Selection of Deep Reinforcement Learning Using a Genetic Algorithm

Yurii Kryvenchuk[1], Dmytro Petrenko[1], Dariusz Cichoń[2], Yuriy Malynovskyy[1]
Tetiana Helzhynska[1]

[1] *Lviv Polytechnic National University, Stepana Bandery Street 12, Lviv, 79013, Ukraine*

[2] *AGH University of Science and Technology, al. Mickiewicza 30, Krakow, 30059, Poland*

Abstract
Neural network models contain many parameters. The selection of these parameters and the selection of the correct model takes a long time. Model developers rely on expert assessment to select models and hyperparameters for them. This paper examines the use of a genetic algorithm as an alternative to the current design process. The genetic algorithm is used to automatically select network hyperparameters and the model of the network itself. This improves the network model and reduces development time. The general model presents an algorithm with input parameters equal to those required to represent possible states and system output parameters sufficient to describe possible actions. The algorithm automatically selects different models for different parameters. It is determined that the algorithm can successfully start working with a low-efficiency model template and show good model performance and adjust the indicators of the number of layers, policy, entropy coefficient, and others. This shows the potential for further application of these algorithms for drone design.

Keywords 1
Artificial Intelligence (AI), reinforcement learning (RL), deep reinforcement learning (DRL), genetic algorithm (GA).

1. Introduction

The application of reinforced learning has grown in popularity during the last few years due to its success in solving complex successive decision-making problems [1]. While the most impressive results have been achieved in classical single-task reinforcement training with a static environment and a fixed reward function, reinforced contextual learning promises to stimulate the next wave of breakthroughs, using similarities between environment and tasks [2].

Reinforced learning (RL) allows agents to learn complex behaviours from interacting with the environment. Combinations of RL paradigms with powerful function approximators, commonly referred to as deep RLs (DRLs), have led to a superhuman performance in various simulated areas [2, 3]. DRL algorithms, despite their outstanding results, suffer from a high sampling complexity. Thus, many studies aim to reduce the complexity of the sample by improving the research behaviours of RL agents in one task.

Recently, a growing number of algorithms for curriculum development have been presented, which empirically demonstrates that curriculum learning (CL) is an appropriate tool for improving the

COLINS-2022: 6th International Conference on Computational Linguistics and Intelligent Systems, May 12–13, 2022, Gliwice, Poland;
EMAIL: yurkokryvenchuk@gmail.com (Yu. Kryvenchuk); skiff120@gmail.com (D. Petrenko); darek4224@tlen.pl (D. Cichoń); inem.news@gmail.com (Y. Malynovskyy); tanyazvarich@ukr.net (T. Helzhynska)
ORCID: 0000-0002-2504-5833 (Yu. Kryvenchuk); 0000-0003-3720-9038 (D. Petrenko); 0000-0003-3720-9038 (D. Petrenko); 0000-0003-4198-1530(D. Cichoń); 0000-0002-7139-5623(Y. Malynovskyy); 0000-0003-3280-5199 (T. Helzhynska)

efficiency of DRL algorithm sampling. However, these algorithms are based on heuristics and concepts that are currently theoretically insufficiently understood, not allowing for significant improvements.
The basic idea of DRL is that an artificial agent can learn by interacting with the environment, much like a biological agent. Using the experience gained, the artificial agent should be able to optimize some of the goals set in the form of cumulative rewards.

This approach, in principle, applies to any type of consistent decision-making problem based on experience. The environment can be stochastic, and the agent can observe only partial information about the current state, observations can be highly dimensional (e.g., frames and time series), the agent can freely gain experience in the environment or, conversely, data may be limited (e.g., lack of access to the accurate simulator or limited data) [1].

2. Related Works

The basic idea of reinforced learning is to teach the agent to interact with the environment in such a way as to achieve the greatest possible success in achieving the goals. Such training is quite similar to the natural way of teaching people and other living beings. The environment itself and the result of decisions made by the agent based on observation of this environment acts as a teacher in such training.

From the beginning of its existence in this world, the brain does not know how to behave in it. However, it sends signals to the organs of movement and receives data from the senses. And based on the data obtained determines whether the previous action led to an approach to the desired result. For example, in the process of learning, the child does not understand the purpose of objects and tries to taste them because it is one of the most trained senses in early childhood. In the case where the found object brings the desired result (pleasure), the pattern of behavior that led to this result is fixed in the brain in the form of a more stable neural connection [1]. Based on the observation of similar learning processes in nature, reinforcement learning algorithms have been developed.

