Co-Evolving Complexity: An Adversarial Framework for Automatic MARL Curricula

Brennen A. Hill
Department of Computer Science
University of Wisconsin-Madison
Madison, WI 53706
bahill4@wisc.edu

Abstract

The advancement of general-purpose intelligent agents is intrinsically linked to the environments in which they are trained. While scaling models and datasets has yielded remarkable capabilities, scaling the complexity, diversity, and interactivity of environments remains a crucial bottleneck. Hand-crafted environments are finite and often contain implicit biases, limiting the potential for agents to develop truly generalizable and robust skills. In this work, we propose a paradigm for generating a boundless and adaptive curriculum of challenges by framing the environment generation process as an adversarial game. We introduce a system where a team of cooperative multi-agent defenders learns to survive against a procedurally generative attacker. The attacker agent learns to produce increasingly challenging configurations of enemy units, dynamically creating novel worlds tailored to exploit the defenders' current weaknesses. Concurrently, the defender team learns cooperative strategies to overcome these generated threats. This co-evolutionary dynamic creates a self-scaling environment where complexity arises organically from the adversarial interaction, providing an effectively infinite stream of novel and relevant training data. We demonstrate that with minimal training, this approach leads to the emergence of complex, intelligent behaviors, such as flanking and shielding by the attacker, and focus-fire and spreading by the defenders. Our findings suggest that adversarial co-evolution is a powerful mechanism for automatically scaling environmental complexity, driving agents towards greater robustness and strategic depth.

1 Introduction

The development of intelligent agents, especially those leveraging Large Language Models (LLMs), has underscored the foundational role of environments in cultivating sophisticated behaviors [Wang et al., 2019]. Environments are not merely passive arenas for evaluation; they are the interactive substrate from which agents learn adaptive behavior, complex reasoning, and long-term planning. The trajectory of progress in machine learning has been marked by scaling laws: increasing model size, dataset volume, and computational power has unlocked emergent capabilities [Silver et al., 2017]. We posit that a similar principle applies to agent development, where scaling the structure, fidelity, and diversity of environments is a critical vector for advancing agent intelligence.

Recent breakthroughs in end-to-end reinforcement learning (RL) have made it feasible to train agents through sustained environmental interaction, moving beyond the limitations of imitation learning from static datasets [Schulman et al., 2017]. This shift places a greater demand on the environments themselves. To foster general-purpose autonomy, we require environments that are not only richly interactive but also perpetually novel, preventing agents from overfitting to a fixed set of scenarios.

The manual design of such environments is an intractable task, as it requires immense human effort and is subject to the designers' inherent biases and limited imagination.

This paper addresses the challenge of scaling environmental complexity through a novel framework: **Learned Adversarial Procedural Generation for Multi-Agent Curricula**. We reframe the problem of environment design as a two-player game between a generative *Attacker* and a team of cooperative *Defender* agents. The Attacker's goal is to learn a policy for procedurally generating sequences of hostile units (i.e., worlds or challenges) that can defeat the Defenders. Simultaneously, the multi-agent Defender team learns a cooperative policy to survive the Attacker's generated worlds for as long as possible.

This adversarial dynamic creates a natural and automatic curriculum. As the Defenders improve, the Attacker is incentivized to generate more sophisticated and complex challenges to remain competitive. This, in turn, forces the Defenders to develop more robust and coordinated strategies. To further drive this complexity, we designed the environment to have a combinatorially large action and state space. The defenders are not identical; each is assigned a unique role with different special abilities. The attacker, in turn, has a wide range of traits it can assign to the units it generates. In much of RL, problems are simple toy examples that do not extend beyond their initial experiment. By intentionally creating a more complex interaction with a vast space of possible environments and defender actions, we work to address that issue and create a more robust training paradigm. The result is a co-evolutionary arms race where the environment, embodied by the Attacker, continuously adapts to challenge the learning agents, effectively generating an infinite stream of increasingly difficult worlds.

Our primary contributions are:

1. We present a system architecture for co-evolving a generative adversarial agent and a team of cooperative agents. The adversary's role is to procedurally generate environmental challenges, creating an open-ended learning process.

2. We demonstrate that this adversarial framework leads to the rapid emergence of complex and intelligent strategies in both the generative Attacker and the cooperative Defender team.

3. We provide qualitative and quantitative evidence of emergent behaviors, such as the Attacker learning to flank and shield its units, and the Defenders learning to coordinate their movements and focus fire, behaviors which were not explicitly programmed.

4. We argue that this paradigm serves as a powerful method for scaling environments for agent training, shifting the focus from hand-crafting content to designing the rules of a self-perpetuating, complexity-generating system.

The remainder of this paper is structured as follows: Section 2 reviews related work in multi-agent reinforcement learning, procedural content generation, and adversarial learning. Section 3 details the formal problem setting and our proposed system architecture. Section 4 describes the implementation, including the agent models and training regime. Section 5 presents our experimental results, focusing on the emergent behaviors. Section 6 discusses the implications of our findings and the limitations of the current work. Finally, Section 7 concludes with a summary of our contributions.

