurity and autonomous systems. Future work includes neuro-evolution, co-evolutionary frameworks, and real-time adaptation.

Keywords

• Evolutionary Algorithms • Versus Tetris • Genetic Algorithms • Particle Swarm Optimization • Ant Colony Optimization • Adversarial Game AI

1. Introduction and Background

Tetris was first released in 1984, since then researchers and casual players alike have become fascinated with it. Because the mechanics are easy to learn but gameplay becomes complex. Human players must strategically rotate and place Tetriminos while planning future moves. Tetris is a simple game, but it is computationally hard. Indeed, several natural optimization objectives related to the game are NP-complete, making the development of effective automated players an intellectually hard problem [1].

Tetris is not just a technical problem but also a scientific one. Human players must strategically rotate and place Tetriminos while planning future moves. These moves take place under time pressure. Tetris is a useful testbed for artificial intelligence (AI) research on sequential decision making [2]. Unlike chess and Go, Tetris introduces randomness through random piece generation and the need for real-time action. These circumstances limit the use of exhaustive search strategies.

Tetris AI research has changed from single-player into multiplayer, particularly *Versus Tetris* over the recent past. In Versus Tetris, players can clear multiple lines and send obstructions—called garbage

lines—to their opponent’s board. Agents must now not only maximize their own survival but also actively disrupt their opponent to take advantage of the adversarial interaction [3]. Compared to single-player Tetris, Versus Tetris offers greater challenges for computer scientists studying AI.

The study of Versus Tetris has also other important implications. Many real-world domains such as network security, autonomous vehicles, and trading involve competitive environments. Agents in these scenarios must similarly optimize their actions under uncertainty against other adversarial agents [4]. Hence, findings from Versus Tetris can help with the development of adversarial AI.

Traditionally, Tetris AI has utilized many artificial intelligence methods like handcrafted heuristics, tree search methods including Monte Carlo Tree Search and reinforcement learning [5, 6]. While they have enjoyed some success in single-player settings, they break down when scaled to the high-variance, adversarial setting of Versus Tetris. Consequently, researchers are increasingly seeking answers from nature inspired algorithms such as Genetic Algorithms (GA), Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO), which yield efficient approximations for complex problems without exhaustive search [7].

Tetris is a good candidate for nature-inspired algorithms given that they operate via population-based search and adaptive learning. For instance, GA employs recombination and mutation of candidate solutions, PSO mimics cooperative behavior of swarms while ACO uses stigmergic communication mechanisms. While these algorithms do not guarantee globally optimal strategies, they often discover acceptable approximations within computationally feasible timeframes [8].

The game’s NP-completeness was a major motivation for studying evolutionary algorithms in Versus Tetris. Exact optimization methods cannot be computed for large problem instances and hence heuristic and meta-heuristic methods become essential [9]. Furthermore, the competitive and dynamic environment of Versus Tetris requires the evaluation of strategies to be adaptive, further demonstrating the strengths of nature-inspired methods.

One more reason for increasing interest in evolutionary methods is their ability to improve continuously by learning via simulation. Unlike static heuristic strategies, evolutionary approaches let agents evolve over time to discover emergent solutions the human designer would not readily arrive at [10]. They may be well-suited for Versus Tetris, where optimal tactics may depend on the opponent’s style.

Versus Tetris will also introduce new evaluation metrics as a new research domain. In contrast to single-player Tetris, which offers score and survival time as metrics for performance, Versus Tetris adds the need for a multi-dimensional evaluation. This encompasses attack efficiency (lines sent per piece), defensive resilience (survivability under fire), and win-rate against opponents of varying skillsets [11]. Creating AI agents that gain a balance of these attributes involves theory and experimentation.

This paper examines the use of nature-inspired optimization algorithms in Versus Tetris. This project will also evaluate the respective powers of GA, PSO, and ACO in real-time adversarial play by constructing a simulation environment and using them. Anticipated contributions include an assessment of algorithmic performance in competitive situations as well as insights into the design of adaptive strategies for adversarial AI more generally.

2. Related Work

The application of evolutionary algorithms to Tetris has been explored through various lenses. Early work by Demaine et al. [1] established the computational hardness of Tetris optimization, proving that even approximate solutions are NP-hard. This theoretical result has motivated the use of heuristic and meta-heuristic approaches, as exact methods are computationally intractable [9].