There are two main goals for reinforcement learning algorithms. The first goal is to minimize the number of errors, and therefore to minimize the number of steps and speed up the achievement of goals. The agent learns to analyze the state of the environment before each subsequent action and predict the expected outcome of following probable steps. The second important goal in the work of reinforcement learning algorithms is to maximize the benefits of the actions taken. In this case, the definition of benefits is programmed in advance. This can be, for example, minimizing execution time or finding as much space for actions as possible, etc. [2]. Returning to the example of nature, for humans it may be the release of the hormone of happiness from achieving the goal.
The primary purpose of the study conducted during the work on this article is to try to simulate a combination of two algorithms that exist in wildlife. Namely, the reinforcement learning algorithm, as an algorithm for achieving the goal of each agent, and the genetic algorithm, as an algorithm for selecting those agents of the environment that have shown the best results in this environment.

The task of the genetic algorithm is to optimize the basic parameters of the set of agents based on natural selection. By analogy with living nature, the genetic algorithm in the presented experiment will select the best representatives of its generation and combine the values of their basic model parameters to create the next and more perfect generation, just as living beings are firstborn with certain parameters inherited from their ancestors and depending on the success of adaptation to the environment have the opportunity to further produce offspring. Thus, it is expected that the best individuals from each generation will be selected, and their descendants will be able to become even better adapted to the environment.

3. State of art

The general problem RL is formalized as a stochastic process of discrete time control, where the agent interacts with its environment as follows: the agent starts in a given state in its enviroement $s_0 \in S$, collecting the initial observation $\omega_0 \in \Omega$. At each time step t, the agent must perform the action $a_t \in A$. As shown in figure 1, this follows three consequences:

- The agent is rewarded $r_t \in R$;

- The state passes to $s_{t+1} \in S$;
- The agent receives the observation $\omega_{t+1} \in \Omega$.

This control parameter was first proposed by Bellman [4], and later extended to Barto's teaching [5]. Comprehensive development of RL basics is provided by Sutton and Barto, 2017 [6].

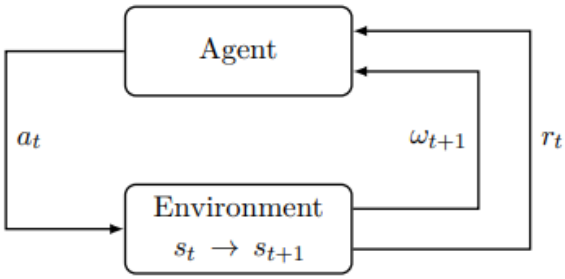

Figure 1: Agent-Environment Interaction in RL

State S is a complete description of the state of the world. Observation O is a partial description of the state in which information may be omitted. When the agent can observe the complete state of the environment, it means that the environment is completely observed. When an agent can only see partial observation, it means that the environment is partially observed. Different environments allow for different types of actions. The totality of all real actions in each environment is often called a space of action. Some environments, such as Atari and Go, have discrete action spaces where only a finite number of moves are available to the agent. Other environments, such as where an agent controls a robot in the physical world, have spaces for continuous action. In continuous spaces, actions are vectors with real values.

A policy is a decision-making rule that an agent uses to choose what action to take. It can be deterministic, and in this case, it is usually denoted by μ: $a_t = \mu(S_t)$. Because politics is essentially the brain of the agent, the word "policy" can often be replaced by "agent," for example, by saying, "Policy tries to maximize reward." DRL deals with parameterized policies: policies which outputs are computational functions that depend on a set of parameters (such as neural network weights and offsets) which can be configured to change behavior using a specific optimization algorithm [20].

The reward function R is critical in reinforcement learning. It is contingent on the current condition of the world, the recent action, and the future state of the world: $r_t = R(s_t, a_t, s_{t+1})$. The agent's goal is to maximize some idea of the total reward for actions, but, it can mean several things. Denoting all these cases by R (τ), it will either be clear from the context which case is meant or it will not matter (because the same equations will apply to all cases).

