ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

Tournament Selection Improves Cartesian Genetic Programming for Atari Games

Tim Cofala, Lars Elend, and Oliver Kramer

Computational Intelligence Group
Department of Computing Science
University of Oldenburg
26122 Oldenburg, Germany

Abstract. The objective of this paper is to extend Cartesian Genetic Programming (CGP) for the evolution of Atari game agents in the Arcade Learning Environment. Based upon preliminary work on the use of CGP playing Atari games, we propose extensions like the repeated evaluation of elite solutions. Furthermore, we improve the CGP optimization process by increasing the diversity in the population with tournament selection. Experimental studies on four exemplary Atari games show that the modifications decrease premature stagnation during the evolutionary optimization process and result in more robust agent strategies.

1 Introduction

Learning behavioral strategies in complex environments is an important challenge in various domains including game playing, robot control and even autonomous driving. The Arcade Learning Environment (ALE) featuring Atari games is an interesting testbed for reinforcement learning in the domain of game AIs. Recently, Wilson *et al.* [10] successfully applied Cartesian Genetic Programming (CGP) to the evolution of Atari game programs achieving state-of-the-art performance for some games. CGP uses genetic operations to create and modify a population of computer programs in search of an optimal solution. In comparison to deep learning approaches [9, 8], whose game strategies are implicitly encoded in weights, CGP has the advantage of producing human-readable program code. However, a fast and robust evolutionary optimization process is difficult to achieve. In this paper, we build upon Wilson's work [10] and extend CGP by, (1) introducing a reevaluation strategy for the evolution of more robust agents without increasing the total number of evaluations and (2) enhancing the diversity in the population with a tournament selection variant.

This paper is structured as follows. Section 2 gives a short introduction to ALE and Atari games. Related work on CGP and Atari games is introduced in Section 3. The extensions of CGP and the tournament selection variant are introduced in Section 4 and experimentally analyzed in Section 5. Conclusions are drawn in Section 6.

2 Arcade Learning Environment

ALE is a free open source software framework introduced by Bellemare *et al.* [1]. It has been designed for the development and evaluation of general, domain

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

independent agents. ALE provides an interface to more than 50 Atari games based on Atari 2600, which is a video game console developed in 1977. Each game can be comprised as a reinforcement learning problem, provided by the framework through a game-handling layer. For every time step, typically each frame, agents receive a 2D array consisting of 160x210 pixel inputs. Agents can respond with one of 18 possible, discrete actions. A reward is provided, depending on the difference in the game score between subsequent frames. ALE also handles game management procedures like the termination and reset of games and saving and restoring of game states.

3 Cartesian Genetic Programming

CGP was introduced by Miller in 1999 [7] as an extension of genetic programming (GP) [5]. Similar to GP it aims at evolving computer programs to solve problems in various domains. CGP solutions are represented by directed acyclic graphs. A CGP graph consists of several nodes, which can be divided into three categories depending on their task in the graph: input nodes, function nodes, and output nodes. Input nodes represent constants or variables as program inputs. They will be passed into the direction of the output nodes in a feed-forward manner. Output nodes represent the program output, e.g., actions in case of reinforcement learning tasks. The function nodes between the input and output nodes allow modification and shaping of the flow of information. In many representations, function nodes are organized in columns and rows and can be explicitly indexed by Cartesian coordinates. Each function node chooses its respective function from a global, domain specific set of primitive functions. The definition of the function set is a key aspect for the design of a successful CGP approach and highly problem dependent. The number of inputs of function nodes depends on the arity of the employed function set.

Harding *et al.* [3] introduced mixed type Cartesian genetic programming (MT-CGP) to enable CGP to work on variable data types. In MT-CGP the functions, used by the computational nodes, are overloaded, so they can work on multiple types of data like real numbers and vectors.

Wilson *et al.* [10] were the first introducing CGP to Atari games. They build upon previous work applying CGP to image processing and filtering tasks. They use pixel matrices as inputs for the CGP agents and a function set featuring mathematical, statistical, and matrix functions. CGP for reinforcement learning tasks has rarely been studied, in contrast to deep learning and neuro-evolution approaches, e.g. [9, 8].

4 Extending CGP for Atari Games

Our experiments are based on a reimplementation of the work by Wilson *et al.* [10]. Figure 1 outlines the CGP ALE game approach. CGP creates a program, which has access to the ALE images as input and to the games' actions as output. The achieved scores are basis of the program's fitness.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

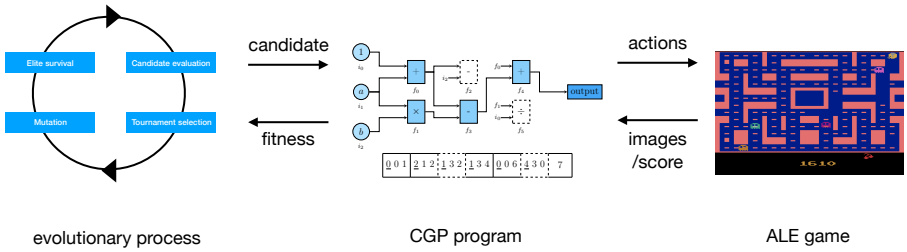

Fig. 1: Overview of CGP playing ALE game framework.