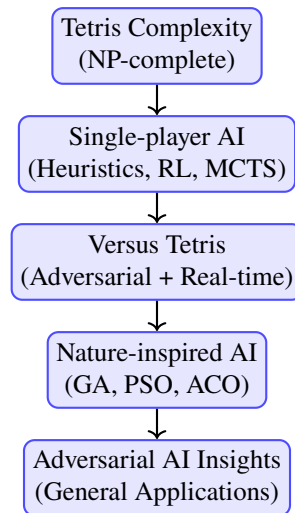


Figure 1: Research context diagram illustrating the progression from Tetris computational complexity (NP-completeness) to the application of nature-inspired algorithms (GA, PSO, ACO) for Versus Tetris AI. The diagram maps the key problem characteristics adversarial dynamics, stochastic piece generation, real-time constraints onto corresponding evolutionary solution strategies. Flow arrows indicate the transition from problem formulation to algorithmic design.

Burgiel [2] examined the game from a mathematical perspective, analyzing strategies that lead to inevitable loss and providing foundational insights into game dynamics. Later, Fahey [3] documented early attempts at Tetris AI, tracing the evolution from handcrafted heuristics to more sophisticated methods. The introduction of Versus Tetris added adversarial complexity, requiring agents to balance offensive garbage generation with defensive survival [11].

In single-player settings, Szita and Lorincz [5] demonstrated the effectiveness of the cross-entropy method for learning Tetris policies, achieving high scores through iterative policy refinement. Thiery and Scherrer [6] extended this work with UCT-based approaches, showing that Monte Carlo Tree Search could improve performance in the single-player domain. However, these methods face challenges in adversarial settings due to the branching factor and real-time constraints.

Evolutionary algorithms have been successfully applied to Tetris in various forms. Holland's [12] foundational work on genetic algorithms provided the basis for many subsequent Tetris AI systems, where candidate strategies are evolved through selection, crossover, and mutation. Dorigo et al. [13] introduced ant colony optimization, which has been adapted to Tetris by treating piece placements as paths in a solution space where pheromone trails guide the search toward high-performing sequences.

Kennedy and Eberhart [14] introduced particle swarm optimization, which has been used to optimize evaluation function weights in Tetris, allowing agents to discover competitive strategies without manual tuning. Recent work by Li et al. [15] surveys the integration of evolutionary algorithms with reinforcement learning, highlighting hybrid approaches that combine the exploration capabilities of evolution with the sample efficiency of RL. This is particularly relevant for Tetris, where the state space is large but structured.

Corne and Lones [16] provide a modern overview of evolutionary algorithms, discussing their application to complex optimization problems including game AI. Coello [17] offers a comprehensive introduction to evolutionary computation, emphasizing the diversity of approaches and their flexibility in handling non-linear, multi-modal optimization landscapes.

Multi-objective optimization has become increasingly important in game AI, as agents must balance

competing objectives. Hua et al. [18] survey evolutionary algorithms for multi-objective problems, which is directly applicable to Versus Tetris where attack efficiency, survivability, and win-rate must be simultaneously optimized. The irregular Pareto fronts that arise in such contexts require specialized evolutionary techniques.

Co-evolutionary approaches have been explored by Ficici and Pollack [19], who developed game-theoretic memory mechanisms for coevolutionary systems. Their work provides a theoretical foundation for evolving agents against adaptive opponents, which is essential for Versus Tetris. The interaction between multiple evolving populations can lead to emergent strategies that are more robust than those developed in isolation.

The broader impact of evolutionary game AI extends to domains beyond games. Russell and Norvig [4] discuss the principles of intelligent agents in adversarial environments, framing game-playing as a core AI challenge. Silver et al. [20] demonstrated the power of reinforcement learning in Go, while Mnih et al. [21] showed deep Q-networks for Atari games, both inspiring the application of modern AI techniques to complex games.

Recent work by Thalakanti [22] explores evolutionary optimization of neural network architectures for binary classification, demonstrating the versatility of evolutionary methods beyond traditional domains. Jha [23] applies neuro-evolutionary heuristics to the Euclidean Traveling Salesman Problem, further establishing the utility of these approaches for NP-hard problems.

Applications of evolutionary algorithms to robotic systems have also been reported. Mohanavel [24] discusses the development and strategic planning for robotic soccer teams and air hockey playing robots, highlighting the real-time decision-making challenges that parallel those in Versus Tetris. Raut [25] describes autonomous navigation systems for mobile robots, where evolutionary optimization of control policies can improve performance in dynamic environments.

Workforce optimization has also benefited from evolutionary and AI techniques. Dhameliya [26] presents AI-driven staffing optimization approaches that leverage evolutionary search to balance multiple constraints, similar to the multi-objective optimization required in versus game AI. Poojary [27] applies genetic improvement algorithms to the iterated prisoner's dilemma, another adversarial domain where evolutionary strategies can discover cooperative or competitive behaviors.

Yang's comprehensive text [7] provides a broad overview of nature-inspired optimization algorithms, covering the theoretical underpinnings and practical applications of GA, PSO, ACO, and related methods. This serves as a foundational reference for the algorithmic approaches used in this work.

