On the Use of Cultural Enhancement Strategies to Improve the NEAT Algorithm

Arthur L. A. Paulino, Yuri L. B. Nogueira, João P. P. Gomes, César L. C. Mattos
Department of Computer Science
Universidade Federal do Ceará
Fortaleza, Brazil
arthurleonardo.ap@gmail.com, yuri@dc.ufc.br, jpaulo@dc.ufc.br, cesarlincoln@dc.ufc.br

Leonardo R. Rodrigues
Electronics Division
Institute of Aeronautics and Space
São José dos Campos, Brazil
leonardolrr2@fab.mil.br

***Abstract*—Knowledge transmitted between generations by nongenetic means can be understood as culture. The capacity of individuals from certain species to teach and learn plays a fundamental role in directing the evolutionary process. The Neuroevolution of Augmenting Topologies (NEAT) framework enables evolving neural structures to iteratively solve a given learning problem. However, the NEAT approach does not consider cultural aspects in its formulation. In such a context, the aim of this paper is to propose and evaluate ways of enhancing the NEAT framework with additional learning approaches. The parameters involved in the analysis comprise the Backpropagation and the Extreme Learning Machine (ELM) learning algorithms, the individuals to be taught, the moment when culture manifests in the system, and the nature of the lessons to be learned. Empirical results on sequential learning tasks indicate that cultural enhancements, as well as some of the proposed variations, accelerate the neuroevolution convergence.**

***Index Terms*—NEAT, neuro-evolution, cultural enhancement, learning strategies.**

I. Introduction

Neuroevolution of Augmenting Topologies (NEAT) is a method for evolving the architecture and the weights of neural networks using genetic algorithms [1]. NEAT was designed to avoid the problem of competing conventions, allowing meaningful crossover between individuals with different genetic length. Its training procedure produces networks of increasing complexity starting from simple ones and protects topological innovations that may initially display lower fitness but later develop into powerful solutions [2].

The NEAT algorithm has attracted much attention from researchers due to its simple formulation and remarkable performance in various problems. Several variants have been proposed in recent years such as rtNEAT [3], HyperNEAT [4], cgNEAT [5], odNEAT [6], and CoDeepNEAT [7]. However, none of the previous works evaluate the impact of cultural enhancements in the NEAT algorithm.

In this line of research, the authors in [8] observed that the evolution of neural networks architectures may achieve better learners than better agents. In [9], the authors proposed a strategy in which young individuals learn from the older ones, resulting in an improved final model. It is argued in [10] that during the neuroevolution the ability to learn may be more important than the ability to execute. In [11], the author showed the benefits of having the fittest individual as a tutor of the youngest individuals. The definition of a syllabus based on parents experience was also proposed in [11], where the offspring receives the same sensory input received by its parents.

According to [11], culture is a transmission of traits from one generation to the next via nongenetic means. In the context of neuroevolutionary computational methods, culture can be seen as any strategy that may enhance the performance of neural models but is not directly related to the learning algorithm itself.

In this paper, we aim to propose and evaluate various cultural enhancements strategies for the NEAT framework. The cultural enhancements comprise different ways to use learning algorithms to optimize the weights of the neural models. We propose to evaluate the impact of the cultural enhancement strategies in two well known sequential decision benchmark tasks, namely the Cart-Pole and the Mountain-Car.

The remainder of the paper is as follows. Section II presents the basic concepts of the NEAT algorithm and the two learning methods considered in this paper. Section III presents the cultural enhancement strategies that were tested. Section IV describes the experiments carried out to evaluate the proposed strategies and the obtained results. Finally, conclusions and future research directions are discussed in section V.

II. Background

In this section, we briefly summarize the methods that comprise the basis of this paper: the NEAT framework, the stochastic optimization approach via genetic algorithms and two neural networks training strategies: the backpropagation algorithm and the extreme learning machine (ELM).

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

A. NeuroEvolution of Augmenting Topologies (NEAT)

NEAT is a technique for evolving Artificial Neural Networks (ANNs) using an evolutionary algorithm. It was originally developed to evolve ANNs to solve control and sequential decision tasks [4].

According to [1], the remarkable performance of the original NEAT can be explained by three main particular aspects that differ from other neuroevolutionay methods. These aspects are listed below, and more detailed information on the NEAT framework can be found in [1].

- NEAT starts with a population of chromosomes of minimal complexity and, over generations, the structure of such chromosomes becomes more complex, leading to increasingly sophisticated behavior.
- NEAT keeps track of the genes by assigning a unique historical marking to every new piece of network structure that appears. The marking is a number assigned to each gene corresponding to its order of appearance over the course of evolution. The numbers are inherited during crossover unchanged, which allows NEAT to perform crossover among diverse topologies without the need for expensive topological analysis [4]. This procedure allows ANN structures to increase in complexity over generations.
- NEAT speciates the population so that individuals compete primarily within their own niches instead of with the whole population. In that way, topological innovations are protected and have time to optimize their structure before they have to compete with other niches in the population [3].