Knowing the value of a state or state-action pair is frequently useful. The value means the expected reward that begins in this state or state-action pair and then continues to operate according to a certain policy. Value functions are used, one way or another in almost every RL algorithm [7]. The on-policy value function $V^{\pi}(s)$, which gives the expected reward if it starts in the state s and always acts in accordance with the policy π [7]:

$$V^{\pi}(s) = \underset{\tau \sim \pi}{E}[R(\tau)|s_0 = s]$$

The on-policy action-value function $Q^{\pi}(s, a)$, which gives the expected reward if it starts in the s state, performs an arbitrary action a (which may not be from policy), and then forever act following the policy π [8]:

$$Q^{\pi}(s, a) = \underset{\tau \sim \pi}{E}[R(\tau)|s_0 = s, a_0 = a].$$

The optimal value function $V^{*}(s)$, which gives the expected reward if it starts in the s state and always acts following the optimal policy in the environment [8]:

$$V^{*}(s) = \max_{\pi} \underset{\tau \sim \pi}{E}[R(\tau)|s_0 = s]$$

The optimal action-value function $Q^*(s,a)$, which gives the expected reward if it starts in the states, performs arbitrary action a, and then acts forever according to the optimal policy in the environment:

$$Q^*(s,a) = \max_{\pi} \underset{\tau\sim\pi}{E}[R(\tau)|s_0 = s, a_0 = a]$$

There is an essential connection between the optimal action-value function $Q^*(s,a)$ and the action chosen by the optimal policy. By definition, $Q^*(s,a)$ gives the expected return to run in state S, perform (arbitrary) action a, and then continuous action according to the optimal policy. The optimal policy in s will choose the action that maximizes the expected reward from the beginning in s. As a result, with Q^* it can be directly obtained the optimal effect $a^*(s)$, through [8]:

$$a^*(s) = arg\max_{a} Q^*(s,a)$$

3.3. Types of RL algorithms

In fact, it is pretty difficult to make an accurate, comprehensive taxonomy of algorithms in the modern RL space [8], because the modularity of algorithms is poorly represented by a tree-like structure (Figure 2).

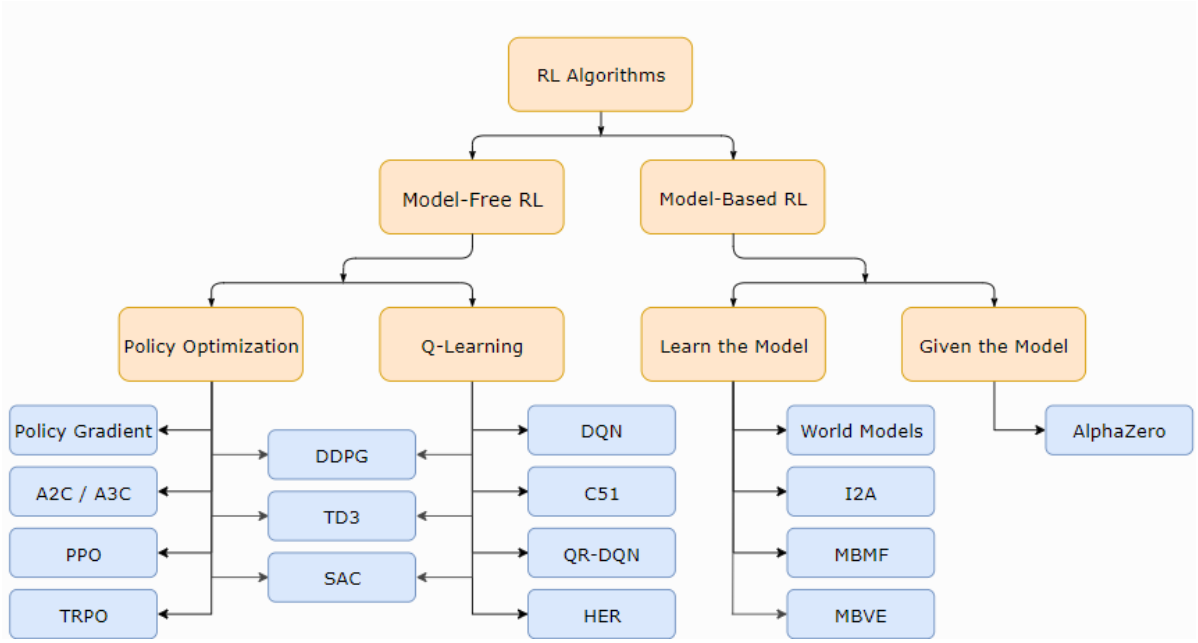

Figure 2: Tree-like structure of RL algorithms

Whether the agent has access (or studies) to the environment model is one of the most critical branching points in the RL algorithm. The environment model means a function that involves the transition of states and rewards. The main disadvantage is that the environment model that corresponds to reality is usually not available to the agent. If the agent wants to use the model in this case, it must study the model solely on experience, which creates several problems. The most serious issue is that the agent can introduce bias into the model. As a result, the agent works well on the studied model but behaves sub-optimally in the natural environment. Learning a model is fundamentally difficult, so even intense effort - the desire to spend a lot of time calculating - may not pay off. Algorithms that use a model are called model-based methods, and those that do not use a model are called non-model. While non-model methods reject the potential for sample efficiency from the model used, they are generally easier to implement and configure [19].