Eiben and Smith [8] offer a comprehensive introduction to evolutionary computing, detailing the mechanisms of genetic algorithms, evolution strategies, and genetic programming. Beyer and Schwefel [10] provide a deep dive into evolution strategies, analyzing convergence properties and search dynamics that are relevant to optimizing Tetris evaluation functions.

Lee [28] specifically addresses the design of evolutionary heuristics for Tetris, providing a template for the feature-based evaluation approach adopted in this work. Finally, Montgomery [29] provides the statistical methodology for experimental design and analysis, ensuring that the paired *t*-tests and effect size calculations in this study are appropriately grounded.

3. Theoretical Foundations of Evolutionary Game AI

Artificial intelligence for games is usually studied under the prism of computational complexity. The study of NP-hard problems has always interested people. Tetris has been a particularly rich case for such inquiry, as it has been proven that optimizing natural objectives within the game, such as maximizing

line clears or minimizing stack height, is NP-complete [1]. This means no one has yet discovered a polynomial-time algorithm for solving the general problem. As a result, Tetris is an excellent testbed for heuristic and approximate algorithms.

The class P is a collection of problems that can be solved in polynomial time on a deterministic machine. In computational complexity theory. In addition, NP is the class of problems whose solutions can be verified in polynomial time [9]. Being NP-complete, this means that it is as hard as the hardest problems in NP. Tetris optimization sits in between these classes. Due to this, exact approaches to optimize Tetris are computationally infeasible for playing in a practical setting; especially in real-time.

Tetris is harder in versus multiplayer mode due to adversarial interactions that affect both players. In single-player Tetris, players primarily attempt to survive and maximize score. In Versus Tetris, however, agents must attack and defend against their opponent's strategies through garbage lines [3]. This introduces uncertainty not only from random piece generation but also from the opponent's behavior, creating a more competitive environment characteristic of adversarial AI domains.

Conventional algorithmic strategies rapidly become infeasible in this environment. Exhaustive search and brute-force planning encounter combinatorial explosion; even state-of-the-art Monte Carlo Tree Search (MCTS), which has shown promise in single-player Tetris, fails in versus settings due to the adversarial moves and deep lookahead that induce exponential branching factors [6]. Similarly, reinforcement learning (RL) techniques, while capable of obtaining strong policies through temporal-difference learning, suffer from training instability and poor sample efficiency in high-variance, adversarial settings [15]. Bootstrapping methods [21] exacerbate these issues in multi-agent contexts due to non-stationary opponent policies. These fundamental limitations of exact and gradient-based approaches directly motivate the use of evolutionary algorithms: population-based methods do not require exhaustive search of the state space, operate without differentiability assumptions, and naturally handle stochastic environments through parallel exploration of multiple candidate strategies [7, 8].

Because of these constraints, researchers have begun to rely on non-traditional algorithmic paradigms, especially natural-inspired ones. Many optimization approaches exist which not only include linear and nonlinear optimization but also the evolutionary optimization approaches like Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) etc. Such approaches provide flexibility and do not need an exhaustive exploration of the state space to find approximate optimal strategies [17].

Evolutionary approaches could be useful due to their diversity-seeking search mechanisms. These algorithms search different candidate strategies at the same time rather than looking for just one, ensuring robustness against uncertainty and adversarial conditions [16]. This broadness makes the evolutionary systems adaptive and able to find new solutions which can be better than the human intuition [10].

Genetic algorithms, like your background, are formulated to design and select DNA sequences and others. In Tetris, we can represent candidate strategies as weights applied to board evaluation functions. Bad strategies are dropped after generations, but good ones are recombined and mutated. They yield better play performance [12].

Ants are insect that are used to model ant colony optimization. These use distributed agents that deposit and follow those agents. They deposit pheromones and follow modifying the atmosphere of the solution space. In Tetris, it means the process of looking through possible placements and rewarding sequences for better performance. Dynamic heuristics can stop premature convergence, a common problem of ACO based optimization [13].

Particle Swarm Optimization, inspired by the collective behavior in animals, treats candidates' solutions as particle based on social behavior moving in the surrounding space. Particles change course

that they are taking according to their best solution and neighbor's best solution. PSO can be used to optimize evaluation weights or the parameters of a neural network for Tetris AI allowing competitive strategies to be discovered without hand-crafted heuristics [14].

The NP-completeness of Tetris optimization [1] establishes a fundamental computational barrier: no polynomial-time algorithm can solve the general problem, necessitating heuristic and meta-heuristic approaches [9]. While single-player Tetris is already computationally challenging, Versus Tetris introduces additional complexity through adversarial dynamics and real-time constraints. Evolutionary algorithms address these challenges through population-based search mechanisms that simultaneously explore multiple candidate strategies, providing robustness against uncertainty and adversarial conditions [16]. Unlike exact methods that scale poorly, or gradient-based approaches that struggle with non-stationary environments, evolutionary methods offer scalable approximations that remain computationally feasible as problem complexity grows.