B. Genetic Algorithms

Genetic algorithms (GA) are a class of stochastic search-based optimization methods inspired by Charles Darwin's evolution theory [12], [13]. Although there is not a unique formulation, GA implementations present the following basic concepts: a population of individuals, a selection strategy and one or more *crossover* and *mutation* operators [14].

In summary, each individual is represented by a single chromosome, a vector that encodes a solution candidate for the optimization task. First, a random population of individuals is generated. Then, at each generation (algorithm iteration), a subset of individuals is chosen to proceed to the mating step. Such selection is random, but must consider the fitness of each individual, which is proportional to its correspondent value in the optimization objective function. Thus, the fittest candidate solutions have a higher probability of being chosen.

The selected individuals (parents), generate an offspring of new candidate solutions, following the application of the crossover and mutation operators. The former randomly combines the components of the mating chromosomes, while the latter randomly changes the values of a few components of the resulting candidate solution. Finally, an additional step must choose which individuals among the original population and the new offspring should proceed to the next generation. We refer the reader to the book [15] for details on several GA variants.

C. Artificial Neural Networks Training Algorithms

The Multilayer Perceptron (MLP) is one of the most commonly used ANN architectures. In its simplest shallow form, it is comprised of a hidden layer of weights and an output layer, where the input of each layer usually includes an independent bias term. Such configuration provides a powerful parametric structure that acts as adaptive basis functions. It is well known that the MLP network with nonlinear hidden activation functions is a universal approximator [16].

In a regression context, the main goal is to learn a nonlinear mapping between a given set of N inputs $\boldsymbol{x}_i|_{i=1}^{N}$ and a corresponding set of outputs $y_i|_{i=1}^{N}$. Here, a single dimension output is considered for the sake of simplicity. The values of the network parameters define how the network transformation process behaves. Thus, the network output is parameterized by the hidden biases $\boldsymbol{b}$ and weights $\boldsymbol{w}_j|_{j=1}^{N_H}$ associated with the N_H hidden units, and the output bias d and weights $\boldsymbol{m}$, and is given by:

$$\hat{y}_i = \boldsymbol{m}^\top \boldsymbol{h}_i + d,$$
$$\boldsymbol{h}_i = \phi_1(\boldsymbol{W}\boldsymbol{x}_i) = [\phi_1(\boldsymbol{w}_1^\top \boldsymbol{x}_i + b_1), \ldots, \phi_1(\boldsymbol{w}_{N_H}^\top \boldsymbol{x}_i + b_{N_H})]^\top,$$

where $\boldsymbol{W}$ is a matrix whose the j-th row is given by the weight vector $\boldsymbol{w}_j$ and $\phi(\cdot)$ is the activation function of the hidden layer.

D. Learning Methods

As follows, we briefly describe two distinct approaches to tackle the parameter learning task in MLP networks: the backpropagation and the extreme learning machine algorithms.

*1) **The Backpropagation Algorithm (BPG)**:* The main idea behind the well known BPG algorithm [17] consists in providing a way to learn the network parameters following a gradient-based approach. First, a loss function $E\left(\boldsymbol{w}_j|_{j=1}^{N_H}, \boldsymbol{b}, \boldsymbol{m}, d\right)$ is defined in terms of the hidden biases $\boldsymbol{b}$ and weights $\boldsymbol{w}_j|_{j=1}^{N_H}$, and the output bias d and weights $\boldsymbol{m}$.

Then, the BPG proceeds by computing the gradients $\partial E/\partial \boldsymbol{\theta}$, where $\boldsymbol{\theta} = \left\{\boldsymbol{w}_j|_{j=1}^{N_H}, \boldsymbol{b}, \boldsymbol{m}, d\right\}$ contains all the network parameters. Such gradients are computed by following the standard chain rule of calculus. Finally, the parameters are updated in a gradient descent fashion:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_{\text{BPG}} \frac{\partial E}{\partial \boldsymbol{\theta}},$$

where α_{BPG} is a learning step towards the opposite direction of the loss gradient. The loss and the gradients are often computed only on a subset of training samples at each iteration, which results in a stochastic gradient descent procedure. Since the loss is a nonlinear function of the network parameters,

the optimization problem is non-convex and prone to local minima.

*2) **The Extreme Learning Machine (ELM)**:* The ELM network [18] is an alternative approach to training feedforward neural networks. In the ELM framework, the hidden weights and biases are randomly chosen and only the output weights, i.e. from the hidden neurons to the output, are analytically determined, usually via the Ordinary Least Squares (OLS) algorithm. The fast learning procedure and the easiness of implementation are the main advantages of the ELM in comparison with over the standard BPG algorithm.

