ALF – A Fitness-Based Artificial Life Form for Evolving Large-Scale Neural Networks

Rune Krauss Marcel Merten Mirco Bockholt Rolf Drechsler
Group of Computer Architecture, University of Bremen, Cyber-Physical Systems, DFKI GmbH, Germany
{krauss, mar_mer, bockholt, drechsler}@uni-bremen.de

ABSTRACT

Topology and Weight Evolving Artificial Neural Network (TWEANN) is a concept to find the topology and weights of *Artificial Neural Networks* (ANNs) using genetic algorithms. However, a well-known downside is that TWEANN algorithms often evolve inefficient large ANNs for large-scale problems and require long runtimes.

To address this issue, we propose a new TWEANN algorithm called *Artificial Life Form* (ALF) with the following technical advancements: (1) speciation via structural and semantic similarity to form better candidate solutions, (2) dynamic adaptation of the observed candidate solutions for better convergence properties, and (3) integration of solution quality into genetic reproduction to increase the probability of optimization success. Experiments on large-scale problems confirm that these approaches allow effective solving of these problems and lead to efficient evolved ANNs.

CCS CONCEPTS

• **Computing methodologies → Genetic algorithms**; **Neural networks**;

KEYWORDS

Neuroevolution, Genetic Algorithms, Neural Networks

1 INTRODUCTION

The concept *Weight Evolving Artificial Neural Network* (TWEANN) allows the evolution of the topology and weights of *Artificial Neural Networks* (ANNs) using genetic algorithms. Several studies have shown that TWEANN algorithms are capable to solve complex control tasks [6]. However, for large-scale problems the ANN size can grow rapidly and lead to inefficient large ANNs [5]. Although efforts have been made to improve the capability of TWEANN algorithms to solve problems with large state spaces, there is still room for improvement for existing technologies [4].

This paper proposes a new TWEANN algorithm called *Artificial Life Form* (ALF), which has the following main advantages over existing TWEANN algorithms: (1) speciation via structural and semantic similarity to form better candidate solutions, (2) dynamic adaptation of the observed candidate solutions for better convergence properties, and (3) integration of solution quality into genetic reproduction to increase the probability of optimization success. These approaches improve the runtime and memory usage, leading to efficient evolved ANNs. Experimental results on large-scale problems confirm this performance.

GECCO '21 Companion, July 10–14, 2021, Lille, France
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3459545

2 ARTIFICIAL LIFE FORM (ALF)

Our new proposed TWEANN algorithm called ALF integrates three main components described in detail below to address the issue of efficiently evolving ANNs for large-scale problems: (1) *Structural and Semantic Speciation* (SSS), (2) *Dynamic Adaptation of Population* (DAP), and (3) *Fitness-Based Genetic Operators* (FBGO).

To protect individuals' ANNs with new innovations, the concept of species was introduced in [7]. In ALF, species are classified structurally and semantically. Depending on the problem, the overall similarity between two individuals is calculated using the weighted arithmetic mean. The structural comparison determines the similarity via the number of shared connections between the ANNs. The semantic comparison measures the similarity of the ANNs' predictions via the Pearson correlation. While structural species comparison protects topological diversity, semantic comparison protects diversity of behavior patterns. Both factors contribute to increasing genetic diversity. Thus, the objective of SSS is to support diversity of complex structures and behavior patterns to form better individuals so that the search for a solution is efficient.

If the population size cannot change within generations, this may lead to stagnation of exploration in unsuitable search regions that do not advance learning. Therefore, dynamic population is introduced in ALF, which is described as follows. In ALF, fitness sharing is used as in [7]. The fitness proportion of the species s_i is denoted as F_{s_i}. Each species is also assigned an age A_{s_i}, which identifies how many generations this very species has been in the population. To decide whether a species is to be deleted, both factors are considered in combination, which is shown by Equation (1)

$$F_{s_i} < \frac{1}{n} \wedge A_{s_i} > A_o, \tag{1}$$

where n is the number of species and A_o is the age of the oldest species. Both factors penalize older species while protecting younger species. In addition, to avoid unnecessary evaluations, the weakest individuals of each species are also deleted depending on F_{s_i}. The new population size P_{new} depends on the proportion of deleted species S_{del}, which is shown by Equation (2)

$$P_{new} = \begin{cases} min(P_{max}, P_{old} \cdot (1 + S_{del})), & S_{del} > 0 \\ max(P_{old} - P_{init} \cdot ebb, P_{init}), & S_{del} = 0 \end{cases}, \tag{2}$$

where ebb is an ebb factor, P_{max} is the maximum population size, and P_{init} is the minimum population size. The number of reproduced offspring O_{s_i} of a respective species also depends on its fitness proportion and is determined by Equation (3)

$$O_{s_i} = F_{s_i} \cdot (1 - o_{init}) \cdot (P_{new} - (P_{max} - P_{vacant})), \tag{3}$$

where P_{vacant} is the available room that can be used for the reproduction and o_{init} is the proportion of newly initialized ANNs, so that possible dead ends can be overcome with a higher probability.

Hence, the objective of DAP is to improve convergence properties by overcoming insufficient search regions through increased simultaneous sampling.