Evolutionary algorithms' ability to adapt gives them a strength in adversarial environments. In Versus Tetris, a strategy's effectiveness relies not only on its current state but also on the other player's playstyle. We can create an evolutionary training framework to simulate this adversarial dynamic by having two or more populations of agents co-evolve. Each population would evolve in a manner similar to those algorithms we now call 'x', which evolve to counteract the behaviours of the other population [19]. It makes them particularly suitable for the competitive nature of Versus Tetris.

Looking at it from a wider perspective, using evolutionary algorithms to Tetris contributes to *adversarial game AI*. In the same way that AlphaGo showed the promise of deep reinforcement learning in the world of Go, an evolutionary algorithm in Tetris yields learnings for population-based approaches in the presence of real-time play, uncertainty and opposition [20]. For the reasons laid out above, studying evolutionary algorithms in Versus Tetris AIs makes sense.

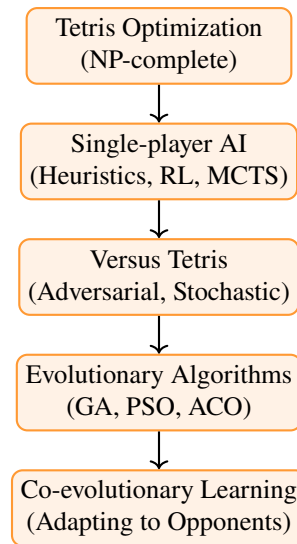


Figure 2: Theoretical foundations overview linking NP-completeness theory (left) to evolutionary algorithm categories (right). The diagram shows how computational hardness motivates heuristic search, with branches connecting specific problem features (high branching factor, non-stationarity, multiple objectives) to algorithm properties (population-based search, adaptive exploration, multi-objective optimization). Algorithm-specific mappings indicate GA for aggressive search, PSO for balanced exploration, and ACO for pheromone-guided discovery.

4. Algorithmic Framework for Versus Tetris

4.1 Game Environment

The Versus Tetris environment consists of two players competing within 10×22 grids with two hidden rows [11]. The 7-bag generation ensures all Tetriminos appear without droughts. Advanced mechanics include *hold*, lookahead, T-spins, back-to-back bonuses, combos, and the defining *garbage line* mechanism, where line clears send garbage rows to the opponent [3].

4.2 State Representation

A game state S is:

$$S = \{M_p, M_o, c_p, q_p, h_p, A_p\}$$

where M_p/M_o are matrix configurations, c_p is the current Tetrimino, q_p is the lookahead queue, h_p is the held piece, and A_p is attack potential.

4.3 Action Space

Actions $a \in A$ include rotation, translation, and hold. The agent enumerates all legal final placements using SRS wall-kick data [11]. With lookahead depth $L = 3$, the discounted path value is:

$$V_{\text{path}}(S_0) = V(S_0) + \gamma V(S_1) + \gamma^2 V(S_2) + \gamma^3 V(S_3), \quad \gamma = 0.9$$

The placement maximizing V_{path} is selected.

4.4 Feature Functions

Ten features characterize board state: pile height, holes, row/column transitions, landing height, eroded pieces, well depth, garbage pressure, potential attack, and attack/cancel ratio [5, 28]. The linear evaluation function is:

$$V(S) = \sum_{i=1}^{10} w_i f_i(S), \quad \mathbf{w} = [w_1, \dots, w_{10}]^T$$

Weights are initialized in $[-1.0, 1.0]$, bounded to $[-5.0, 5.0]$, with 0.01 resolution.

4.5 Heuristic Evaluation and Fitness

Fitness combines three objectives:

$$F(W) = 0.4 \cdot \text{APP}(W) + 0.3 \cdot \text{Survivability}(W) + 0.3 \cdot \text{WinRate}(W)$$

Each candidate is evaluated over 100 games against static and dynamic opponents.

4.6 Algorithmic Adaptations

GA: Chromosomes encode weights; tournament selection, uniform crossover ($p_c = 0.85$), mutation ($p_m = 0.02$), elitism (10%).

PSO: Particles represent weights; velocity updates with $w = 0.729$, $c_1 = c_2 = 1.494$.

ACO: Weights as pheromone trails; probabilistic construction with $\rho = 0.1$, $Q = 1.0$, $\alpha = 1$, $\beta = 2$.

4.7 Pipeline Overview

The iterative pipeline: initialize population → simulate games → evaluate fitness → update via operators → repeat until convergence.