In the ELM configuration, there is usually no output bias and, following the OLS algorithm, the output weights $\boldsymbol{m}$ are computed in batch according to (1).

$$\boldsymbol{m} = (\boldsymbol{H}\boldsymbol{H}^{\top})^{-1}\boldsymbol{H}\boldsymbol{y}, \tag{1}$$

where $\boldsymbol{y}$ contains the N target training outputs, and $\boldsymbol{H}$ is a matrix whose columns are given by the hidden activation vectors $\boldsymbol{h}_i|_{i=1}^{N}$.

Alternatively, the ELM algorithm can follow an iterative approach based on the Least Mean Squares (LMS) algorithm to optimize the output weights [19].

III. Proposed Approach

In the original version of the NEAT framework, both topology and weights are evolved according to an evolutionary algorithm. Indeed, the knowledge encoded in the weights is transferred to the next generation by genetic operators like mutation and crossover.

Although this approach completely agrees with the evolution theory metaphor, some authors believe that adding knowledge by non-genetic means may improve the performance of neuroevolutionary methods, as we discuss in the following lines.

A. The Baldwin Effect and the Lamarckian Inheritance

The culture developed by a population is capable of directing the evolutionary process, even if behaviors acquired after birth are not transmitted to the next generations by genetic means. This is the so-called *Baldwin effect* [20]. Its main justification is that the learning capacity of the individuals is also a genetic feature. Thus, in scenarios where the culture plays an important role for individual success, the genetic patterns of good learners have more probability of being perpetuated. Inspired by those ideas, the authors in [21] and [8] have argued that the Baldwin effect may improve the performance of neuroevolutionary algorithms.

Before Darwin's work [12], Jean-Baptiste Lamarck defended that changes acquired along the lives of animals could be transmitted via mating [22]. Nowadays, such a hypothesis is not accepted as a general rule for the continuity of heritable traits observable in nature. However, it remains as a possible strategy within computational simulations.

In the context of neuroevolutionary approaches which include learning procedures, the so-called *Lamarckian inheritance* may be implemented by updating the network parameters encoded in the chromosomes after the training step, simulating a genetic change triggered by a modification in the phenotype. Such a strategy can be understood as if the experience of other population members could influence the performance of an individual. Some previous works, such as [23], [24] and [25], have already discussed the positive consequences of considering the Baldwin effect and the Lamarckian inheritance in neuroevolutionary algorithms.

Given the above discussion, cultural enhancement strategies for the NEAT framework proposed in this paper are presented in the next section.

B. Cultural Enhancement Strategies

Besides the choice of the learning method, i.e. one of the two methods presented in the previous section (BPG or ELM), it is necessary to define three main points:

1) When the learning process is going to start.
2) Who is going to be trained.
3) Which input-output pairs will be used in the learning procedure.

For the first point (the start of the learning process), we will evaluate the performance of NEAT when the learning process starts either at the beginning of the NEAT execution or a late start, where the original NEAT is used in the first iterations and the learning procedure starts afterward. The intention of the late start variant is to enable a more extensive exploration of the search space in the first NEAT iterations. We emphasize that the learning step aims to improve an individual via the chosen learning strategy, while the NEAT iterations aim to improve the population as a whole via evolution. Both mechanisms should cooperate to achieve better solutions for the task under consideration.

With respect to the chromosomes that will be modified during the learning process, we have decided to evaluate two variants, where the offspring or the parents are modified. While the first option has been used in previous works such as [26], training the parents before the crossover step aims to preserve genetic variability of NEAT since directly training the offspring may lead to very similar parents in the next generation.

Finally, the input-output pairs are generated by providing inputs to *tutor* ANNs. The tutor of each generation is the ANN with the best fitness value among the entire population. In [11], the author recommended that a knowledge base, named *syllabus*, should be formed by 20 to 40 lessons (questions and answers): questions are inputs to the tutor ANN and answers are the correspondent network outputs. Therefore, a syllabus of size 30 is used in this paper. The nature of the inputs may be either random, as suggested in [26], or a specialized one, according to the task under consideration.

In this paper, we present results for experiments involving random and specialized inputs.

We name our proposal, which considers the above strategies in the context of the NEAT framework, as *culturally enhanced* NEAT (ceNEAT). The numerical experiments carried out to evaluate its performance will be presented and discussed in the next section.

IV. Experiments and Results

We conducted tests with two well-known sequential decision benchmark problems to evaluate the performance of all proposed NEAT variants: the Cart-Pole and the Mountain-Car problems. Fig. 1 illustrates the benchmark problems. A brief description of each problem is presented as follows.