2 Related Work

Our research is situated at the intersection of three key areas: Multi-Agent Reinforcement Learning (MARL), Procedural Content Generation (PCG), and the use of adversarial dynamics to create automatic curricula.

2.1 Multi-Agent Reinforcement Learning (MARL)

MARL extends reinforcement learning to scenarios with multiple interacting agents. A central challenge in MARL is non-stationarity: from any single agent's perspective, the environment is constantly changing as other agents adapt their policies [Buşoniu et al., 2008]. This makes learning unstable. Our system embraces this non-stationarity, leveraging it as the primary driver of learning for both the Attacker and the Defenders.

MARL problems can be categorized as cooperative, competitive, or mixed-motive [Zhang et al., 2019]. Our work features a mixed structure: the Defender team is fully cooperative, while the relationship between the Defender team and the Attacker is fully competitive, forming a game that is close to zero-sum. The formal framework for such interactions is the Partially Observable Markov Game (POMG), where agents must make decisions based on incomplete information about the true game state [Hansen et al., 2004, Liu et al., 2022]. In our setup, the Defenders have only partial observability of the Attacker's internal state, not seeing the Attacker's energy reserves and policy.

A dominant paradigm in modern MARL is Centralized Training with Decentralized Execution (CTDE) [de Witt et al., 2020]. In CTDE, agents use global information (e.g., a shared value function) during training to stabilize learning but act based only on their local observations during execution. Proximal Policy Optimization (PPO) [Schulman et al., 2017] has proven surprisingly effective in cooperative MARL settings when adapted to this paradigm (e.g., MAPPO), challenging the notion that on-policy methods are too sample-inefficient [Yu et al., 2022]. This body of work provides strong justification for our choice of PPO as the learning algorithm for the cooperative Defender team and the competitive Attacker.

2.2 Procedural Content Generation (PCG)

Procedural Content Generation refers to the algorithmic creation of game content. Traditional PCG methods are often constructive or search-based. A more recent paradigm is PCG via Machine Learning (PCGML), where models are trained on existing content to generate new, similar content [Summerville et al., 2018]. For example, models can learn to blend existing levels to create novel combinations [Guzdial and Riedl, 2016]. However, PCGML is fundamentally imitative and its creative potential is bounded by its training data.

To overcome this limitation, PCG via Reinforcement Learning (PCGRL) was introduced, framing content generation as an RL problem where an agent learns to iteratively modify a level to maximize a reward function based on desired properties like playability [Khalifa et al., 2020]. This approach is inventive rather than imitative, as it can discover novel content through exploration. Our work builds directly upon this idea, but instead of using a static, hand-crafted reward function, the reward signal for our generative Attacker is derived dynamically from the performance of another learning agent (the Defender team).

2.3 Adversarial Learning and Automatic Curricula

The core mechanism of our system is the adversarial dynamic between the generator and the solvers. This concept has deep roots in machine learning, most notably in Generative Adversarial Networks (GANs). In the context of RL, adversarial self-play has been shown to be a powerful engine for generating complexity and achieving superhuman performance without human data, as exemplified by AlphaGo and AlphaZero [Silver et al., 2016, 2017]. Similarly, competitive multi-agent environments have been shown to produce a natural curriculum, leading to the emergence of complex skills and strategies as agents continually adapt to one another [Bansal et al., 2018, Tampuu et al., 2017, Narvekar et al., 2020].

The explicit use of an adversary for PCG was explored by Volz et al. [Volz et al., 2021] and Gisslén et al. [Gisslén et al., 2021], who proposed a Generator-Solver framework where the generator is rewarded for creating challenging but solvable levels for a single solver agent. Our work extends this adversarial PCG paradigm in several critical dimensions. We transition from a single-solver setting to a multi-agent cooperative team, elevating the task from solving static puzzles to developing dynamic, coordinated strategies against a learning adversary. Second, our generator operates at a more fundamental level with fine-grained control over the challenge. We shift the focus from generating solvable static environments to orchestrating a dynamic, self-scaling curriculum.

This process of co-evolution, where agents and their environments develop in tandem, has been identified as a powerful method for open-ended learning. The POET algorithm, for instance, co-evolves a population of environments and agent policies, leading to the continual generation of novel and complex challenges [Wang et al., 2019]. Other work has explored co-evolving an agent's morphology alongside its environment [Ao et al., 2023]. Our system can be seen as a specific instantiation of this broader principle, using a competitive game to drive the co-evolution of environmental challenges (from the Attacker) and solving policies (from the Defenders). This dynamic automatically generates

goals of appropriate difficulty, a key principle in automatic curriculum generation [Florensa et al., 2018]. Finally, our use of PPO is further supported by its extensibility for training policies robust to adversarial perturbations [Wu et al., 2021, Zhang et al., 2020].