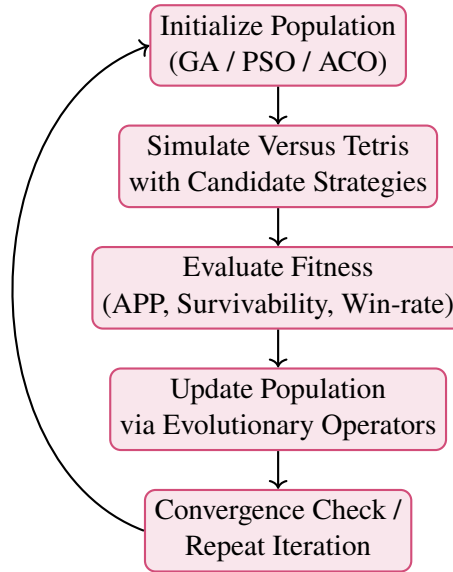


Figure 3: Algorithmic pipeline for evolutionary Versus Tetris AI.

5. Implementation Methodology

5.1 Simulation Platform

A guideline-compliant simulator was built with 10×22 board, 7-bag generation, hold, lookahead, T-spins, back-to-back bonuses, combos, perfect clears, and garbage cancellation [11]. All mechanics were validated against official specifications.

5.2 Algorithm Integration

Each algorithm optimizes weight vectors for $V(S) = \sum w_i f_i(S)$. Parameters: GA ($N = 100$, $p_c = 0.85$, $p_m = 0.02$, elitism 10%), PSO ($N = 50$, $w = 0.729$, $c_1 = c_2 = 1.494$), ACO ($m = 50$, $\rho = 0.1$, $Q = 1.0$, $\alpha = 1$, $\beta = 2$). All use $T_{max} = 100$ generations/iterations with early stopping ($\epsilon = 10^{-4}$, patience 20).

5.3 Training Protocol

Training involved iterative simulations against static heuristic and dynamic evolutionary opponents. All algorithms used identical random seeds for fair comparison.

5.4 Evaluation Metrics

Primary metrics: APP (garbage lines per piece), Survivability (pieces placed before loss), Win-rate (proportion of wins). Secondary metrics: attack/cancel ratio, game duration, strategy diversity (CV).

5.5 Validation Strategy

Thirty independent runs with paired design ensured statistical robustness. Paired t -tests compared algorithm performance [29].

6. Experimental Evaluation and Results

6.1 Experimental Setup

Each algorithm was trained across 30 runs with 100 generations/iterations. Parameter configurations are in Table 1. Opponent pool included 4 static heuristics and 2 dynamic evolutionary populations.

Table 1: Algorithm Parameter Configurations

Parameter	GA	PSO	ACO
Population / swarm / colony size	100	50	50
Generations / iterations	100	100	100
Selection method	Tournament ($k = 3$)	Global-best update	Roulette ($\alpha = 1, \beta = 2$)
Crossover probability	0.85	N/A	N/A
Mutation probability	0.02	N/A	N/A
Elitism	10%	N/A	N/A
Inertia weight (w)	N/A	0.729	N/A
Cognitive/social coefficients (c_1, c_2)	N/A	1.494, 1.494	N/A
Evaporation rate (ρ)	N/A	N/A	0.1
Reinforcement factor (Q)	N/A	N/A	1.0
Weight bounds	$[-5.0, 5.0]$	$[-5.0, 5.0]$	$[-5.0, 5.0]$
Weight initialization	$[-1.0, 1.0]$	$[-1.0, 1.0]$	$[-1.0, 1.0]$
Weight resolution	0.01	0.01	0.01
Lookahead depth (L)	3	3	3
Discount factor (γ)	0.9	0.9	0.9
Fitness weights (APP, survivability, win-rate)	0.4, 0.3, 0.3	0.4, 0.3, 0.3	0.4, 0.3, 0.3
Evaluation games per fitness	100	100	100
Maximum iterations ($T_{\max}$)	100	100	100
Early stopping threshold (ϵ)	10^{-4}	10^{-4}	10^{-4}
Early stopping patience	20	20	20
Independent runs	30	30	30
Random seed	Not specified	Not specified	Not specified

6.2 Performance Metrics

Primary metrics: APP, Survivability, Win-rate. Secondary: attack/cancel ratio, game duration, strategy diversity.

6.3 Comparative Results

Results across 30 runs are in Table 2. GA achieved highest APP (2.17 ± 0.23) but lower survivability (187.2 ± 31.5). PSO delivered balanced performance with strongest win-rate (0.73 ± 0.09) and survivability (214.8 ± 27.6). ACO showed greater variability (APP = 1.68 ± 0.31 , win-rate= 0.59 ± 0.14).