A. The Cart-Pole Problem

In this task, the goal is to balance a pole of length 1m and mass 0.1kg vertically, attached to a cart of mass 1.0kg, which can move to the left or right without friction. The pole is connected to the cart by one end and is free to rotate clockwise and counterclockwise according to the gravitational acceleration of 9.8m/s^2. To achieve the goal, the intelligent agent can interfere in the system by applying a horizontal force on the cart of 10.0N or -10.0N. The ANN has access to the readings of four sensors: position and speed of the cart, as well as the angle and the angular velocity of the pole. The evaluation environment used is an adaptation of the experiment defined in [27]. The simulation ends when at least one of the following conditions is verified:

- The cart distances more than 2.4m from the center (failure);
- The angle between the pole and the vertical axis becomes greater than 12^o (failure);
- The evaluation lasts 500 iterations without violating the previous conditions (success).

For each iteration, the agent's fitness is incremented by $1/(1+d)$, where d is the distance in meters to the center of the environment.

B. The Mountain-Car Problem

In this task, the car needs to climb the mountain but does not have sufficient acceleration to climb it in a single step. Therefore, the solution is to behave like a pendulum until it achieves enough speed to reach the flag at the top of the mountain. The minimum horizontal position is -1.2m, the maximum is 0.6m and the target flag is at 0.5m. The maximum modulus for the horizontal velocity is 0.07m/s and the gravitational acceleration is simulated by a simple harmonic motion governed by a cosine function. The agent can interfere in the system by accelerating to the left, to the right or not accelerating at all. The ANN has access to the readings of two sensors: the horizontal position and the speed of the car. The environment used is an adaptation of the problem proposed in [28]. The simulation ends when at least one of the following conditions is verified:

- The evaluation lasts 500 iterations (failure);
- The car climbs the mountain and reaches the flag (success).

At the end of the evaluation, the fitness of the agent is equal to $(x_{\text{max}} + 0.5)\times 10^4/t$, where x_{max} is the maximum position reached by the car and t is the duration of the simulation (in number of iterations).

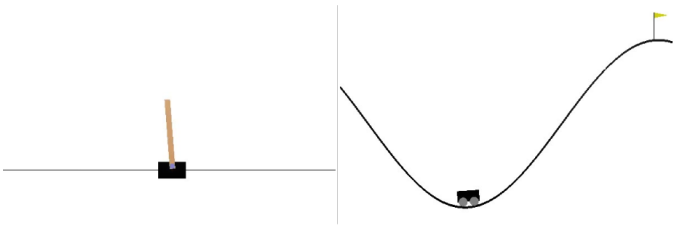

Fig. 1. Illustration of the Cart-Pole (left) and the Mountain-Car (right) problems

C. Results and Discussion

The goal of the experiments is not to find the best set of parameters for each problem, but to validate the influence of the variants on the quality of the solutions. The population was composed of 100 individuals and, in each generation, each of the individuals was evaluated three times. Their fitness were defined as the average of the fitness attained in each evaluation. The graphs were generated by averaging the fitness of the best agent of each generation since the experiments were repeated 500 times for each setup.

The experiments were implemented in Python 2.7.14, based on the open libraries NEAT-Python 0.92 and Gym 0.9.4. The experiments were performed on the Arch Linux x86_64 operating system, Kernel 4.15.14-1-ARCH, which had an Intel i5-7200U (4) processor at 3.1 GHz and sufficient RAM (7.6 GiB).

As an initial test, we designed the culturally enhanced NEAT (ceNEAT) with variants that were previously used in other neuroevolutionay approaches. Here, we use the BPG algorithm along with a random syllabus and the training offspring strategy. All cultural modifications started at the beginning of the execution. This configuration will be used in the subsequent tests as a baseline for further comparisons. The results obtained by NEAT and ceNEAT are presented in Fig. 2.

As can be noticed, in both benchmark problems the baseline ceNEAT was able to find better individuals (higher fitness) after a few iterations of the method. This result corroborates with our hypothesis that cultural enhancement strategies may improve the performance of NEAT. In the next sections, we evaluate the performance impact of all other cultural enhancements strategies when the training procedure is either the BPG or the ELM algorithms.