The trade-offs between policy optimization and Q-Learning are that they are fundamental because they directly optimize what is needed. This makes them stable and reliable. On the contrary, Q-learning

methods only indirectly maximize the agent's work, teaching Q_θ to satisfy the self-consistency equation. There are many failure modes for such training, so it is usually less stable [9]. But Q-learning methods have an advantage because they are much more effective when they work because they can reuse data more efficiently than policy optimization methods.

Interpolation between policy optimization and Q-Learning. Ironically, policy optimization and Q-learning are not incompatible (and in some circumstances appear to be equivalent), and there are a few algorithms that live between the two extremes. Algorithms that live in this spectrum are able to find a compromise between the strengths and weaknesses of either side. Examples include DDPG and SAC [10, 11, 17].

3.4. Metaheuristic algorithms

Metaheuristic algorithms, in recent years, have been utilized to address real-world complicated issues in a variety of sectors, such as economics, engineering, politics and management. Intensification and diversification are key elements of the metaheuristic algorithm. A proper balance between these elements is necessary to effectively solve a real problem. Most metaheuristic algorithms are based on the process of biological evolution, swarm behavior and the laws of physics [12]. These algorithms are broadly classified into two categories, namely: a single solution and a metaheuristic algorithm based on the set (Figure 3). Metaheuristic algorithms based on a single solution use a single candidate solution and improve this solution through local search. However, the solution obtained with the help of metaheuristics based on one solution may remain in the local optimum [13].

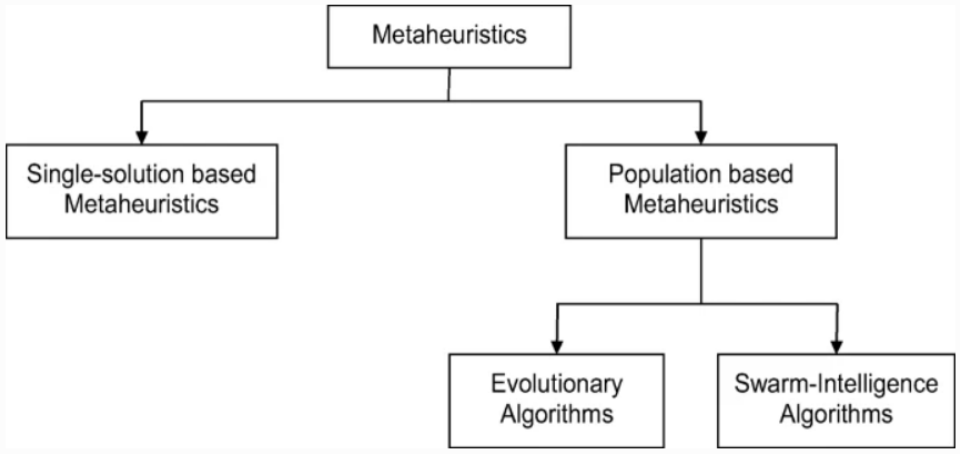

Figure 3: Classification of metaheuristic algorithms

Evolutionary algorithms work on a set or population of solutions and use two mechanisms to find good solutions: selecting mostly high-quality solutions from the set and combining the qualities of two or more solutions with specialized operators to create new solutions. New solutions are reintroduced into the population after recombination, which may require them to meet conditions such as feasibility or minimum quality requirements to replace other (usually low-quality) solutions. Operators used in evolutionary algorithms (selection, recombination, and re-insertion) almost without exception make extensive use of randomness. A mutation operator is also often used, which randomly changes the solution after its recombination. Most evolutionary algorithms repeat the selection, recombination, mutation, and reintroduction phases several times and report the best solution in a population [16].

Among the metaheuristic algorithms, a well-known algorithm is the genetic algorithm (GA), which is inspired by the process of biological evolution. GA mimics Darwin's theory of the survival of the fittest. GA was proposed by J.G. Holland in 1992. The main elements of GA are chromosome representation, selection, and biological operators [15].