4.1 Reevaluation

Wilson *et al.* used a the common $(1+\lambda)$ evolutionary strategy for their approach, i.e, in each generation λ child individuals are created by mutating the one parent individual. The best individual among parent and children is selected as parent for the next genreation. However, with the stochasticity of the ALE an individual can have varying fitness values when evaluated multiple times. Thus, evaluating an agent only once could lead to an overestimation of its performance. Therefore, we introduce a revaluation strategy for CGP. When evaluation takes place every individual in the population is evaluated regardless of whether it has been evaluated before. In case of a $(1 + \lambda)$ this only applies to the parent individual. Its fitness is made up by the average of the fitness values of all its evaluations. Reevaluation should therefore lead to a better estimation of the parent's real performance, while keeping the overhead of additional evaluations minimal. In general, this strategy could increase the robustness of evolved agents, i.e., develop a game AI that performs similarly in repeated games.

4.2 Tournament Selection

As next step, we aim at increasing diversity in the population and preserve individuals that have performed excellently multiple times. To achieve this, the classic $(1 + \lambda)$-CGP strategy is abandoned in favor of the following strategy, which is more in oriented to the fashion of classic genetic algorithms [6] and has successfully been applied to CGP variants using crossover [2]:

1. We introduce tournament selection allowing every individual to be selected as a parent which could lead to an increase of population diversity.
2. We increase the population size λ of child individuals produced in each generation and the number κ of surviving elites.

Algorithm 1 shows the pseudocode of the proposed selection strategy. Tournament selection with tournament size ξ selects one parent, which is mutated afterwards. This process is repeated until λ children are created. Additionally, κ elites are added that survive from the previous generation. Last, the new population is evaluated including a reevaluation of the surviving elites.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

Algorithm 1 Population generation with tournament selection

1: **repeat**
2: parent $\mathbf{p} \leftarrow$ tournament selection (size ξ, population $\mathcal{P}$)
3: child $\mathbf{c} \leftarrow$ mutation of $\mathbf{p}$
4: **until** λ children created
5: new population $\mathcal{P}' \leftarrow$ children $\mathbf{c} \cup \kappa$ elites
6: evaluate population $\mathcal{P}'$

5 Experiments

In the following, we experimentally analyze Wilson's CGP strategy (*reimpl.*) equipped with (1) repeated evaluations of a candidate solutions (*reeval.*), (2) increased population sizes, and (3) tournament selection (*tourn.*) on the games Asteroids, Boxing, Ms. Pacman, and Space Invaders.

5.1 Settings

In this paragraph we describe the experimental setup. CGP is parameterized oriented to the work of Wilson *et al.*, in particular using the same function set. Individuals consist of 40 function nodes aligned in one row. Point mutation is used for the creation of new individuals with a mutation rate 0.1. Likewise, a $(1+9)$ strategy is used in the *reeval.* condition. For the *tourn.* runs, the number of elite individuals is set to $\kappa = 3$, while the number of children is increased to $\lambda = 12$, resulting in a population size of 15 individuals. Tournaments are held between $\kappa = 4$ randomly selected individuals. For comparability, each run is limited to 10,000 evaluations taking the reevaluations into consideration. The ALE is configured to include stochasticity by setting the repeat action probability to 0.25.

After termination of CGP the last generation's best performing agent is evaluated in 20 separate trials of the respective game. This *test phase* provides a better understanding of the agent's real performance. In contrast, the score of the *evolution phase* is based on the best individual's fitness of the last generation for every run.

5.2 Results

Per experiment and game 10 runs are executed and the results are presented with mean and standard deviation in Table 1 and Figure 2. In comparison to the original *reimpl.* strategy, *reeval.* and *tourn.* runs terminate with a lower score in all tested games (*evolution phase*). However, the results of the *test phase* indicate a drastic difference to the results of the *evolution phase*. Although *reeval.* and *tourn.* terminated with lower fitness values, these scores are a better estimation of the agents' real performance, i.e. the results in the *test phase*. Furthermore, comparing the *test phase* results, *reeval.* evolved better performing agents in the games Asteroids, Ms. Pacman and Space Invaders. Agents created with

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

the *tourn.* strategy reached superior results in the games Asteroids (M = 3312) and Ms. Pacman (M = 1156.5) compared to the other configurations. Merely for the game Boxing, *reimpl.* evolved the best performing agents.