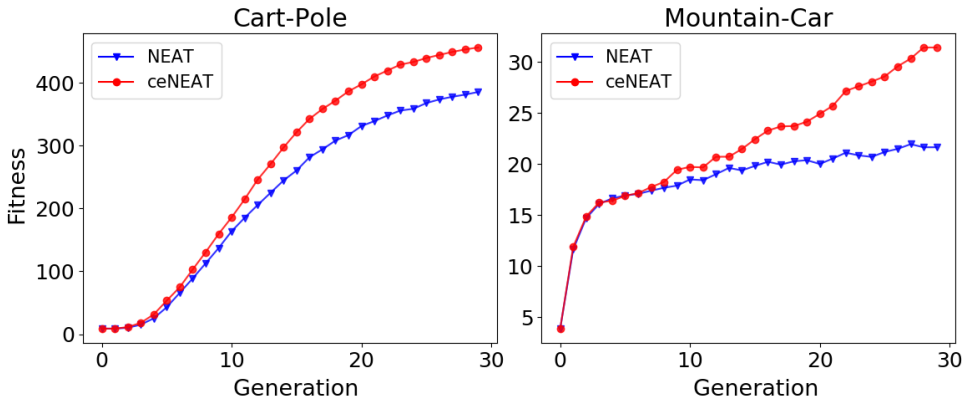

Fig. 2. Baseline results for NEAT and ceNEAT

*1) **Results Using the BPG Algorithm**:* In the first of three tested scenarios, we evaluated the impact of training parents or offspring along with the iterations. The results are presented in Fig. 3.

The obtained results indicate that training parents instead of offspring could not lead to any improvement of the best individual found at each iteration. Even though the parents training strategy did not achieve good results, it is important to highlight that in such variant the number of trained individuals is reduced, since the number of parents is lower than the number of individuals in the offspring. Thus, this variant could provide, at least, the benefit of a reduced computational cost.

In the second experiment, we investigated the impact of performing the training procedure just after a few iterations of ceNEAT. Fig. 4 shows the result obtained for both benchmark problems.

Based on the results, we could not verify the benefit of postponing the beginning of the training procedure. Although we expected a more extensive exploration of the search space in the initial iterations, we could not find any evidence that this happened or impacted the final result.

The last variant consisted of designing a syllabus based on the experience achieved in the tackled problems. This approach is an alternative to the standard random generation of the input patterns. Fig. 5 shows the results for this scenario.

Although both the baseline and the modified ceNEAT achieved similar results, it is possible to verify an improvement in both benchmark tasks when using the syllabus that considers specialized inputs based on previous experiences.

*2) **Results Using the ELM Algorithm**:* In the second set of experiments, all three modifications were evaluated once again but using the ELM training algorithm. We also started our analysis by comparing NEAT, the baseline ceNEAT with BPG and the baseline ceNEAT with ELM. The results are shown in Fig. 6.

Once again we observed that both ceNEAT versions were able to outperform the standard NEAT. We also could verify that the ELM variant was able to achieve better results in the Mountain-Car problem.

The results for all other variants (training parents, late start and experienced syllabus) are shown in Figs. 7, 8 and 9 respectively.

In general, for the Cart-Pole problem, we can observe that no ELM variant could outperform the BPG-based learning approach. Concerning the comparison among variants of the ELM-based ceNEAT, the results were very similar to the ones found for the BPG, only experienced syllabus obtained better results.

An interesting result can be noticed for the Mountain-Car problem: in all tested scenarios, ELM variants reached significantly better results than BPG variants. Again, the only variant that presented significant improvements was the experienced syllabus.

V. Conclusions and Further Work

The results observed in the experiments indicate that our approach, ceNEAT, which includes cultural enhancements, was able to improve the quality of the solutions found by the standard NEAT on both benchmark problems.

With respect to the choice of the learning algorithm, both the BPG and the ELM achieved better results than those achieved with standalone NEAT. However, BPG was more appropriate for the Cart-Pole problem, while ELM attained better results on the Mountain-Car problem.

We have also observed that teaching only the parents and triggering the cultural process belatedly may simplify the teaching process, but such variations may also diminish the benefits of cultural enhancement.

The strategy of extracting the lessons from tutor experiences also achieved good results on both problems, regardless the learning algorithm. Thus, such a strategy can be seen as a generally good practice to implement cultural enhancements on the NEAT algorithm.

In this paper, culture was treated as knowledge transmitted from one generation to the next by non-genetic means. However, this process was done in a rigid way, that is, there was a phase in each generation in which some individuals were

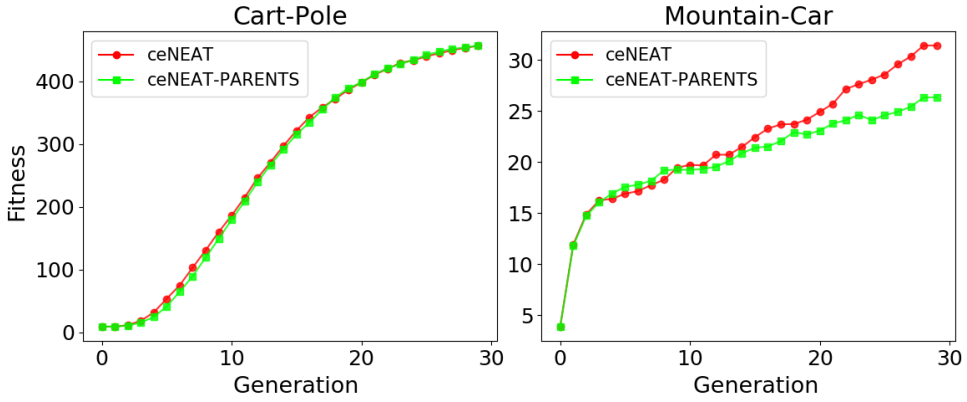

Fig. 3. Results for ceNEAT (offspring training) and ceNEAT-PARENTS (parents training)