GA dynamically changes the search process due to the probabilities of crossover and mutation and achieves the optimal solution. GA can modify encoded genes. GA is capable of evaluating numerous individuals and making several optimum decisions. Therefore, GA has better global search capabilities. Offspring derived from parental chromosome crossover are likely to override the excellent genetic

patterns of parental chromosomes, and the crossover formula is defined as: $R = \frac{G+2\sqrt{g}}{3G}$, where g denotes the number of generations and G denotes the population's total number of evolutionary generations. The equation shows that R changes dynamically and increases with increasing number of evolutionary generations. Individual similarity is quite low in the early stages of GA. To guarantee that the new population does not disrupt the great genetic pattern of individuals, R should be set to a low number. The individual similarity is relatively high at the end of evolution, hence the value of R should be high too [13].

The classical genetic algorithm is of the following shape [13]:

Incoming data:
Population size, n;
Maximum number of iterations, MAX.

Entrance:
The best global solution, Yb.

Beginning:
Creating an initial population of n chromosomes Y, (i = 1,2, ..., n);
Set the iteration counter t = 0;
Calculate the value of the fit of each chromosome;
While (t <MAX)
Select a pair of chromosomes from the initial population based on suitability;
Apply crossover operation to the selected pairs with the probability of crossing
Apply the mutation to the offspring with the probability of mutation;
Replace the old population with the newly created population;
Increase the current iteration of t by 1.
Return the best solution, Yb.
End.

According to the scheme theorem, the original scheme must be replaced by a modified scheme. To preserve population diversity, the new scheme preserves the original population at an early stage of evolution. At the end of evolution, an appropriate scheme will be created to prevent any distortion of the excellent genetic scheme [14, 18].

4. The results of the study

As mentioned above, the main idea of the study is to create a combination of analogues of two types of algorithms that occur in nature - namely, the reinforcement learning algorithm and the genetic algorithm of selection. However, there is more than one reinforcement learning algorithm in machine learning. And it is sometimes difficult to determine which one to choose to best solve the problem. In addition to choosing the actual algorithm, you must also select parameters for this algorithm. In nature, this function was taken over by evolution, which from generation to generation changed the parameters and adapted organisms to survive in their environment. In this experiment, the role of evolution is performed by a genetic algorithm. And the role of creatures that have to adapt to the environment is performed by the agents of each of the reinforcement learning algorithms.

Since it is not possible to cross different types of reinforcement learning algorithms, as this can lead to uncertain results, each of these types of algorithms creates its own initial population of agents (Figure 4). Next, each of these populations will be handled almost separately in the genetic algorithm. Only during selection based on results of the test in the environment, if all agents with a certain machine learning algorithm with reinforcement showed significantly worse results, it will be a chance that none of these agents will not have offspring, and therefore this algorithm will not be presented in the final sample of results.

The results were obtained using Python and such libraries as Tensorflow, Stable baselines, and PyGAD. The flowchart of the developed algorithm is presented in Figure 5.