3   System Design and Methodology

We formalize our system as a two-team, nearly zero-sum, partially observable Markov game (POMG) [Hansen et al., 2004]. The game consists of Team D, a set of $N = 4$ cooperative Defender agents, and Team A, a single adversarial Attacker agent.

3.1   Environment

The game takes place on a discrete 2D grid, representing a board with 10 lanes (x-axis) and 30 tiles of depth (y-axis). The Defender agents are constrained to the first four rows ($y \in [0, 3]$), while the Attacker operates from the far end of the board ($y = 29$). Time proceeds in discrete timesteps. The Defenders win by surviving, while the Attacker wins if a unit reaches the bottom edge or if any Defender is defeated.
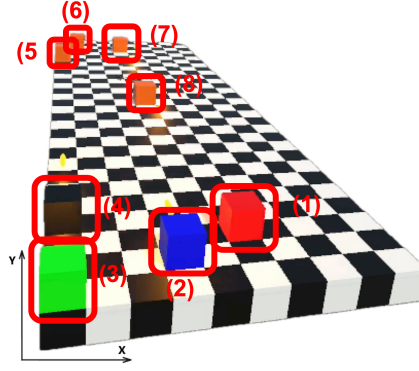

Figure 1: The game environment. The four Defender agents (1), (2), (3), and (4) can only move horizontally. The arbitrary number of orange Units (5), (6), (7), and (8) generated by the Attacker move vertically from the top of the board downwards.

3.2   Agents and Actions

3.2.1   The Defender Team

The Defender team consists of four agents, each assigned one of four unique roles, detailed in Table 5. Each Defender has a persistent state defined by its position (x_i, z_i), current health (max 100), and current energy (max 1000), which replenishes at 1 unit/tick. At each tick, each Defender $i \in \{1, ..., 4\}$ chooses an action from the discrete action space detailed in Table 1.

Table 1: Defender agent action space and energy costs.

Action	Energy Cost	Description
Move Left/Right	5	Change the agent's x-coordinate by -1 or +1.
Shoot	10	Fire a projectile down the current lane.
Heal	50	Restore a portion of its own health.
Special Ability	200	A powerful, role-specific action.
Do Nothing	0	Conserve energy.

4

3.2.2 The Attacker (Generative) Agent

The Attacker agent's role is to procedurally generate challenges. Its state is defined by its current and maximum energy, which slowly increases over time. At each tick, the Attacker chooses an action a^A: generate a unit with specific parameters or do nothing to conserve energy. The parameter vector θ that defines a unit's attributes is the core of the procedural generation, creating a vast combinatorial space of possible enemies, detailed in Table 2.

Table 2: Parameter space for the procedurally generated units by the Attacker agent. The agent can choose from a discrete set of values within the specified ranges to define a unit's attributes.

Category	Parameter & Range
Placement	Lane $x \in [0, 9]$
Core Stats	Health (1-15), Damage (1-5), Speed (1-5), Range (1-25)
Special Attributes	Regeneration (0-3 health/tick), Leech (0-5 health on attack)
Defenses	Physical Defense (0-5), Magic Defense (0-5)
Offenses	Physical and (0-5), Magic Penetration (0-5), Type (Physical/Magic)

The energy cost of generating a unit is a superlinear, multiplicative function of its parameters θ. More powerful units are exponentially more expensive, forcing the Attacker to make strategic trade-offs between quantity and quality.

3.2.3 Unit Behavior

Generated units are not controlled by the Attacker. They follow a hard-coded behavior: move forward (decrease y-coordinate) each tick. If a Defender is in the same lane and within range, the unit stops moving and attacks. This ensures challenge complexity arises from the Attacker's generative choices, not from complex unit AI.

3.3 Game Dynamics and Objectives

An episode begins with four Defenders and the Attacker. The episode ends when one of two termination conditions is met: (1) An enemy unit reaches the Defenders' baseline ($y < 0$), or (2) Any Defender's health drops to zero. If either occurs, the Attacker wins. The Defenders' objective is to survive as long as possible; the Attacker's is to win as quickly as possible.

4 Experiments and Results

The primary goal was to investigate whether intelligent, complex, and adaptive strategies would emerge from the adversarial co-evolutionary process. Success was measured by the qualitative and quantitative richness of the observed behaviors after 500 episodes, compared against a baseline where both sides selected actions uniformly at random.

4.1 Baseline Comparison

The baseline agents exhibited no intelligent behavior. The random attacker generated units with arbitrary parameters at random locations and times. The random defenders moved without purpose, failing to engage units or wasting energy. Episodes were brief, with defenders being quickly overwhelmed.

4.2 Emergent Behaviors

Intelligent strategies had clearly emerged after 500 episodes. The co-evolutionary pressure forced both sides to develop sophisticated tactics to counter each other.