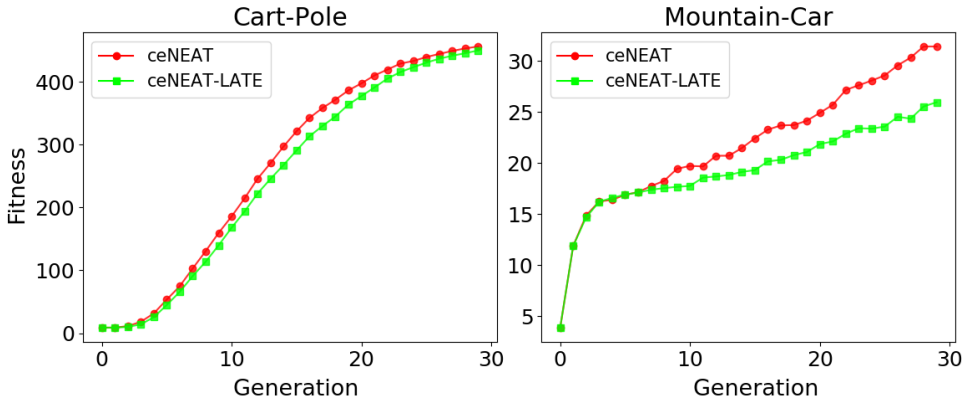

Fig. 4. Results for ceNEAT (training since the beginning) and ceNEAT-LATE (training belatedly)

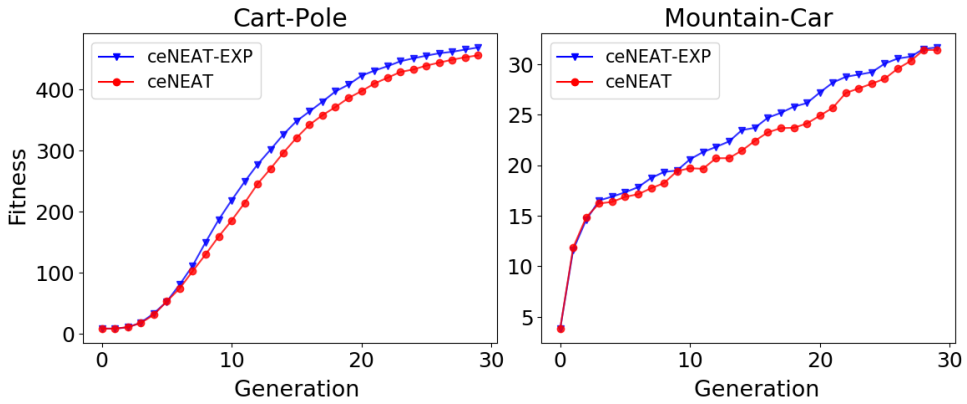

Fig. 5. Results for ceNEAT (random inputs) and ceNEAT-EXP (specialized inputs)

destined to go through a training procedure. But how would culture manifest in a freer system?

A coding for neuroevolutionary algorithms was proposed in [29] in order to study the emerging patterns when there is no explicitly defined objective function and when individuals are able to interact with each other since they share the same simulation environment. Such work is a door to research the above question.

By giving agents the ability to learn from those in their surroundings and to choose whether they would do so, would it be possible to verify whether culture would arise by some interpretable pattern? Would individuals become more judgmental as to whom they would choose as tutors in advanced generations? Since individuals have different network topologies and therefore different learning capabilities, would groups of good learners emerge? All these questions motivate further