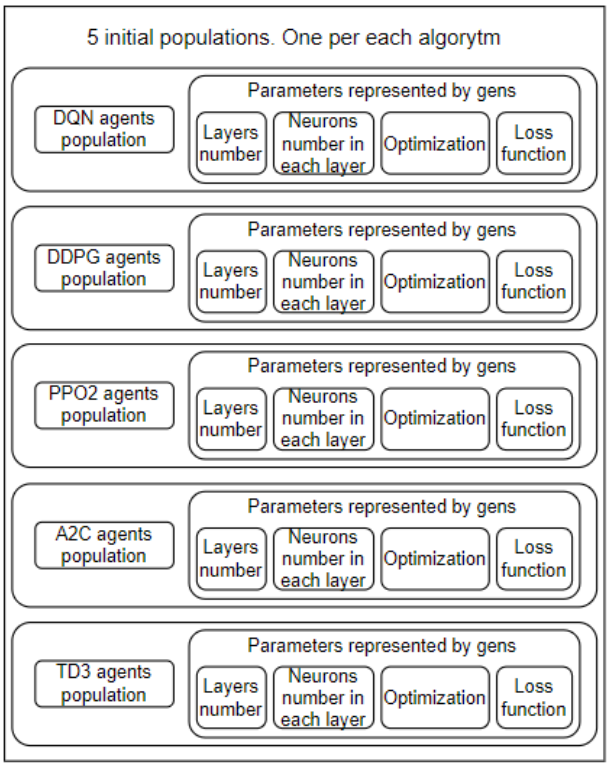

Figure 4: Flowchart of researched algorithm

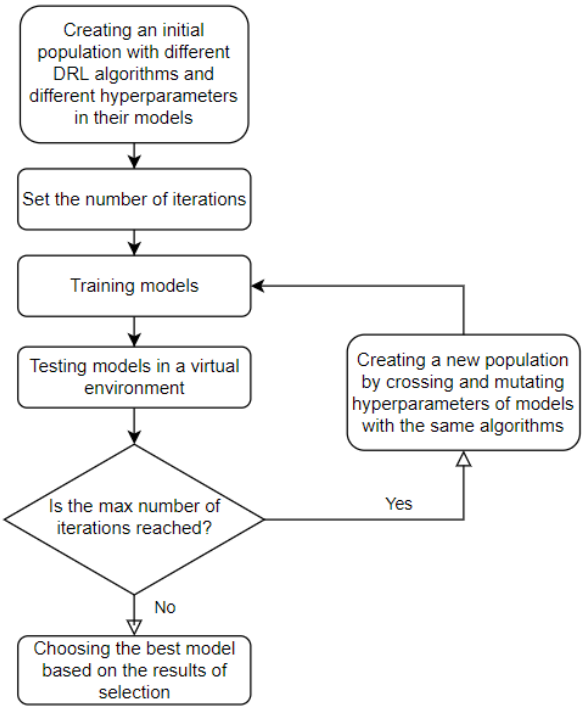

Figure 5: Initial populations for each algorithm

At first, the initial population is created with different DRL algorithms. Chromosomes are presented like hyperparameters of RL algorithms in each individual of the population. This determines the diversity of individuals in the population. During the genetic algorithm cycles, crossings and mutations occur between individuals with the same DRL algorithm in a population, which gradually determines

the selection of the best individuals with the most appropriate DRL algorithm. The genetic algorithm completes its work when the maximum number of generations is reached. It can also complete work when the agent's reward result's specified accuracy is achieved.

The results were tested in a virtual gym environment. When testing the CartPole-v1 agent from 30 samples of the initial population for 500 iterations, the 5 most successful samples were selected. They all turned out to be individuals with the DQN algorithm (Figure 6). The differences between them are only in the internal construction of the models and hyperparameters to them.

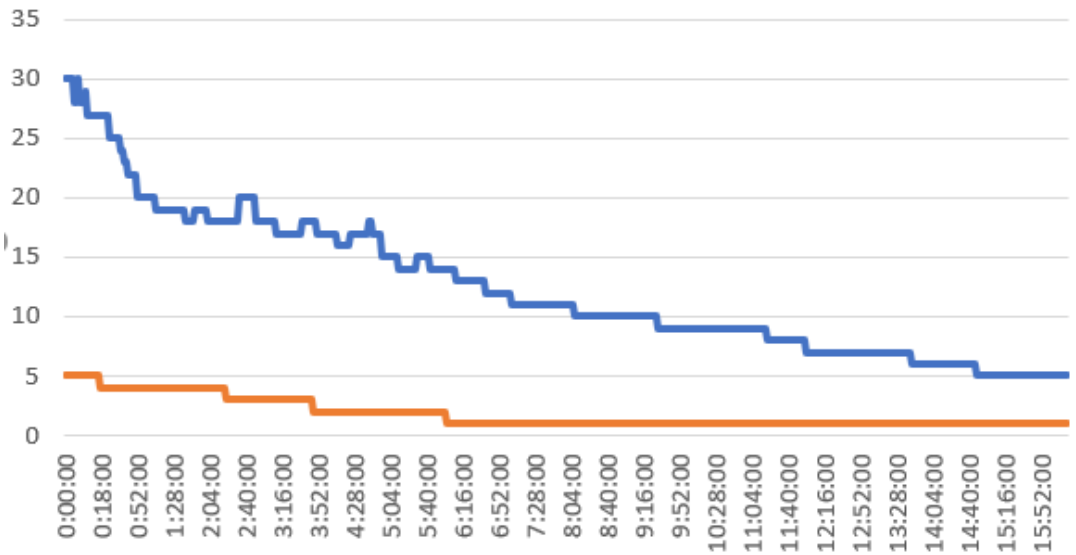

Figure 6: Selecting the RL algorithm for the CartPole-v1 agent

The dependence of the time the agent receives a positive reward on the sample generation is shown in Figure 7.

