A coevolutionary approach to deep multi-agent reinforcement learning
Klijn, Daan; Eiben, A. E.

published in
GECCO 2021
2021

DOI (link to publisher)
10.1145/3449726.3459576

document version
Publisher's PDF, also known as Version of record
document license
Article 25fa Dutch Copyright Act

Link to publication in VU Research Portal

citation for published version (APA)

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:
vuresearchportal.ub@vu.nl

Download date: 15. Oct. 2023
A Coevolutionary Approach to Deep Multi-Agent Reinforcement Learning

Daan Klijn  
Vrije Universiteit Amsterdam  
d.s.klijn@student.vu.nl

A.E. Eiben  
Vrije Universiteit Amsterdam  
a.e.eiben@vu.nl

ABSTRACT
Traditionally, Deep Artificial Neural Networks (DNN’s) are trained through gradient descent. Recent research shows that Deep Neuroevolution (DNE) is also capable of evolving multi-million-parameter DNN’s, which proved to be particularly useful in the field of Reinforcement Learning (RL). This is mainly due to its excellent scalability and simplicity compared to the traditional MDP-based RL methods. So far, DNE has only been applied to complex single-agent problems. As evolutionary methods are a natural choice for multi-agent problems, the question arises whether DNE can also be applied in a complex multi-agent setting. In this paper, we describe and validate a new approach based on coevolution. To validate our approach, we benchmark two Deep coevolutionary Algorithms on a range of multi-agent Atari games and compare our results against the results of Ape-X DQN. Our results show that these Deep coevolutionary algorithms (1) can be successfully trained to play various games, (2) outperform Ape-X DQN in some of them, and therefore (3) show that coevolution can be a viable approach to solving complex multi-agent decision-making problems.

CCS CONCEPTS
• Computer systems organization → Neural Networks; • Theory of computation → Evolutionary algorithms.

KEYWORDS
Deep Neuroevolution, Coevolution, Evolution Strategies, Genetic Algorithm, Multi-agent Reinforcement Learning

ACM Reference Format:

1 INTRODUCTION
Recent developments have caused Evolutionary Reinforcement Learning to gain momentum. Salimans et al. [4] showed that the combination of Evolution Strategies (ES) and Neuroevolution is a perfect alternative to the MDP-based approaches that are typically used in RL. Their ES was capable of evolving DNN’s that achieved state-of-the-art performance on some of the benchmarks while maintaining near-linear scalability. It also proved to be invariant to action frequency and is resistant to delayed rewards and long horizons. Such et al. [5] took the ES as an inspiration and showed that a similar thing is possible using a simple Genetic Algorithm (GA). While ES relies on approximations of the gradients, GA’s are truly gradient-free. It is therefore remarkable to see that this approach is able to train networks with millions of parameters. These advances demonstrated the far-reaching capabilities of DNE and have laid the foundations for new breakthroughs in RL.

While most RL problems consist of an agent solely interacting with an environment, a subfield called Multi-Agent Reinforcement Learning (MARL) encompasses the problems where multiple agents are present. The goal of MARL is to develop agents that can successfully cooperate or compete with other agents. Similarly, Evolutionary Computation has a class of algorithms that is called Coevolutionary Algorithms. Compared to the single-agent flavor of these problems, multi-agent problems tend to be more challenging as they include another uncertainty, namely, the other agents. Due to this, the measured performance of an agent is subjective, as it depends on the other agents’ performance. While MARL has been proven to be an effective solution for complex Multi-Agent problems, coevolution has never been scaled to high-complexity problems.

The main goal of this paper is to investigate whether the combination of DNE and Coevolution could open up a new door in the field of MARL. To this end, we develop two Deep coevolutionary algorithms, one based on the ES from Salimans et al. [4] and another based on the GA from Such et al. [5]. To assess our approach’s viability, we train these algorithms on eight different games from the PettingZoo benchmark [7] and compare the evolved agents to the ones delivered by Ape-X DQN.

2 METHODOLOGY
In this work, we combine DNE with a coevolutionary approach. We apply this approach to two successful DNE algorithms; the ES from Salimans et al. [4] and the GA from Such et al. [5]. To prove that even a simple coevolutionary setup works in combination with DNE, we only transform the original two algorithms to their coevolutionary counterparts and limit the number of features we add to them. Below we shortly describe the specifications of the forged coevolutionary algorithms.

2.1 Coevolutionary Evolution Strategies
Salimans et al. [4] proposed an ES that is capable of evolving DNN’s with millions of parameters and showed that it could solve various complex Reinforcement Learning problems. The ES proposed here is more scalable a variant of Natural Evolution Strategies [8]. We
altered this ES in such a way that it can be applied to multi-agent problems. While the original ES evaluates each of the mutations \( \theta + \varepsilon \) using the fitness function, we now need to consider that we also need to provide individuals to evaluate against. We have tried various approaches and discovered that evaluating each individual against the \( k \) parents of the previous populations (i.e., \( \theta_t \) to \( \theta_{t-k} \)) resulted in the most stable convergence. Sometimes, using only the previous parent was enough. The \( k \) fitnesses are then averaged for each individual and used in combination with the update rule of the original ES.