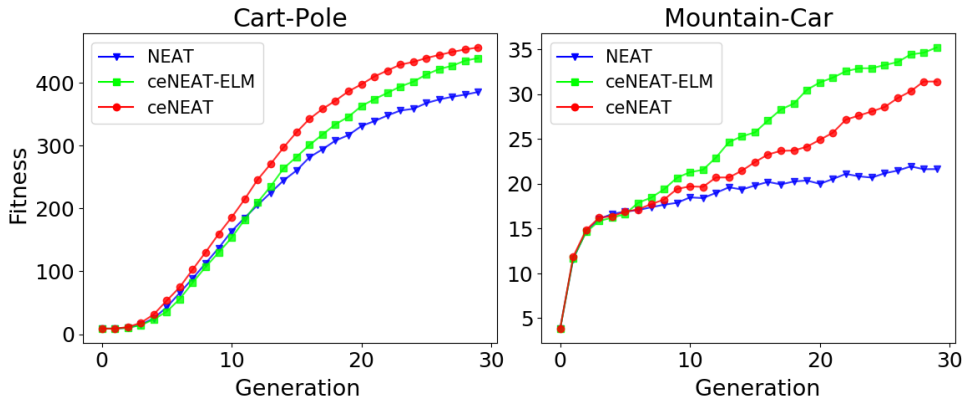

Fig. 6. Baseline results for ceNEAT-ELM

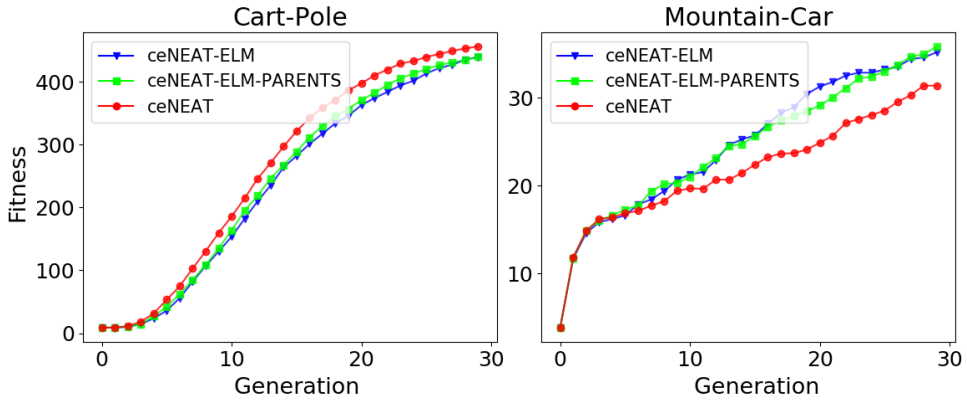

Fig. 7. Results for ceNEAT, ceNEAT-ELM (offspring training), and ceNEAT-ELM-PARENTS (parents training)

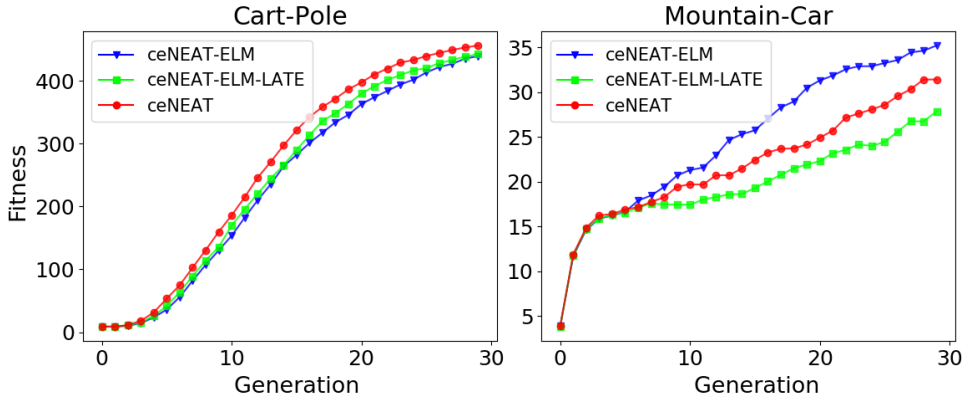

Fig. 8. Results for ceNEAT, ceNEAT-ELM (training since the beginning), and ceNEAT-ELM-LATE (training belatedly)

research in the themes of this paper.

REFERENCES

[1] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, Jun. 2002. [Online]. Available: http://dx.doi.org/10.1162/106365602320169811

[2] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, Mar 2008. [Online]. Available: https://doi.org/10.1007/s12065-007-0002-4

[3] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the nero video game," *IEEE Transactions on Evolutionary Computation*, pp. 653–668, 2005. [Online]. Available: http://nn.cs.utexas.edu/?stanley:ieeetec05

[4] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial Life*, vol. 15, no. 2, pp. 185–212, April 2009.

[5] E. J. Hastings, R. K. Guha, and K. O. Stanley, "Automatic content generation in the galactic arms race video game," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 4, pp. 245–263, Dec 2009.