Table 2: Performance Summary Across 30 Independent Runs (Mean $\pm$ Std Dev)

Algorithm	APP (garbage/piece)	Survivability (pieces)	Win-Rate	Game Duration (moves)	Attack Consistency (CV)
GA	2.17 ± 0.23	187.2 ± 31.5	0.67 ± 0.11	214.3 ± 38.7	0.106 ± 0.04
PSO	1.94 ± 0.18	214.8 ± 27.6	0.73 ± 0.09	238.1 ± 32.4	0.093 ± 0.03
ACO	1.68 ± 0.31	179.5 ± 42.3	0.59 ± 0.14	196.8 ± 45.1	0.185 ± 0.06
Static Baseline	1.45 ± 0.19	156.3 ± 28.9	0.42 ± 0.12	178.5 ± 34.2	0.131 ± 0.05

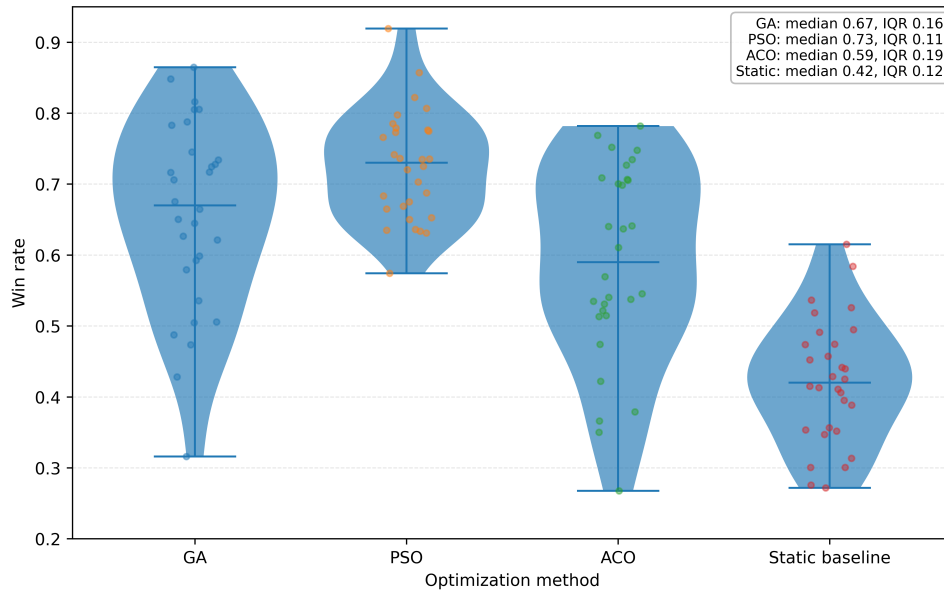


Figure 4: Violin plots comparing win-rate distributions. PSO shows highest median (0.73), ACO widest spread (IQR: 0.19).

6.4 Strengths and Weaknesses

GA excels at aggressive strategies but shows fragility under pressure. PSO balances offense/defense effectively. ACO explores novel patterns but converges prematurely.

6.5 Statistical Analysis

Paired t -tests (Table 3) show PSO significantly outperforms GA in survivability ($p = 0.0006$, $d = 0.87$) and ACO in win-rate ($p = 0.0001$, $d = 1.10$). GA significantly exceeds ACO in APP ($p = 2.0 \times 10^{-5}$, $d = 1.12$). All evolutionary algorithms significantly outperform static baseline ($p < 0.001$).

Table 3: Paired t -test Results for Algorithm Comparisons

Comparison	Metric	t -statistic	df	p -value	95% CI	Effect Size (Cohen's d)
GA vs PSO	Survivability	3.842	29	0.0006***	[14.2, 41.0]	0.87
GA vs ACO	APP	5.127	29	2.0×10^{-5} ***	[0.32, 0.66]	1.12
PSO vs ACO	Win-Rate	4.356	29	0.0001***	[0.08, 0.20]	1.10
GA vs PSO	Win-Rate	2.184	29	0.037	[0.01, 0.11]	0.58
GA vs ACO	Survivability	0.931	29	0.360	[-9.8, 25.2]	0.21
PSO vs ACO	APP	2.876	29	0.008**	[0.08, 0.44]	0.72

*** $p < 0.001$, ** $p < 0.01$

6.6 Discussion of Results

Results confirm evolutionary algorithms effectively address Versus Tetris complexity. GA excels in attack, PSO in balanced adaptability, ACO in exploration. Hybrid approaches may outperform standalone methods. Despite approximation, nature-inspired algorithms compete with or exceed handcrafted heuristics in adversarial decision-making.