4.2.1 Emergent Attacker Strategies

The Attacker learned to move beyond simply generating strong enemies and began to exhibit temporal and spatial reasoning.

The Attacker learned to move beyond simply generating strong enemies and began to exhibit temporal and spatial reasoning. It developed several distinct tactics, including: the **Rusher strategy**, where it generated units with very high speed and minimal other stats to race past defenders; the **Tandem strategy** (Figure 3a), a notable tactic of generating a high-health unit to act as a shield for a high-damage unit directly behind it in the same lane; and the **Flanking strategy** (Figure 3b), where it exploited the Defenders' limited mobility by generating simultaneous threats on opposite sides of the board.

4.2.2 Emergent Defender Strategies

In response, the Defender team developed coordinated behaviors. Despite a shared policy and no explicit communication channel, their actions became implicitly coordinated.

In response, the Defender team developed coordinated behaviors. Despite a shared policy and no explicit communication channel, their actions became implicitly coordinated. For example, they learned **Cooperative Spreading (Figure 4a)**, where, when faced with multiple threats (such as from Flanking), defenders learned to spread out to cover the relevant lanes. Conversely, they also learned **Cooperative Focusing (Figure 4b)**, where multiple defenders would converge on the same lane to concentrate firepower on a single, high-priority threat (such as from the Tandem strategy).

4.3 Quantitative Analysis

To ground our observations, we quantified the frequency of emergent strategies by defining and watching for four signature behaviors detailed in Table 7. quantitative metrics reported in this section are the average values obtained from 100 independent runs to ensure statistical reliability.

Table 3 presents the statistics for these strategies after 500 training episodes, averaged across 100 independent runs, comparing the trained agents to the random baseline. The difference is stark. The trained agents' average episode length was over four times longer than the baseline (83 steps vs. 19), demonstrating a vastly superior ability to survive. This increased survival time is directly attributable to the adoption of coherent strategies.

As shown in Table 3, the trained Attacker employed the Tandem and Flanking strategies in over 98% and 94% of episodes, respectively. These were not rare occurrences but the core of its learned policy. Similarly, the Defender team utilized Cooperative Spreading and Focusing in 92.6% and 81.4% of episodes. In contrast, the random baseline agents triggered these strategic patterns at rates below 11%, consistent with chance occurrences in a short episode. The average uses per episode show that the trained agents repeatedly and deliberately execute these tactics, whereas the random agents barely perform them once across ten episodes. This data provides strong quantitative validation that the adversarial process did not just improve agent performance but induced the learning of specific, recognizable, and effective multi-agent tactics.

Table 3: Frequency of Emergent Strategies After 500 Episodes, averaged across 100 runs.

Agent Type	Strategy Metric	Trained Agents	Random Baseline
Defender Cooperative Strategies			
Defender	Cooperative Spreading (Avg. Uses/Ep)	3.61	0.0104
	Cooperative Spreading (Usage Rate)	92.6%	0.837%
	Cooperative Focusing (Avg. Uses/Ep)	2.97	0.00906
	Cooperative Focusing (Usage Rate)	81.4%	0.548%
Attacker Generative Strategies			
Attacker	Flanking (Avg. Uses/Ep)	4.85	0.128
	Flanking (Usage Rate)	94.0%	10.3%
	Tandem (Avg. Uses/Ep)	8.01	0.0919
	Tandem (Usage Rate)	98.2%	6.73%
Avg. Episode Length (steps)		**83**	**19**

4.4 Ablation Study: The Necessity of Co-Evolution

To isolate the impact of the adversarial dynamic, we conducted two additional experiments where one agent was trained against a non-learning, random opponent for 500 episodes. The results, summarized in Table 4, underscore that co-evolution is the primary driver of strategic complexity.

First, we trained the Defender team against a perpetually random Attacker. The Defenders survived significantly longer against a random generator, with an average episode length of 216 steps. They rarely employed cooperative strategies, using Spreading at a rate of 13.2% and Focusing at 9.30%. Accounting for the longer episodes, the Defenders hardly improved from the random baseline. Qualitatively, many of these rare instances appeared more as random chance than intentional maneuvers. The challenge remained simple, and the learning plateaued.

Conversely, we trained the Attacker against a team of random Defenders. The results were even more telling. The average episode length plummeted to just 14 steps, as the random Defenders offered virtually no resistance. Critically, the incentive to develop intelligent strategies vanished. The Flanking and Tandem strategies appeared at rates of only 13.7% and 21.2%, respectively. Without a competent opponent to challenge it, the Attacker's policy failed to develop strategic depth, succeeding simple unit spawns. Together, these ablations provide strong evidence that it is the mutual, reciprocal adaptation, the co-evolutionary arms race, that generates the rich, emergent behaviors observed in our main experiment.

Table 4: Ablation Study: Strategy Frequency When Training Against a Random Opponent, averaged over 100 independent runs.