Table 1: Experimental comparison of the three CGP approaches.

approach	game	evolution phase	test phase	diff
reimpl.	Asteroids	6555.0 ±1120.9	2172.3±520.2	4382.7
	Boxing	49.0 ±24.1	11.1 ±17.3	37.9
	Ms. Pacman	2469.0 ±676.9	611.0±475.1	1858.0
	Space Invaders	987.0 ±169.6	595.1±291.6	391.9
reeval.	Asteroids	2799.1 ±1120.6	2560.3±956.4	238.9
	Boxing	14.0 ±17.6	7.8 ±17.1	6.2
	Ms. Pacman	946.5 ±539.5	855.7±572.6	90.8
	Space Invaders	644.7 ±143.9	642.3±158.9	2.4
tourn.	Asteroids	4455.9 ±909.7	3312.0±651.8	1143.9
	Boxing	13.2 ±12.5	8.4 ±13.8	4.9
	Ms. Pacman	1254.7 ±490.1	1156.5±556.8	98.2
	Space Invaders	649.9 ±53.2	650.0 ±56.3	−0.7

The significance of the experimental results is tested with the Mann-Whitney-Wilcoxon test. The analysis reveal that the *tourn.* strategy produces significantly better solutions for the game Asteroids with $W = 7$, $p < .001$ and for Ms. Pacman with $W = 20$, $p = .023$, but not for Space Invaders with $W = 46$, $p = .78$ and Boxing with $W = 48$, $p = .91$. For the latter, the average performance in the test runs was slightly lower for both strategies featuring *reeval.* albeit this difference is not significant.

Figure 2 shows that even if the CGP programs are created with the same procedure and configuration, their performance in different games can vary considerably. Some games show outliers like the high performing *reimpl.* strategy agent for Boxing. To compensate such outliers, which may disturb the evolutionary process, a further increase of repetitions at a high cost may be advantageous.

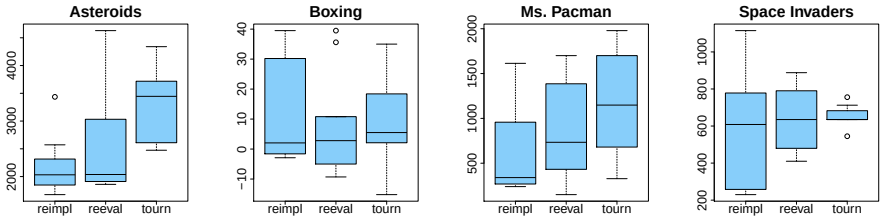

Fig. 2: Results of the agents in the test phase for the four games: Asteroids, Boxing, Ms. Pacman, and Space Invaders

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

6 Conclusions

The significance of experiments at Atari games with only one repetition can be limited due to outliers. Reevaluating individuals during the course of evolution leads to a better estimation of their capabilities and in general to more robust agents. Tournament selection offers increased population diversity due to its adjustable selection pressure. Our combination of the reevaluation strategy and tournament selection is a significant step towards the prevention of premature stagnation. Allowing an increased number of elite individuals can protect well evaluated ones from being replaced by outliers. All in all, our results indicate that for complex problem domains like Atari games, CGP requires more complex evolutionary operators and algorithmic modifications than the simple $(1 + \lambda)$-CGP offers. Future work may concentrate on the analysis of advanced genetic operators like subgraph crossover [4]. Further, we plan to employ niching strategies to increase diversity in the population.

Acknowledgements

We thank the German Research Foundation (DFG) for supporting our work within the Research Training Group SCARE (GRK 1765/2).

References

[1] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[2] J. Clegg, J. A. Walker, and J. F. Miller. A new crossover technique for Cartesian genetic programming. *Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07*, page 1580, 2007.

[3] S. Harding, V. Graziano, J. Leitner, and J. Schmidhuber. Mt-cgp: Mixed type cartesian genetic programming. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 751–758. ACM, 2012.

[4] R. Kalkreuth, G. Rudolph, and A. Droschinsky. A new subgraph crossover for cartesian genetic programming. In *European Conference on Genetic Programming*, pages 294–310. Springer, 2017.

[5] J. R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, jun 1994.

[6] B. L. Miller, D. E. Goldberg, et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.

[7] J. F. Miller. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2*, pages 1135–1142. Morgan Kaufmann Publishers Inc., 1999.

[8] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[9] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[10] D. G. Wilson, S. Cussat-Blanc, H. Luga, and J. F. Miller. Evolving simple programs for playing Atari games.