

MIT Open Access Articles

Evolutionary and Coevolutionary Multi-Agent Design Choices and Dynamics

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

Citation: Erik Hemberg, Stephen Moskal, Una-May O'Reilly, Eric Liu, and Lucille Fuller. 2025. Evolutionary and Coevolutionary Multi-Agent Design Choices and Dynamics. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '25 Companion). Association for Computing Machinery, New York, NY, USA, 559–562.

As Published: <https://doi.org/10.1145/3712255.3726712>

Publisher: ACM|Genetic and Evolutionary Computation Conference

Persistent URL: <https://hdl.handle.net/1721.1/162637>

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Terms of Use: Article is made available in accordance with the publisher's policy and may be subject to US copyright law. Please refer to the publisher's site for terms of use.



Evolutionary and Coevolutionary Multi-Agent Design Choices and Dynamics

Erik Hemberg
hembergeri@csail.mit.edu
MIT CSAIL, US

Eric Liu
eliu3765@mit.edu
MIT CSAIL, US

Lucille Fuller
lucifull@mit.edu
MIT CSAIL, US

Stephen Moskal
smoskal@mit.edu
MIT CSAIL, US

Una-May O'Reilly
unamay@csail.mit.edu
MIT CSAIL, US

ABSTRACT

We investigate two representation alternatives for the controllers of teams of cyber agents. We combine these controller representations with different evolutionary algorithms, one of which introduces a novel LLM-supported mutation operator. Using a cyber security scenario, we evaluate agent learning when one side is trained to compete against a side that does not evolve and when two sides coevolve with each other. This allows us to quantify the relative merits and tradeoffs of representation and algorithm combinations in terms of team performance. The scenario also allows us to compare the performance impact and dynamics of coevolution versus evolution under different combinations. Across the algorithms and representations, we observe that coevolution reduces the performance highs and lows of both sides while it induces fluctuations on both sides. In contrast, when only one-side is optimized, performance peaks are higher and is more sustained than when both sides are optimized with coevolution.

CCS CONCEPTS

• **Theory of computation** → **Evolutionary algorithms**.

KEYWORDS

coevolution, competitive, evolutionary algorithms

ACM Reference Format:

Erik Hemberg, Eric Liu, Lucille Fuller, Stephen Moskal, and Una-May O'Reilly. 2025. Evolutionary and Coevolutionary Multi-Agent Design Choices and Dynamics. In *Genetic and Evolutionary Computation Conference (GECCO '25 Companion)*, July 14–18, 2025, Malaga, Spain. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3712255.3726712>

1 INTRODUCTION

With larger scale and closer-to-reality scenarios, complex challenges aim to push the limits of autonomous agent training. The CybORG gym, an engine that models cyber networks, threats, and security response, has supported several such challenges [16]. A recent example is the Cyber Cage Challenge 4 (CCC4).

The CCC4 scenario poses the fundamental problem of learning a sequential decision controller for an agent or team of agents. The agent's controller, with only partial observability must decide two things, one simulation step at a time: the host on which it will target its next action, and what that action will be. Additionally, the multiple agents must work in coordination while being positioned in different zones. Their actions realistically model the defender's preventative, monitoring and repairing options. CyBORG provides easy performance comparison among controllers and teams with a point-based system of rewards for different accomplishments.

While CCC4 requires multi-agent training, this training is confined to blue teams. Because we are interested in training autonomous red and blue cyber agents, we extend CCC4 in two ways. Red teams can also be trained and, second, more information about the scenario is made accessible through new observation functions.

This leads to the question of how to best train red and blue teams when they compete against each other. To start, we consider controller design in terms of representation and algorithm choices. These are critical to agent training and agent performance. A designer must decide what part of the solution, in this case the agent's controller, to hard code versus what part they ask the Evolutionary Algorithm (EA) to optimize for them. They can opt for largely predetermined solution logic that sets up a smaller and simpler search space for optimization. Or they can hard code only basic solution logic and provide the EA with powerful logic elements that must be composed into a controller. This implies the EA must optimize in a more complex and larger search space. Choosing between these two extremes or finding a design choice between them is an open question in the challenge setting we have created. To address this, we explore agent controller representations and their variants which have different search space size and expressivity tradeoffs. Our research question is: What is the best combination of algorithm and representation that achieves the best performing red and blue teams?