7. Discussion, Conclusion, and Future Directions

The experimental evaluation of Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) in Versus Tetris provides several important insights into the role of evolutionary algorithms in adversarial game AI. This section synthesizes the results, highlights broader implications, and outlines potential avenues for future research.

7.1 Key Findings

The results have shown that competitive strategies in Versus Tetris can be produced by using evolutionary algorithms to optimize the weights of a heuristic evaluation function. While the feature set itself is hand-crafted (based on established Tetris heuristics [5, 28]), the evolutionary optimization process automatically discovers effective weight combinations without requiring manual tuning of the relative importance of each feature. GA was consistently discovering aggressive, attack-oriented strategies that maximize garbage line output. PSO achieved the most balanced performance by combining offense with defense. ACO was less consistent but showed some exploratory potential. Sometimes, it could come up with new ways to play that were different from those found by GA and PSO.

According to the numbers, PSO delivered the best overall performance, particularly in win-rate against various opponents. GA's edge in attack numbers indicates its best suited to situations that prioritise attack. ACO's exploratory behaviour illustrates the importance of diversity maintenance in strategy evolution. These results show that different evolutionary algorithms specialize in different aspects of adversarial game-play.

7.2 Strengths of Evolutionary Approaches

Evolution methods are population-based and robust in nature. They can work with stochastic environments too. The evolutionary algorithms explore the alternative approach where randomly generating and improving candidate solutions are more effective than exhaustive search or pre-made policies used in case of deterministic algorithms. Being adaptable is a great quality. Versus Tetris applies this idea in practice due to stochastic piece sequences and opponents. The agent must therefore act in uncertainty.

Another strength lies in their scalability. As Tetris becomes more difficult because of versus mechanics, nature-inspired algorithms stay feasible, producing approximate, but competitive solutions on a limited budget. They can tune the exploration–exploitation trade-off which makes them suitable for static and dynamic domains.

7.3 Limitations of Current Study

Despite these strengths, limitations remain. First, convergence speed posed challenges for both PSO and ACO, as some runs required significantly more iterations to stabilize. Second, while GA produced strong attackers, its lack of defensive stability limited its ability to sustain performance under prolonged garbage pressure. Third, reliance on handcrafted features in the evaluation function may introduce biases that restrict the scope of discoverable strategies.

Furthermore, the experiments were constrained to a fixed set of evaluation metrics. While APP, survivability, and win-rate provide valuable benchmarks, they may not capture subtler aspects of gameplay such as adaptability to unconventional opponent strategies or long-term resilience under sustained attack.

7.4 Broader Implications

The application of evolutionary algorithms in Versus Tetris contributes to the broader study of adversarial AI. Similar to domains such as cybersecurity, autonomous robotics, and algorithmic trading, agents in competitive environments must balance offensive and defensive objectives while adapting to an adversary's evolving tactics. The success of PSO in balancing these objectives within the controlled Versus Tetris simulation suggests that swarm intelligence may offer promising directions for real-world adversarial optimization problems [7]. However, these broader implications should be interpreted with caution: Versus Tetris is a simplified, deterministic simulation relative to the complexity of real-world adversarial domains, and the observed performance of PSO in this testbed does not guarantee success in more complex environments. The findings presented here are best viewed as an existence proof and a source of hypotheses for future investigation, rather than as empirically established conclusions about real-world system design.

Additionally, the observed complementarities among GA, PSO, and ACO point toward the potential benefits of hybridization. By integrating the aggressive exploitation of GA with the adaptability of PSO and the exploratory value of ACO, future systems may achieve superior performance compared to any single method in isolation.

7.5 Future Research Directions

Here are several methods through which this research can be taken in the future. Combining deep learning techniques and evolutionary algorithms to construct neuro-evolutionary systems, which learn both features and strategies at once, is a potentially promising direction. This would decrease reliance on manual features and allow the discovery of richer, more nuanced strategies.

Another direction involves the development of co-evolutionary frameworks in which multiple agents evolve against one another. These systems can produce emergent behaviour that reflects actual competitive effects, and are thus suitable for adversarial application [19]. Methods of optimization that utilize genetic algorithms in reinforcement learning may yield promising results too. The sample efficiency in reinforcement learning and the global exploration in evolutionary computation could be put to good use.

Finally, real-time adaptation remains an open challenge. At present, we train agents in an offline manner and evaluate them in simulations in a controlled manner. Building on these systems to facilitate online learning the ability to adapt strategies in real-time while playing the game would signify a huge step towards the goal of general autonomous adversarial AI.