Agent Type	Metric	Value
Trained Defender vs. Random Attacker		
Defender	Avg. Episode Length (steps)	216
	Cooperative Spreading (Avg. Uses/Ep)	0.188
	Cooperative Spreading (Usage Rate)	13.2%
	Cooperative Focusing (Avg. Uses/Ep)	0.113
	Cooperative Focusing (Usage Rate)	9.30%
Trained Attacker vs. Random Defender		
Attacker	Avg. Episode Length (steps)	14
	Flanking (Avg. Uses/Ep)	0.184
	Flanking (Usage Rate)	13.7%
	Tandem (Avg. Uses/Ep)	0.328
	Tandem (Usage Rate)	21.2%

This strategic evolution is also reflected in the overall training dynamics. As shown in Figure 2, the average survival time of the defenders generally increased, indicating skill improvement. However, the curve is not monotonic; it exhibits significant oscillations. These dips often correspond to moments where the Attacker discovers a new, effective strategy that temporarily overcomes the Defenders' current policy. The Defenders then adapt, and the survival time climbs again. This oscillating pattern is characteristic of a co-evolutionary arms race and provides evidence of the ongoing adaptive process.

5 Discussion

Our results demonstrate that a co-evolutionary, adversarial framework can automatically generate a rich and adaptive curriculum for multi-agent systems. This section discusses the broader implications of this finding, situating it within the context of automatic curriculum generation, the nature of the emergent arms race, its impact on agent generalization, and its limitations.
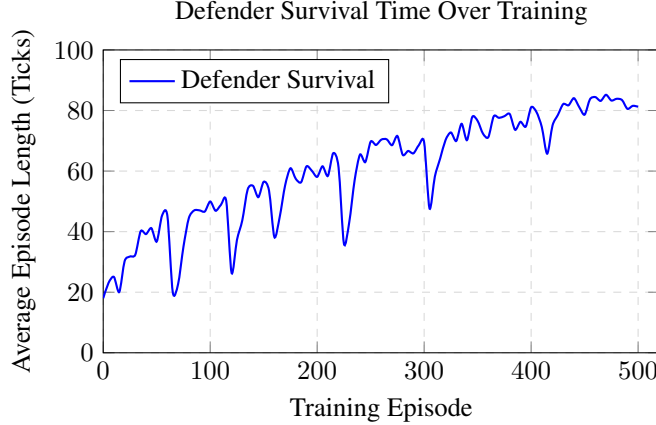
Figure 2: An illustration of the average episode length (Defender survival time in ticks) over training Episode. The plot represents the general upward trend indicating skill improvement, alongside the oscillations suggesting an ongoing arms race where the Attacker discovers new strategies. This curve is representative of the observed dynamic rather than a plot of raw data from a single training run.

5.1 Adversarial Generation as a Self-Scaling Curriculum

The findings strongly support the hypothesis that adversarial co-evolution is a potent mechanism for automatic curriculum generation. The Attacker agent, motivated to defeat the Defenders, is intrinsically driven to generate challenges at the frontier of the Defenders' capabilities. Challenges that are too easy do not yield rewards, while those that are truly impossible are prohibitively expensive in terms of energy, disincentivizing their creation. Consequently, the Attacker is naturally guided to probe and exploit specific weaknesses in the Defenders' current collective policy.

This process transforms the environment generator into a learned, adaptive loss function for the solver agents. Instead of training against a static dataset of challenges, the Defenders learn against a dynamic adversary whose sole purpose is to maximize their failure. This dynamic efficiently searches the combinatorially vast space of possible worlds (diverse units in Table 2) to find the small subset of configurations that are maximally informative for driving learning. The efficacy of this curriculum is quantitatively evident; trained agents adopt specific, complex tactics such as the Attacker's Tandem strategy (employed in 98.2% of episodes) and the Defenders' Cooperative Spreading (92.6%) (Table 3). This stands in stark contrast to our ablation studies, where training against a static, random opponent resulted in significantly less strategic depth (Table 4). The infinite stream of novel never-ending worlds is not just a theoretical construct; it is the emergent outcome of this adversarial dynamic, ensuring that the agents never exhaust their supply of relevant training data.

5.2 The Nature of the Co-evolutionary Arms Race

The oscillating performance curve seen in Figure 2 is a hallmark of a co-evolutionary arms race, a dynamic akin to the Red Queen hypothesis in evolutionary biology, where species must constantly adapt simply to maintain their viability against evolving competitors [Van Valen, 1973]. The dips in Defender survival time likely correspond to moments where the Attacker discovers a new, effective strategy (e.g., a novel combination of unit parameters that bypasses the Defenders' current meta). This creates a strong learning signal for the Defender team, which must then adapt and develop a counter-strategy, leading to a subsequent rise in survival time.