Our interest in competing cyber agents that take offensive and defensive sides motivates a related interest in adversarial dynamics. Defenses in cyber systems are mostly optimized to repel specific attackers whose activities are well characterized, see, for example, the CCC4 Challenge itself. We seek optimizations that assume attackers will evolve and adapt counter measures to defensive measures. Though two-sided competition and optimization are more challenging in complexity, modeling them is important to better understand the dynamics [10]. Our second research question centers on this model of adversarial dynamics. How do team performance

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
GECCO '25 Companion, July 14–18, 2025, Malaga, Spain
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1464-1/2025/07.
<https://doi.org/10.1145/3712255.3726712>

levels change over time as the two sides adapt while competing? Does peak performance of either side diminish when both sides co-optimize versus when they are optimized independently? Can we expect to see more or less rewards going to each team? To answer this second set of research questions, our baseline is when one side (red or blue) evolves and optimizes with combinations of representations and evolutionary algorithms, while the other (blue or red respectively) does not. We then compare baseline performance with the performance of our combinations of representations and coevolutionary algorithms.

We introduce a version of GE that incorporates a large language model as part of its mutation operator. We compare this algorithm to GE in both one-side learning (with 1 GE algorithms) and two-sided learning (2 GE algorithms connected by competition) and conduct a small study into the LLM supported operator. We observe that the LLM operator is very sensitive to the LLM-model and prompt used. In addition, given the action selection matrix representation, we compare the performance under continuous or discrete parameter optimization. We find that the discrete parameter optimization is a better choice since it works for both cases and is a better for red.

2 RELATED WORK

Table 1 summarizes work related to the agent training in a number of cyber security oriented gyms, including CybORG, and CyberBattleSim. Closely related to this work is [8] where neuroevolution was used for to evolve Deep Reinforcement Learning agents in a CybORG environment. As well, competitive coevolutionary agent-learning for cybersecurity was investigated in the CyberBattleSim environment [14]. Here the investigation was a comparison of different algorithms that used the same agent representation. Finally, early work on a coevolutionary agent-based network defense used a lightweight event system (CANDLES) [13] represented attack and defense agents with rules.

We note that there exists a gap in the related work regarding studying adversarial dynamics and exploring different agent representations and evolutionary computation methods.

3 REPRESENTATION CHOICES AND ALGORITHMS

3.1 Representation 1: Action Selection Matrix

An agent controller is a matrix of n rows, one per state and m columns, one per action. Each cell (i, j) encodes the probability of choosing an action j in state i .

Optimization of Continuous Probabilities with an ES: When the representation of each action probability is a floating point, the probabilities are optimized by an ES [1]. Our ES implementation uses uniform random initialization of solutions. For variation, we use Gaussian mutation.

Optimization of Discretized Probabilities with a GA: Per CCC4 the representation of each action probability is discretized to four choices: (0.0, 0.25, 0.5, 0.75). and a Genetic Algorithm which uses integer vectors [5] is used for optimization. Our GA implementation uses uniform random initialization of solutions. For variation, we use random integer mutation and one point crossover. Evolved values are normalized across all actions for each state.

3.2 Representation 2: Context Free Grammars that Express a Block of Python Code

In this representation there are no agent states. Instead, a controller is some Python code block (in some cases a function), structured to be executable in the CybORG environment. A context free grammar $G \in \mathcal{G}$ expresses the controller code. It includes elements that do not evolve but that are necessary for syntax, such as the block's function header, return values and reserved words such as "if". The start symbol is statements and the terminals are actions, targets, and constants. Non-terminals include the test and true branch elements of an if-statement and the observation functions that require an observation argument and return a count of the observed item.

3.2.1 Algorithms for Grammars. An integer vector can guide the generation (rewriting) of grammar productions. We use Grammatical Evolution (GE) to optimize this vector [2, 12]. Our GE implementation uses uniform random initialization. For variation, we use random integer mutation and one point crossover. The decoding of the population uses a context free grammar. If the decoding is invalid a new random individual is generated.

Our GE-LLM implementation uses uniform random initialization of solutions with a grammar as in GE. For variation, however we use an LLM-supported mutation. The mutation operator works on the code block, not the genome. The code block and other contextual information and a mutation task description are sent to a LLM. The LLM is expected to return a mutated block of code.

A Competitive Coevolutionary algorithm (CCA) has two competing populations with conflicting objectives engaged in a mini-max search. One way to describe a CCA is to note that it is really two EAs that run independently except for a coupling at their fitness function evaluation steps. At that step, one member from each EA's population is selected and competed against a member from the other EA's population.