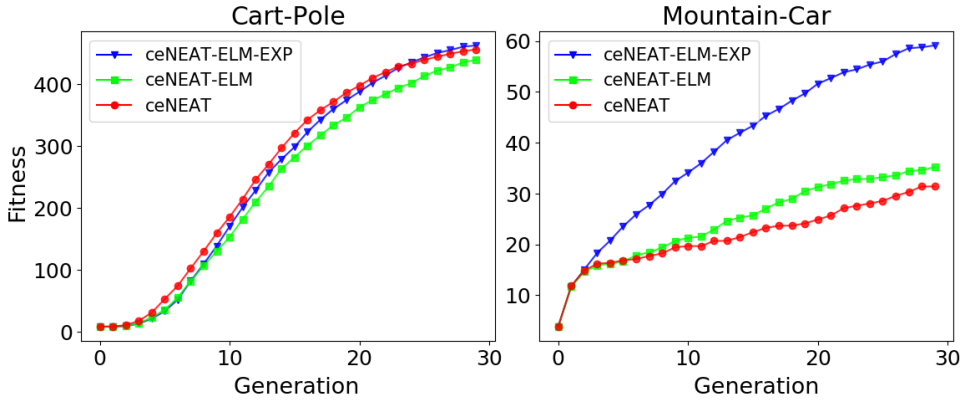

Fig. 9. Results for ceNEAT, ceNEAT-ELM (random inputs), and ceNEAT-ELM-EXP (specialized inputs)

[6] F. Silva, P. Urbano, L. Correia, and A. L. Christensen, "odneat: An algorithm for decentralised online evolution of robotic controllers," *Evolutionary Computation*, vol. 23, no. 3, pp. 421–449, 2015.

[7] R. Miikkulainen, J. Z. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat, "Evolving deep neural networks," *CoRR*, vol. abs/1703.00548, 2017. [Online]. Available: http://arxiv.org/abs/1703.00548

[8] S. Nolfi, J. L. Elman, and D. Parisi, "Learning and evolution in neural networks," *Adaptive Behavior*, vol. 3, pp. 5–28, 1990.

[9] R. K. Belew, F. Menczer, F. Cecconi, and F. Cecconi, "Maturation and the evolution of imitative learning in artificial organisms," 1981.

[10] T. Sasaki and M. Tokoro, "Adaptation toward changing environments: Why darwinian in nature?" in *In P. Husbands and I. Harvey (Eds.), Fourth European Conference on Artificial Life (ECAL97*. MIT Press, 1997, pp. 145–153.

[11] P. H. McQuesten, "Cultural enhancement of neuroevolution," Ph.D. dissertation, The University of Texas at Austin, Austin, TX 78712, 2002.

[12] C. R. Darwin, *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. United Kingdom of Great Britain and Ireland.: John Murray, November 1859.

[13] D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part 1, fundamentals," 1993.

[14] M. Melanie, *An Introduction to Genetic Algorithms*. London, England - Cambridge, Massachusetts: The MIT Press, 1999.

[15] S. Luke, *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013, available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[16] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[18] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Networks*, vol. 61, no. 1, pp. 32–48, 2015.

[19] B. Widrow, A. Greenblatt, Y. Kim, and D. Park, "The no-prop algorithm: A new learning algorithm for multilayer neural networks," *Neural Networks*, vol. 37, pp. 182–188, 2013.

[20] J. M. Baldwin, "A new factor in evolution," *The American Naturalist*, vol. 30, no. 354, pp. 441–451, 1896. [Online]. Available: https://doi.org/10.1086/276408

[21] R. C. Keesing and D. G. Stork, "Evolution and learning in neural networks: the number and distribution of learning trials affect the rate of evolution," September 1993.

[22] J.-B. P. A. de Monet de Lamarck, *Philosophie Zoologique*. Frana: Museum d'Histoire Naturelle (Jardin des Plantes), 1809.

[23] F. Gruau, D. Whitley, and L. Pyeatt, "Adding learning to the cellular development of neural networks: Evolution and the baldwin effect." *Evolutionary Computation*, vol. 1, pp. 213–233, 1993.

[24] D. Whitley, V. S. Gordon, and K. Mathias, "Lamarckian evolution, the baldwin effect and function optimization." Springer-Verlag, 1994, pp. 6–15.

[25] B. A. Julstrom, "Comparing darwinian, baldwinian, and lamarckian search in a genetic algorithm for the 4-cycle problem," in *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, 1999, pp. 134–138.

[26] K. O. Stanley, "Efficient evolution of neural networks through complexification," Ph.D. dissertation, Department of Computer Sciences, The University of Texas at Austin, 2004. [Online]. Available: http://nn.cs.utexas.edu/?stanley:phd2004

[27] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, Sept 1983.

[28] A. W. Moore, "Efficient memory-based learning for robot control," Ph.D. dissertation, University of Cambridge, Trinity Hall, 1990.

[29] Y. L. B. Nogueira, C. E. F. de Brito, C. A. Vidal, and J. B. C. Neto, "Emergent vision system of autonomous virtual characters," *Neurocomputing*, vol. 173, pp. 1851 – 1867, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231215013922