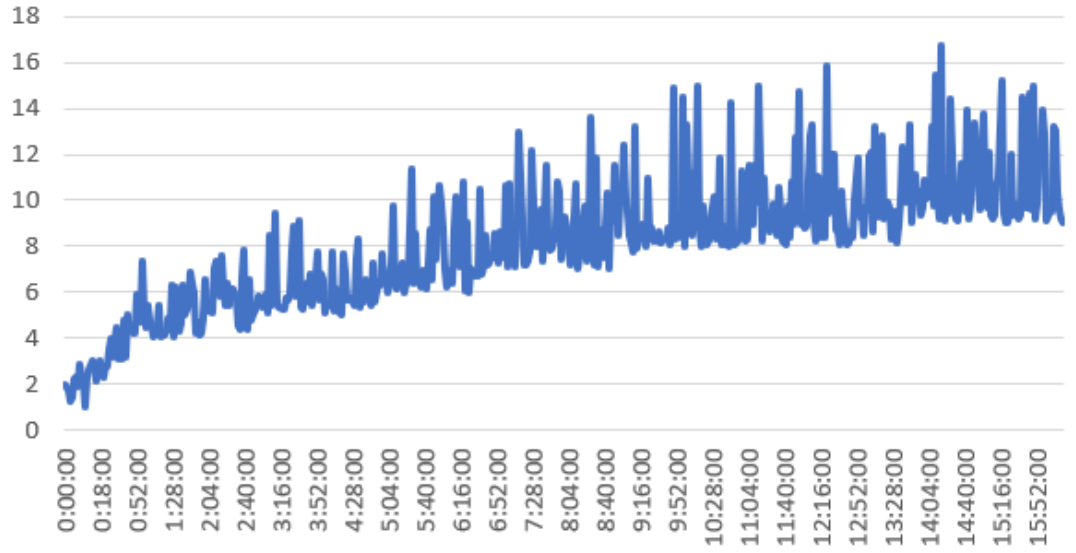

Figure 7: Time of receiving positive reward in dependence of generation

5. Discusiion

For this algorithm, improvements in the selection method of individuals from the population for gene inheritance are possible, as individuals with poorly selected initial parameters in some RL

algorithms quickly lose the competition and the ability to produce offspring (Figure 8). Also, this problem could probably be solved by more individuals in the population, but it will take more time and hardware.

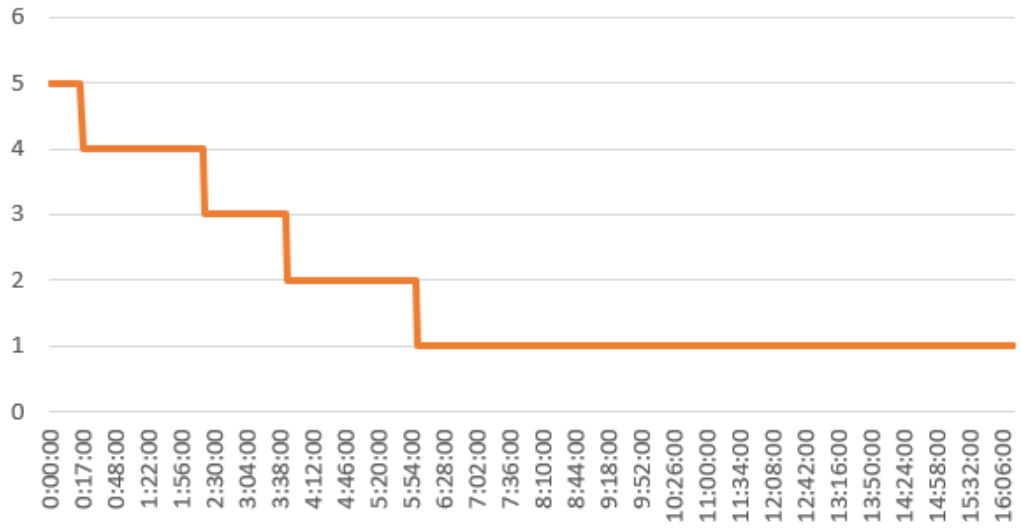

Figure 8: Changing the number of RL algorithms during the experiment

Another possible improvement in the algorithm may be an improvement in the mutation method. It would be a significant improvement in the final characteristics if some genes were given the possibility of mutation.

6. Conclusions

The article considers machine learning algorithms with complement and metaheuristic algorithms. The result of the study was the ability to combine different DRL algorithms with a genetic algorithm and automatically select the best DRL models to solve the solution.