4 EXPERIMENTS

4.1 Setup

In CCC4 we add two functions that extract from the red agent's observations dictionary. They are respectively the number of hosts it has discovered, and the number of sessions in which it has root access. These are updated after every action. We support red agent selection of where their action will execute by maintaining a list of known hosts, ordered by when they are discovered. The host acquired by the oldest or newest access can be selected or the selection can be random. We also add a blue agent to act as an (non-learning) adversary to red. It uses a finite state machine to move from state to state, using the provided set of blue actions, conditioned on success of an action.

The experimental settings are shown in Table 2.

4.1.1 Experimental Design. We investigate the agent representation, solution representation, algorithm and adversary learning combinations shown in Table 3.

4.2 Results & Analysis

We report the mean fitness of the population for each generation over multiple trials. In addition, the best reward for the blue team

Table 1: Overview of work at intersection of learning agents with coevolutionary algorithms. Columns show the agent environment (Env.), agent representation (A. repr.), evolutionary computation method (EC), and learning dynamics (Dyn.).

| Ath. | Title | Env. | Agent Repr. | EC? | Coev? |
|------------------------------------|--|------------------|-------------------------|-----|-------|
| <i>Cyber Security Environments</i> | | | | | |
| Us | Agent representation and heuristics in CybORG | CybORG CAGE-4 | FSM, Rules | Y | Y |
| [6] | Optimal Defender Strategies for CAGE-2 using Causal Modeling and Tree Search | CybORG CAGE-2 | Structural Causal Model | N | N |
| [8] | Neuroevolution for Autonomous Cyber Defense | CybORG | DRL | Y | N |
| [14] | Adversarial agent-learning for cybersecurity: a comparison of algorithms | CyberBattleSim | DRL | Y | Y |
| [13] | Coevolutionary Agent-based Network Defense Lightweight Event System | CANDLES | Rules | Y | Y |
| [15] | Initiating a Moving Target Network Defense with a Real-time Neuro-evolutionary Detector | DNN | NEAT | Y | N |
| [11] | Evolving Assembly Code in an Adversarial Environment | Code Guru | Assembly | Y | Y |
| [17] | Offline Reinforcement Learning for Autonomous Cyber Defense Agents | CyberVAN | Rules | N | N |
| <i>Theoretical Studies</i> | | | | | |
| [4] | Hardening Active Directory Graphs via Evolutionary Diversity Optimization based Policies | Stackleberg Game | DRL | Y | Y |
| [3] | A Self-adaptive Coevolutionary Algorithm | Defend-It | Discrete | Y | Y |
| <i>Other Environments</i> | | | | | |
| [7] | Competitive coevolution for defense and security: Elo-based similar-strength opponent sampling | Predator-Prey | Y | Y | Y |
| [9] | Emergent Tangled Graph Representations for Atari Game Playing Agents | Atari Games | Code | Y | N |