2.2 Coevolutionary Genetic Algorithm

Based on the work of Salimans et al. [4], Such et al. [5] showed that a GA is also capable of evolving similar DNN’s. The GA they proposed consists of a single population, only uses the mutation operator and performs fitness-based truncation. Our coevolutionary variant is based on this GA and will stay as close to the original design as possible. Again, we replace the original fitness evaluation with an evaluator-based fitness evaluation. For the GA, we will use a Hall of Fame (HoF) that stores the best individuals of previous generations. To evaluate individuals, we select the last \( k \) elites from the HoF. This makes sure that the fitness approximation is reliable and that historical traits are not lost. Note that for both the Coevolutionary ES (CoES) and the Coevolutionary GA (CoGA), \( k \) should not be too high as that will increase the number of evaluations per individual, which can increase runtime if the fitness function is expensive to compute.

3 EXPERIMENTS

Numerous RL works benchmark algorithms using the Atari games, as they require the agent to perform a range of complex tasks that can be compared to the difficulties of some real-life tasks. While Salimans et al. [4] and Such et al. [5] use single-player Atari games to benchmark their algorithms, we will use a diverse set of multi-player Atari games from the PettingZoo benchmark [7]. Below a short overview is given of our experimental setup. Next to that, the code that was used for these experiments can be found on GitHub 1

- **Preprocessing** - We process the game frames as proposed by Mnih et al. [1]. Besides that, we also add a form of agent indication [1] as it is essential for each agent to know whether it is player one or player two.
- **Network architecture** - We use a DNN that is similar to the large DQN that was used by Mnih et al. [2]. Furthermore, we use virtual batch normalization (VBN) [3] to ensure that the evolved agents are diverse enough. Without VBN, many evolved agents were not diverse and always performed the same actions, regardless of the input.
- **Hyperparameters** - In comparison to the original ES and GA, we used a slightly higher noise standard deviation and a smaller population size. We were able to get the best results using a population size of 200 and standard deviations of 0.005 (GA) and 0.05 (ES). Furthermore, we found that using \( k = 3 \) for the GA and \( k = 1 \) for the ES was enough to ensure accurate evaluations of individuals.

We trained both the CoES and the CoGA on eight different games, whereas we start learning from scratch for each of the games. After 200 million frames, we evaluate the trained agents by playing 50 games against a random agent. The average performance against the random agent of the ES, GA and Ape-X[7] can be found in Table 1. Videos of selected gameplay of both the CoES and CoGA are available 2. Furthermore, a more comprehensive set of results combined with a more thorough analysis can be found in the full version of this paper. 3

<table>
<thead>
<tr>
<th>Game frames</th>
<th>Random</th>
<th>Ape-X DQN</th>
<th>CoES</th>
<th>CoGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training frames</td>
<td>-</td>
<td>80M</td>
<td>200M</td>
<td>200M</td>
</tr>
<tr>
<td>Generations</td>
<td>-</td>
<td>20M</td>
<td>50M</td>
<td>50M</td>
</tr>
<tr>
<td>Wall time (96 cores)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Basketball Pong</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>2.1</td>
</tr>
<tr>
<td>Boxing</td>
<td>0.0</td>
<td>80.0</td>
<td>0.7</td>
<td>1.0</td>
</tr>
<tr>
<td>Combat Plane</td>
<td>0.0</td>
<td>-2.0</td>
<td>2.5</td>
<td>1.8</td>
</tr>
<tr>
<td>Combat Tank</td>
<td>0.0</td>
<td>0.3</td>
<td>0.1</td>
<td>0.4</td>
</tr>
<tr>
<td>Joust</td>
<td>3951.0</td>
<td>3846.0</td>
<td>5362.2</td>
<td>4910.0</td>
</tr>
<tr>
<td>Pong</td>
<td>0.0</td>
<td>20.5</td>
<td>17.6</td>
<td>15.8</td>
</tr>
<tr>
<td>Space war</td>
<td>0.0</td>
<td>1.1</td>
<td>1.8</td>
<td>0.8</td>
</tr>
<tr>
<td>Tennis</td>
<td>-3.6</td>
<td>22.5</td>
<td>6.9</td>
<td>-20.2</td>
</tr>
</tbody>
</table>

Table 1: Performance of Ape-X, CoES and CoGA against a Random Policy for various games. The Ape-X results are derived from [6] and are based on the last 20 evaluations.

Our results demonstrate that the coevolutionary approach can train agents that can play most of the games to some extent while even outperforming Ape-X DQN in several of them. We hypothesize that this approach’s effectiveness might be due to the EA’s being more resistant to the non-stationarity that is introduced by the multi-agent aspect of these problems. Although this work only explores the capabilities of simple coevolutionary algorithms on a small range of benchmarks, we do believe that this might open up a new door in the field of MARL.

REFERENCES


1https://github.com/daanKlijn/neuroevolution/
2https://youtube.com/playlist?list=PLdGO0x9WS6eCCGA1w4jOPLz9ahWSmZEw