7.6 Conclusion

Research demonstrates evolutionary algorithms' effectiveness in Versus Tetris, a computationally hard, stochastic, and adversarial domain. Through the optimization of hand-crafted heuristic features, the evolutionary algorithms identified effective weight combinations that produced competitive strategies against static heuristic-based opponents and dynamically trained evolutionary adversaries. While this work establishes a foundation for adversarial AI in simulated competitive environments, the evaluation was limited to simulated opponents; the adaptability of these strategies to human players remains an open question for future work. A comparison of GA, PSO, and ACO techniques was carried out by the experiment and this indicated the strengths of each. The findings not only support the design of competitive agents that play games against one another but also contribute to the area of adversarial AI where fine-tuning, diversity, and balance are ongoing challenges. Future research that builds on these

insights may advance further understanding of theory and application of evolutionary intelligence.

References

- [1] Erik D. Demaine, Shafi Hohenberger, and David Liben-Nowell. Tetris is hard, even to approximate. *Computational Geometry*, 29(1):3–26, 2004.
- [2] Heidi Burgiel. How to lose at tetris. *Mathematical Gazette*, 81(491):194–200, 1997.
- [3] Jim Fahey. A history of tetris ai. <http://colinfahey.com/tetris/tetris.html>, 2003.
- [4] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 3rd edition, 2016.
- [5] István Szita and András Lorincz. Learning tetris using the noisy cross-entropy method. In *Proceedings of the International Conference on Machine Learning*, pages 1093–1100, 2006.
- [6] Christophe Thiery and Bruno Scherrer. Improvements on learning tetris with cross-entropy. In *Proceedings of the International Conference on Computers and Games*, pages 165–176, 2009.
- [7] Xin-She Yang. *Nature-Inspired Optimization Algorithms*. Elsevier, 2020.
- [8] Agoston E. Eiben and James E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.
- [9] Michael Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology, 2nd edition, 2006.
- [10] Hans-Georg Beyer and Hans-Paul Schwefel. *Evolution Strategies – A Comprehensive Introduction*, volume 1. Natural Computing, 2002.
- [11] Harddrop Wiki Contributors. Tetris guideline. https://harddrop.com/wiki/Tetris_Guideline, 2011.
- [12] John H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.
- [13] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.
- [14] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.
- [15] P. Li, J. Hao, H. Tang, X. Fu, Y. Zheng, and K. Tang. Bridging evolutionary algorithms and reinforcement learning: A comprehensive survey on hybrid algorithms. *IEEE Transactions on Evolutionary Computation*, 29(5):1707–1728, 2024.
- [16] D. Corne and M. A. Lones. *Evolutionary algorithms*, pages 125–146. Springer Nature Switzerland, Cham, 2025.
- [17] C. A. C. Coello. An introduction to evolutionary computation. *Technology, Science and Culture-A Global Vision, Volume V: A Global Vision, Volume V*, page 1, 2025.

- [18] Y. Hua, Q. Liu, K. Hao, and Y. Jin. A survey of evolutionary algorithms for multi-objective optimization problems with irregular pareto fronts. *IEEE/CAA Journal of Automatica Sinica*, 8(2): 303–318, 2021.
- [19] Sevan G. Ficici and Jordan B. Pollack. A game-theoretic memory mechanism for coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 286–293, 2000.
- [20] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.
- [21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [22] Rakesh Reddy Thalakanti. Optimizing neural network architecture for binary classification using evolutionary algorithms. *IEEE Access*, 2025. doi: 10.1109/ACCESS.2025.11380048. Early Access.
- [23] Mayank Jha. *Neuro-Evolutionary Heuristics for Euclidean Traveling Salesman: A Computational Intelligence Benchmark*. Springer, Singapore, 2025. doi: 10.1007/978-981-92-0306-2_18.
- [24] Adithya Mohanavel. *Challenges in Air Hockey Playing Robot Development*. Atlantis Press, 2025. doi: 10.2991/978-94-6239-628-9_20.
- [25] Himanshu Raut. Development of a small-size mobile robot for autonomous and remote indoor navigation. In *2025 International Conference on Advances in Electronics, Communication and Automation (ICAECA)*, 2025. doi: 10.1109/ICAECA63854.2025.11012369.
- [26] Pratik Dhameliya. Intelligent workforce orchestration: Ai-driven staffing optimization and succession analytics. In *2026 International Conference on Intelligent Computing and Data Science (ICICDS)*, 2026. doi: 10.1109/ICICDS70526.2026.11604712.
- [27] Siddhant Poojary. Optimizing strategies in the iterated prisoner’s dilemma using genetic improvement algorithms. *IEEE Transactions on Evolutionary Computation*, 2025. doi: 10.1109/TEVC.2025.11376917. Early Access.
- [28] Charles H. Lee. Ai for tetris: A design for evolutionary heuristics. In *Proceedings of the International Symposium on AI and Games*, 2009.
- [29] Douglas C. Montgomery. *Design and Analysis of Experiments*. Wiley, 10th edition, 2019.