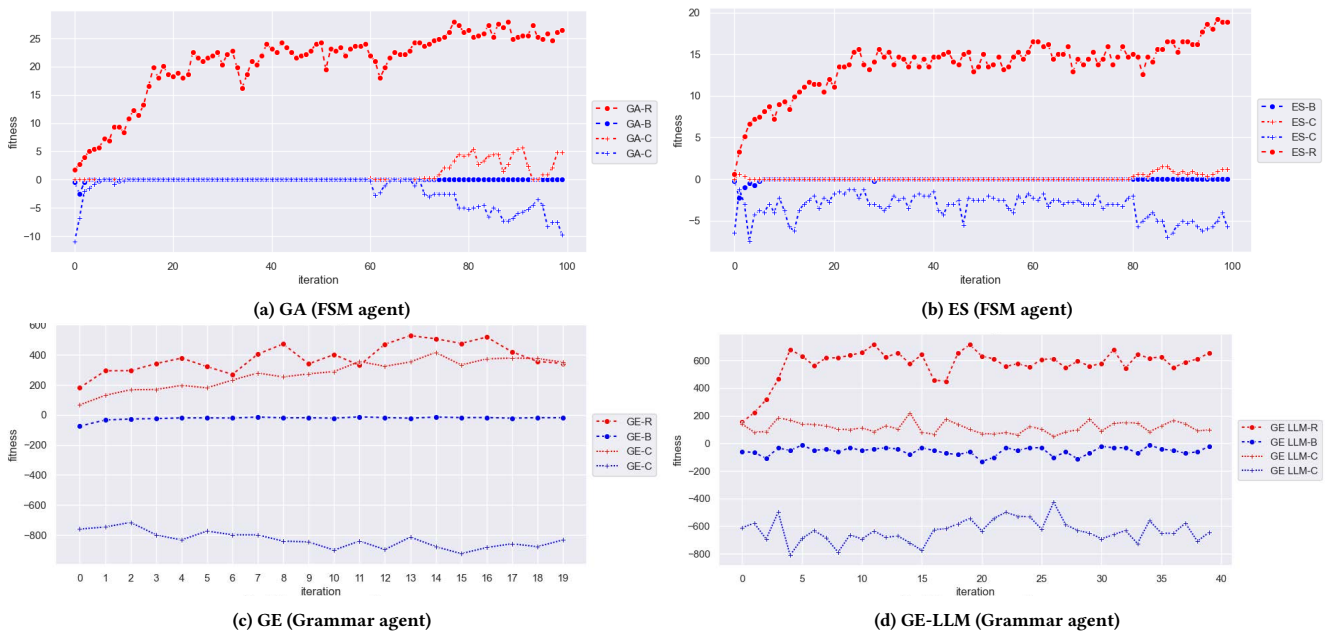


Figure 1: Agent evolution in CCC4. X-axis it the generation. Y-axis mean population fitness (reward). Lines show one-sided or two-sided evolution.

is 0, and best for the red team is scenario dependent. In Figure 1 we show the evolution of the rewards for different algorithms (and agent representations). We observe that coevolution consistently dampens the reward highs and lows of both sides while it induces fluctuations in both sides. In contrast, when only one-side is optimized, reward peaks higher and is more sustained than when both sides are optimized with coevolution. Note that the fitness GE are order of magnitude higher than the FSM. This could indicate a limit in the FSM representation and a benefit of the code rule representation.

We consistently observe that the fitness (reward) from the two-sided learning of dynamically evolved solutions to be lower than the one-sided learning of statically evolved solutions. The representations and algorithms all have different properties and biases. We observed key properties of representation size, search space size and domain knowledge. The fixed size representations and mappings

of ES and GA are often suitable for fixed size problems, e.g. Finite State Machines. In addition, the ES search space size is larger than the GA search space. The GE representations are variable length which can be more suitable to variable length representations like code. The variable length grammar search space of code depends on the grammar size. This is often larger than the GA search space. Finally, the fixed mapping of GA and ES can require less domain knowledge than the grammar used in GE.

4.3 Conclusion

We investigate evolutionary algorithms for training autonomous agents in CybORG’s cyber security simulation. By extending the CAGE 4 Challenge Scenario to include red agent training, we find co-evolved teams achieve lower rewards than those trained against fixed adversaries. Our exploration of FSM and grammar-based controllers reveals that discrete representations work best for FSM,

Table 2: Experimental setup.

| Name | Value |
|-------------------------------|-----------------------|
| CybORG | |
| Steps | 75 |
| Repetitions | 2 |
| Red team agents | 6 |
| Blue team agents | 5 |
| Fixed red agent | FSM |
| Fixed blue agent | FSM |
| Evolutionary Algorithm | |
| Trials | 6 |
| Population Size | 10 |
| Elite size | 1 |
| Tournament Size | 2 |
| Generations | ≥ 20 |
| Crossover probability | 0.5 |
| Mutation probability | 0.5 |
| Red team length | 180 FSM, 1000 GE |
| Blue team length | 180 FSM, 1000 GE |
| Coevolution solution concept | Mean Expected Utility |

Table 3: Experimental design. A. Repr. is agent representation. S. Repr. is the solution representation. Alg. is the algorithm. Adv. is the fixed adversary, where coev means both are learning.

| Name | A. Repr. | S. Repr. | Alg. | Adv. |
|--------------------------|----------|----------|---------|------|
| Static Adversary | | | | |
| ES-B | FSM | Cont. | ES | Red |
| ES-R | FSM | Cont. | ES | Blue |
| GA-B | FSM | Discrete | GA | Red |
| GA-R | FSM | Discrete | GA | Blue |
| GE-B | Code | Grammar | GE | Red |
| GE-R | Code | Grammar | GE | Blue |
| GE-LLM-B | Code | Grammar | GE-LLM | Red |
| GE-LLM-R | Code | Grammar | GE-LLM | Blue |
| Dynamic Adversary | | | | |
| ES-C | FSM | Cont. | ES | Coev |
| GA-C | GA | Discrete | GA | Coev |
| GE-C | Code | Grammar | GE | Coev |
| GE-LLM-C | Code | Grammar | GE, LLM | Coev |

while grammar-based agents achieve highest training fitness. Grammar expressiveness significantly impacts performance, with larger grammars performing well given appropriate terminals. LLM-based grammar operators prove highly sensitive to prompt design and model selection. Future work will examine out-of-sample performance, parameter sensitivity, grammar variations, and the transferability of rule agents across different scenarios and emulation environments.