Based on the selected parents, variation operators are applied to reproduce offspring and explore the search space. In a standard mutation, ANNs are changed randomly. While small changes can increase the time required to explore the search space, large changes are not suitable for local optimization. Furthermore, crossover of ANNs of unequally fit individuals may result in unsuitable ANNs, leading to slow convergence. To counteract these problems, ALF introduces fitness-based mutation and crossover. Mutations in ALF can change connection weights and ANN structures. For all mutations, a mutation rate m_r is calculated, which is defined in Equation (4).

$$m_r = min\left(max\left(1, \left(\left((1+m_o) - \frac{F}{f_t}\right) \cdot m_a\right)\right), m_a\right) \tag{4}$$

The offset m_o determines up to which proportion of the current fitness F of an individual in relation to the global fitness threshold f_t the maximum number of mutation attempts m_a is performed. The *weight mutation* changes a connection weight by adding a random number from the normal distribution $\mathcal{N}\left(0, 1.25 - \frac{F}{f_t}\right)$. The fitter an individual is, the smaller changes are made. If F is low compared to f_t, the search process has more influence. If F is high, the optimization process is strengthened. In order to change connections more frequently than layers or nodes, it is determined for these mutations in addition to m_r whether the respective mutation is performed. The corresponding probability calculation P_{mut} is defined in Equation (5)

$$\mathsf{P}_{mut} = min\left(1 + m_o - \frac{\frac{E}{E_{full}} + \frac{F}{f_t}}{2}, 1\right), \tag{5}$$

where E is the current number of connections and E_{full} is the number of possible connections (assuming the ANN is fully connected). The proportion of connections describes the potential for new connections, i. e., new connection possibilities (nodes or layers) become more likely if the proportion is closer to 1. In this context, *node mutation* randomly adds hidden nodes into layers. The corresponding number of nodes V_{mut} is determined by Equation (6)

$$V_{mut} = \left\lceil V_{max} - \frac{F}{f_t} \cdot V_{max} \right\rceil, \tag{6}$$

where V_{max} is predetermined and denotes the maximum number of nodes to be added. The *layer mutation* inserts a new hidden layer. While mutations are always attempted and gradually increase the ANN size, a predetermined hyper-parameter determines the crossover probability with respect to two selected individuals of a species. Before crossover, the respective fitness proportion of the individuals is interpreted as probability during the ANN comparison. Nodes and connections are inherited with a higher probability from the fitter parent. As a result, the objective of FBGO is to increase the probability of optimization success of individuals.

3 EXPERIMENTAL RESULTS

In order to measure the performance of ALF's components, we chose *NeuroEvolution of Augmenting Topologies* (NEAT) [7] as test algorithm for comparison, since it has already proven to be successful in complex control learning tasks and games compared to other algorithms [2]. We configured *Super Mario Bros.* (SMB) as a benchmark task by setting genetic and ANN hyper-parameters for NEAT and ALF, which we determined through experiments and observations [3]. To allow a representative comparison, the same available parameters, such as population size, were adjusted accordingly. In order to study as many game elements as possible, we considered the SMB game worlds 4-2, 4-4, and 7-2 as large-scale problems. The criterion for success was to complete the respective level. Since this task is PSPACE-complete [1], we considered it as sufficient hard to test ALF's capabilities. Specifically, the average time in minutes (t in min), average number of generations (G), ANN connections (E), and hidden nodes (H) required to solve the problems was compared in 10 runs.

Table 1: Comparison between NEAT and ALF in terms of runtime and ANN sizes to solve different large-scale problems

SMB	Algorithm							
	NEAT				ALF			
	t in min	G	E	H	t in min	G	E	H
World 4-2	1,438	387	3,873	1,159	978	194	2,445	304
World 4-4	1,373	372	3,621	1,106	889	157	2,219	339
World 7-2	1,894	486	4,305	1,438	1,454	273	3,252	483

The results are summarized in Table 1 and clearly confirm that the components proposed satisfy the objectives of this work. ALF outperforms NEAT in all comparisons, solving the problems about 461 minutes faster on average and requires 207 generations less compared to NEAT. In addition, the ANN sizes are reduced by about 41.67 % on average.

4 CONCLUSION

This work focused on one of the main challenges in TWEANN: the efficient solving of large-scale problems. For this purpose, we have developed ALF with different main components – structural and semantic speciation, dynamic adaptation of observed candidate solutions, and integration of solution quality into genetic reproduction. Experimental results confirmed that via these components, effective solving of large-scale problems is allowed, leading to efficient evolved ANNs. Future work will include optimizing the approaches described and extending the comparisons with other test algorithms and benchmark tasks.

REFERENCES

[1] Erik Demaine, Giovanni Viglietta, and Aaron Williams. 2016. Super Mario Bros. Is Harder/Easier than We Thought. (06 2016).

[2] Matthew Hausknecht, Joel Lehman, Risto Miikkulainen, and Peter Stone. 2014. A Neuroevolution Approach to General Atari Game Playing. *Computational Intelligence and AI in Games, IEEE Transactions on* 6 (12 2014), 355–366. https://doi.org/10.1109/TCIAIG.2013.2294713

[3] Rune Krauss, Marcel Merten, Mirco Bockholt, and Rolf Drechsler. 2021. ALF – A Fitness-Based Artificial Life Form for Evolving Large-Scale Neural Networks. (2021). arXiv:2104.08252

[4] Yiming Peng, Gang Chen, Harman Singh, and Mengjie Zhang. 2018. NEAT for large-scale reinforcement learning through evolutionary feature learning and policy gradient search. 490–497. https://doi.org/10.1145/3205455.3205536

[5] Kenneth Stanley, David D'Ambrosio, and Jason Gauci. 2009. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial life* 15 (02 2009), 185–212. https://doi.org/10.1162/artl.2009.15.2.15202

[6] Kenneth Stanley and Risto Miikkulainen. 2003. Competitive Coevolution through Evolutionary Complexification. *Journal of Artificial Intelligence Research* 21 (02 2003). https://doi.org/10.1613/jair.1338

[7] K. O. Stanley and R. Miikkulainen. 2002. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* 10, 2 (2002), 99–127. https://doi.org/10.1162/106365602320169811