This perpetual non-stationarity, driven by the arms race, is a desirable feature for open-ended learning. Our ablation studies confirm its necessity; when one side ceased to adapt, the strategic evolution of the other quickly stalled (Table 4). It prevents the agents from converging to a single, brittle policy. Instead of reaching a stable Nash equilibrium, the system navigates a continuous cycle of strategies and counter-strategies. For instance, the Attacker may favor Flanking, forcing the Defenders to master Spreading. A proficient Spreading defense then incentivizes the Attacker to pivot to a Tandem strategy, which in turn requires the Defenders to learn Focusing. This cycle prevents catastrophic

forgetting of old skills, as any abandoned strategy may be re-exploited by the adversary, compelling agents to maintain a broad and robust repertoire of behaviors.

5.3 Implications for Generalization and Robustness

A primary goal in scaling environments is to produce agents that generalize to unseen situations. Our adversarial framework directly promotes generalization by design. Because the environment is actively hostile and non-stationary, the Defender agents cannot succeed by merely memorizing solutions to a fixed set of problems. They are forced to learn a more general, adaptive policy.

Furthermore, the Attacker functions as an automated red-teaming agent. It is trained to find the edge cases and blind spots in the Defenders' collective policy. This provides a far more efficient and exhaustive method for improving agent robustness than manual testing or curation of test cases. The system inherently generates a stress-testing suite that is always tailored to the agent's present capabilities, hardening the agent against a wide range of potential exploits. This suggests that such adversarial generation frameworks could become a standard component in pipelines for developing safe and reliable autonomous agents.

5.4 Limitations and Future Work

While promising, this work has several limitations that open avenues for future research. The training was conducted on consumer hardware for only 500 episodes; a longer training period would likely reveal even more sophisticated, multi-layered strategies.

A key direction for future work is the integration of Large Language Models (LLMs). An LLM could act as the generative Attacker, tasked with formulating high-level strategic goals (e.g., "create a pincer movement using fast units") which are then translated into specific unit generation actions. This would test the LLM's capacity for strategic reasoning in an interactive setting. Conversely, LLMs could be used by the Defender team for high-level planning or explicit communication, enabling more complex coordination.

Another promising avenue lies in scaling the complexity of the generator itself. The Attacker could be empowered to modify the environment's topology, place obstacles, or even design new types of units with unique, hard-coded behaviors. This concept also relates to tool-use; the Attacker's current action space is compositional, as it combines attributes (tools) to create a unit. Expanding this to a richer set of environmental tools would be a natural next step.

Finally, while we identified emergent strategies qualitatively and quantitatively, a deeper analysis of the learned policies is warranted. Techniques from explainable AI could be used to dissect the agents' decision-making processes, providing clearer insight into the mechanics of the co-evolutionary learning process. Exploring population-based training, where multiple species of Attackers and Defender teams co-evolve, could also lead to a richer and more diverse ecosystem of emergent strategies.

6 Conclusion

We have presented a system for multi-agent learning driven by adversarial procedural generation. By framing the interaction between a generative Attacker and a cooperative Defender team as a nearly zero-sum game, we successfully created a co-evolutionary dynamic that automatically scales environmental complexity in a targeted, adaptive manner. Our implementation demonstrates that this approach fosters the rapid emergence of intelligent, coordinated, and non-trivial strategies in both the generator and the solver agents, a finding supported by both qualitative observation and quantitative analysis.

This work contributes to the growing body of evidence that adversarial self-play is a powerful paradigm for generating complexity without human data. It offers a practical path forward for scaling environments to meet the demands of increasingly general and autonomous agents. By shifting the research focus from the manual design of static content to the architectural design of self-scaling, complexity-generating systems, we can create training methodologies that continuously challenge our agents, pushing them towards greater robustness, strategic depth, and general intelligence.

References

Shuang Ao, Tian He, Ziao Zhou, Weiming Liu, and Hao Sun. Curriculum reinforcement learning via morphology-environment co-evolution. *arXiv preprint arXiv:2309.12529*, 2023.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations (ICLR)*, 2018.

Lucian Buşoniu, Robert Babuška, and Bart De Schutter. A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2): 156–172, 2008. doi: 10.1109/TSMCC.2007.913919.

Christian Schroeder de Witt, Tarun Gupta, Dmytro Makoviichuk, Viktor Makar, Gregory Farquhar, Philip Torr, Shimon Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? In *Deep RL Workshop, NeurIPS 2020*, 2020.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning (ICML)*, pages 1515–1524, 2018.

Linus Gisslén, Andy Eakins, Camilo Gordillo, Joakim Bergdahl, and Konrad Tollmar. Adversarial reinforcement learning for procedural content generation. In *2021 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2021. doi: 10.1109/CoG52621.2021.9619062.

Matthew Guzdial and Mark Riedl. Learning to blend computer game levels. In *Proceedings of the International Conference on the Foundations of Digital Games*, pages 354 – 361, 2016.

Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the 19th National Conference on Artificial Intelligence*, pages 709–715, 2004.