ACKNOWLEDGMENTS

We acknowledge funding for this work under US Government Contract #FA8075-18-D-0008.

REFERENCES

- [1] Thomas Back. 1996. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.
- [2] Leonardo Lucio Custode, Chiara Camilla Migliore Rambaldi, Marco Roveri, and Giovanni Iacca. 2024. Comparing large language models and grammatical evolution for code generation. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1830–1837.
- [3] Mario Alejandro Hevia Fajardo, Erik Hemberg, Jamal Toutouh, Una-May O'Reilly, and P. Lehre. 2024. A Self-adaptive Coevolutionary Algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference (2024)*. <https://api.semanticscholar.org/CorpusID:271065215>
- [4] Diksha Goel, Max Ward, Aneta Neumann, Frank Neumann, Hung Nguyen, and Mingyu Guo. 2024. Hardening Active Directory Graphs via Evolutionary Diversity Optimization based Policies. *ACM Transactions on Evolutionary Learning (2024)*.
- [5] David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- [6] Kim Hammar, Neil Dhir, and Rolf Stadler. 2024. Optimal Defender Strategies for CAGE-2 using Causal Modeling and Tree Search. *arXiv preprint arXiv:2407.11070 (2024)*.
- [7] Sean N. Harris and Daniel R. Tauritz. 2021. Competitive coevolution for defense and security: Elo-based similar-strength opponent sampling. *Proceedings of the Genetic and Evolutionary Computation Conference Companion (2021)*. <https://api.semanticscholar.org/CorpusID:235770343>
- [8] Kade Heckel. 2023. Neuroevolution for Autonomous Cyber Defense. *Proceedings of the Companion Conference on Genetic and Evolutionary Computation (2023)*. <https://api.semanticscholar.org/CorpusID:260119489>
- [9] Stephen Kelly and Malcolm I. Heywood. 2017. Emergent Tangled Graph Representations for Atari Game Playing Agents. In *European Conference on Genetic Programming*. <https://api.semanticscholar.org/CorpusID:26568610>
- [10] Krzysztof Krawiec and Malcolm Heywood. 2016. Solving Complex Problems with Coevolutionary Algorithms. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*. ACM, 687–713.
- [11] Irina Maliukov, Gera Weiss, Oded Margalit, and Achiya Elyasaf. 2024. Evolving Assembly Code in an Adversarial Environment. *ArXiv abs/2403.19489 (2024)*. <https://api.semanticscholar.org/CorpusID:268732824>
- [12] Michael O'Neill and Conor Ryan. 2001. Grammatical evolution. *IEEE Transactions on Evolutionary Computation* 5, 4 (2001), 349–358.
- [13] George Rush, Daniel R. Tauritz, and Alexander D. Kent. 2015. Coevolutionary Agent-based Network Defense Lightweight Event System (CANDLES). *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation (2015)*. <https://api.semanticscholar.org/CorpusID:15753509>
- [14] Alexander Shashkov, Erik Hemberg, Miguel Tulla, and Una-May O'Reilly. 2023. Adversarial agent-learning for cybersecurity: a comparison of algorithms. *The Knowledge Engineering Review* 38 (2023). <https://api.semanticscholar.org/CorpusID:257354029>
- [15] Robert J. Smith, Ayse Nur Zincir-Heywood, Malcolm I. Heywood, and John T. Jacobs. 2016. Initiating a Moving Target Network Defense with a Real-time Neuro-evolutionary Detector. *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion (2016)*. <https://api.semanticscholar.org/CorpusID:16359957>
- [16] Maxwell Standen, Martin Lucas, David Bowman, Toby J. Richer, Junae Kim, and Damian A. Marriott. 2021. CybORG: A Gym for the Development of Autonomous Cyber Agents. *ArXiv abs/2108.09118 (2021)*. <https://api.semanticscholar.org/CorpusID:237259783>
- [17] Alexander Wei, David A. Bierbrauer, Emily A. Nack, John Pavlik, and Nathaniel Bastian. 2024. Offline Reinforcement Learning for Autonomous Cyber Defense Agents. *2024 Winter Simulation Conference (WSC) (2024)*, 1978–1989. <https://api.semanticscholar.org/CorpusID:275773130>