During the experiment, a population of 30 CartPole agents was analyzed in a virtual gym environment. The result of the experiment was the selection of one DRL algorithm from the sample with some differences in the hyperparameters of the model.

7. References

[1] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare and Joelle Pineau, “An Introduction to Deep Reinforcement Learning”, Foundations and Trends in Machine Learning: Vol. 11, No. 3-4. 2018.

[2] Pascal Klink, Hany Abdulsamad, Boris Belousov, and Jan Peters. Self-paced contextual reinforcement learning. In CoRL, 2019.

[3] Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In AAAI, 2020.

[4] Bellman, R. 1957b. “Dynamic Programming”, 2017.

[5] Barto, A. G., R. S. Sutton, and C. W. Anderson. “Neuronlike adaptive elements that can solve difficult learning control problems”. IEEE transactions on systems, man, and cybernetics. (5): 834–846. 1983.

[6] Sutton, R. S. and A. G. Barto. Reinforcement Learning: An Introduction (2nd Edition, in progress). MIT Press. 2017.

[7] OpenAI documentation page. https://spinningup.openai.com/

[8] Joshua Achiam. Spinning Up Documentation. Release. 2020
[9] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. 2016
[10] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra. Continuous control with deep reinforcement learning. 2015.
[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. 2018.
[12] Bonabeau E, Dorigo M, Theraulaz G Swarm intelligence: from natural to artificial systems. Oxford University Press, Inc. (1999)
[13] Sourabh Katoch, Sumit Singh Chauhan & Vijay Kumar. A review on genetic algorithm: past, present, and future. 2020
[14] Goldberg D (1989) Genetic algorithm in search. Optimization and Machine Learning, Addison - Wesley, Reading, MA 1989
[15] Genetic Algorithm Implementation in Python. https://towardsdatascience.com/genetic-algorithm-implementation-in-python-5ab67bb124a6
[16] Metaheuristics. Kenneth Sörensen University of Antwerp, Belgium Fred Glover University of Colorado and OptTek Systems, Inc., USA
[17] Held, D., Geng, X., Florensa, C., and Abbeel, P. (2017). Automatic goal generation for reinforcement learning agents. arXiv preprint arXiv:1705.06366.
[18] Kryvenchuk Y., Shvorob I., and ect. Research by statistical methods of models of the function of transformation of optical circuits of the means of measuring the temperature based on the effect of Raman. CEUR Workshop Proceedings Vol. 2654. 2020.
[19] Xie, Q., Chen, Y., Wang, Z., and Yang, Z. Learning ZeroSum Simultaneous-Move Markov Games Using Function Approximation and Correlated Equilibrium. arXiv preprint arXiv:2002.07066, 2020.
[20] Yuan, J. and Lamperski, A. Online convex optimization for cumulative constraints. In Advances in Neural Information Processing Systems, pp. 6137–6146, 2018.
[21] Miryoosefi, S., Brantley, K., Daume III, H., Dudık, M., and Schapire, R. Reinforcement learning with convex constraints. arXiv preprint arXiv:1906.09323, 2019.
[22] Liu, Q., Yu, T., Bai, Y., and Jin, C. A sharp analysis of model-based reinforcement learning with self-play. arXiv preprint arXiv:2010.01604, 2020.
[23] Brantley, K., Dudik, M., Lykouris, T., Miryoosefi, S., Simchowitz, M., Slivkins, A., and Sun, W. Constrained episodic reinforcement learning in concave-convex and knapsack settings. arXiv preprint arXiv:2006.05051, 2020.
[24] Chen, X., Hu, J., Li, L., and Wang, L. Efficient reinforcement learning in factored mdps with application to constrained rl. arXiv preprint arXiv:2008.13319, 2020.
[25] Shevcov, A. H., and O. V. Il'yina. "Nejropsyxolohichnyj pidxid u korekciyi rozvytku ditej z psyxofizychnymy porushennyamy." Naukovyj chasopys Nacional"noho pedahohichnoho universytetu imeni MP Drahomanova. Seriya 5: 347-360.
[26] François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. (2018). An introduction to deep reinforcement learning. arXiv preprint arXiv:1811.12560.
[27] Okwu, Modestus O., and Lagouge K. Tartibu. "Genetic Algorithm." Metaheuristic Optimization: Nature-Inspired Algorithms Swarm and Computational Intelligence, Theory and Applications. Springer, Cham, 2021. 125-132.