Ahmed Khalifa, Philip Earle, Sebastian Bąk, and Julian Togelius. PCGRL: Procedural content generation via reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, pages 45–51, 2020.

Qinghua Liu, Csaba Szepesvári, and Chi Jin. Sample-efficient reinforcement learning of partially observable markov games. In *Advances in Neural Information Processing Systems*, volume 35, pages 21798–21811, 2022.

Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. Curriculum learning for reinforcement learning agents. *Autonomous Agents and Multi-Agent Systems*, 34(2):32, 2020. doi: 10.1007/s10458-020-09462-y.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 10(3):257–270, 2018. doi: 10.1109/TG.2018.2842628.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4): e0172395, 2017.

Leigh Van Valen. A new evolutionary law. *Evolutionary Theory*, 1(1):1–30, 1973.

Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M. Lucas, Adam Smith, and Sebastian Risi. Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2021. doi: 10.1109/CEC45853.2021.9504783.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (POET): Endlessly generating increasingly complex and diverse learning environments and their solvers. *arXiv preprint arXiv:1901.01753*, 2019.

Xian Wu, Wenbo Guo, Hua Wei, and Xinyu Xing. Adversarial policy training against deep reinforcement learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 233–250, 2021.

Chao Yu, Aravind Velu, Eugene Vinitsky, Yuandong Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative, multi-agent games. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pages 35263–35275, 2022.

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 21008–21019, 2020.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635*, 2019.

A Environment Design Details

This section provides additional details on the game's formal structure, reward functions, and the specific roles of the Defender agents.

A.1 Formalism as a Partially Observable Markov Game

The system is a POMG defined by the tuple $\langle I, S, \{A_i\}_{i \in I}, T, R, \{\Omega_i\}_{i \in I}, O \rangle$. Here, $I = \{A, D_1, D_2, D_3, D_4\}$ is the set of agents; S is the global state space containing the positions, health, and energy of all agents and units; A_i is the action space for agent i; $T(s'|s, \vec{a})$ is the state transition function; $R_i(s, \vec{a})$ is the reward function, which is nearly zero-sum such that $R_A \approx -\sum_{j=1}^{4} R_{D_j}$; Ω_i is the observation space for agent i; and $O(o_i|s, a_i)$ is the observation function. Each Defender observes its own state, the state of other defenders, and nearby units, but not the Attacker's energy. The Attacker observes the full game state.

A.2 Reward Structure

The reward functions are designed to incentivize the core objectives of each side.

- **For the Defenders**, the reward signal includes a large negative reward for losing ($R_{loss} = -1.0$), a small positive reward for each tick survived ($R_{tick} = +0.001$), and a shaping reward for destroying an enemy unit ($R_{kill} = +0.05$).
- **For the Attacker**, the signal is symmetrical: a large positive reward for winning ($R_{win} = +1.0$), a small penalty per tick ($R_{tick} = -0.001$), and a shaping penalty for attempting to spawn a unit with insufficient energy ($R_{fail} = -0.03$).

This structure creates a strong competitive pressure, driving both sides to improve.

A.3 Defender Role Specifications

Each of the four Defender agents is assigned a unique role with distinct statistics and a powerful special ability, encouraging strategic differentiation within the cooperative team. The details are specified in Table 5.

B Implementation Details

This section describes the technical implementation, including the agent architectures and the hyperparameters used for training.

B.1 Training Environment and Agent Architectures

We implemented our system using the Unity game engine and the ML-Agents Toolkit. Both agent types use a Multi-Layer Perceptron (MLP) architecture with two hidden layers of 128 neurons each using the ReLU activation function. The choice of a simple architecture was deliberate to emphasize that emergent complexity arises from environmental interaction rather than from an overly complex model.

Table 5: Detailed Defender role specifications, including damage type, base statistics, and unique special abilities.

Role	Damage Type	Base Statistics	Special Ability (Cost: 200 Energy)
Mage (Blue)	Magic	Low Phys. Def. High Magic Def.	***Debuff Enemies***: Removes all physical and magic defense from all active enemy units.
Healer (Green)	Magic	Low Overall Stats	***Total Party Heal***: Performs a large heal on all four friendly defenders.
Tank (Black)	Physical	High Phys. Def. Low Magic Def.	***Cannon***: Deals massive area-of-effect physical damage to the densest cluster of enemies.
Sharpshooter (Red)	Physical	High Phys. Pen. Low Defenses	***Clear Lane***: Deals very high damage to all enemy units in the Sharpshooter's current lane.

B.1.1 Defender Model

The Defender model uses a 126-dimensional input observation vector, which includes the agent's own status (energy, health, position, role), the status of the other three defenders, and the attributes of up to 16 nearby enemy units. The output is a 6-node layer for the discrete action space, producing a probability distribution via a softmax function.

B.1.2 Attacker (Generator) Model

The Attacker model takes a 254-dimensional input observation vector, containing its own status (energy and max energy) and the full status of all defenders and up to 16 active units. Its complex action is modeled with a multi-branched output consisting of 13 separate heads, one for each unit parameter detailed in Table 2. A softmax is applied to each head independently, allowing the agent to learn a joint distribution over all unit parameters.

B.2 Training Algorithm and Hyperparameters

Both the Defender team and the Attacker are trained simultaneously using Proximal Policy Optimization (PPO) [Schulman et al., 2017]. PPO is an on-policy, actor-critic algorithm known for its stability, making it a strong choice for this complex multi-agent setting [Yu et al., 2022]. The four Defender agents are trained using a shared policy to encourage the development of cooperative strategies. The policies of the Attacker and Defenders are updated concurrently. 100 training runs were conducted for 500 episodes on a consumer-grade laptop (Intel Core i5-1035G7 CPU). An episode corresponds to a single game ending in a Defender loss. Key hyperparameters are listed in Table 6.

Table 6: Training Hyperparameters for PPO.

Hyperparameter	Value
Learning Rate (α)	3.0×10^{-4}
Batch Size	128
PPO Epsilon (ϵ)	0.2
Entropy Bonus (β)	5.0×10^{-4}

C   Emergent Strategy Details

This section provides the specific definitions used to quantify the emergent strategies discussed in the main paper, along with visualizations of these strategies in action.
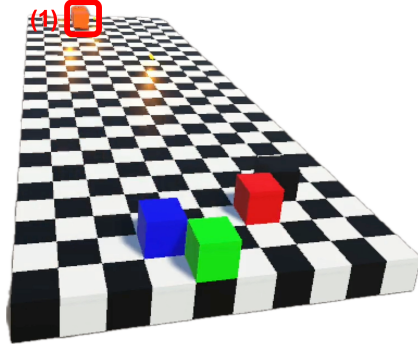
C.1   Strategy Definitions

To ground our qualitative observations, we quantified the frequency of emergent strategies by defining four signature behaviors. These definitions, provided in Table 7, were used to generate the statistics in Table 3.

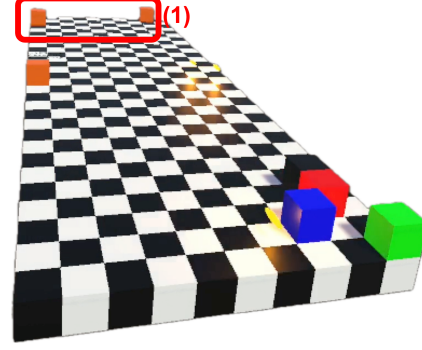Table 7: Definitions used for quantifying emergent strategies.

Strategy	Quantifiable Definition
Cooperative Spreading	Triggered when no two defenders occupy the same lane for at least 5 consecutive timesteps.
Cooperative Focusing	Triggered when at least three defenders occupy the same lane for at least 2 consecutive timesteps.
Flanking	Triggered when the Attacker spawns units on both the far-left lanes (0 or 1) and far-right lanes (8 or 9) within a 2-timestep window.
Tandem	Triggered when the Attacker spawns a unit into a lane that already contains a unit spawned on the previous timestep.

C.2   Visualizations of Emergent Strategies

Figures 3 and 4 provide visual examples of the key strategies that emerged during co-evolutionary training.
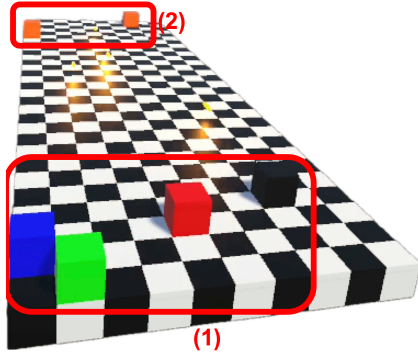
(a) The Tandem Strategy. The attacker generates a durable unit followed immediately by a high-damage unit (1) in the same lane.
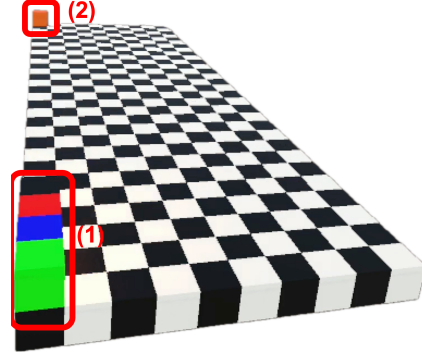
(b) The Flanking Strategy. The attacker generates two threats (1) on opposite sides of the board simultaneously.

Figure 3: Examples of emergent adversarial strategies from the generative Attacker agent.

(a) Cooperative Spreading. Defenders (1) position themselves in different lanes to counter multiple, spread-out units (2).

(b) Cooperative Focusing. All defenders (1) converge on a single lane to focus fire on a high-priority target (2).

Figure 4: Examples of emergent cooperative strategies from the Defender team.

14