On Overview of Neuroevolution and NEAT

Paul Pauls
Advisor: Michael Adam

CONTENTS

***Abstract*—Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.**

I. INTRODUCTION

THIS shall be my introduction. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

II. NEUROEVOLUTION AND EVOLUTIONARY ALGORITHMS

Neuroevolution is a form of evolutionary algorithm that generates specific artificial neural networks (short form: ANN) through modification of its parameters, topology and rules in order to maximize the ANN's accuracy or fitness score. The neuroevolution algorithm seeks to modify the ANN in an evolutionary process similar to the Darwinian process that produced human brains and its process-summarizing maxim 'survival of the fittest'. First methods using neuroevolution can be traced back to the 1980s and 1990s [cite], while the first evolutionary algorithms were conceived in the 1950s by Alan Turing and Nils Barricelli. [cite]

To better characterize neuroevolution is it best to first roughly categorize it, whereupon the following sections describe the categories in detail. A *neuroevolution algorithm* is a *genetic algorithm*, whose search-space (or genotypes) consist only of artificial neural networks. A *genetic algorithm* in turn is a *evolutionary algorithm* that evolves genotypes - genetically encoded representations of the actual solution (phenotype) - through mutation, recombination and selection.

A. Evolutionary Algorithms

A evolutionary algorithm (short form: EA) is defined as 'a generic population-based and meta-heuristically optimized algorithmic solution to an applied problem' [cite]. When breaking this complex definition down into simpler terms, is the first thing that should be clarified about EAs the fact that they are no algorithmic solution to the applied problem in and of themselves per se, but rather that they are meta-algorithms that create another optimized algorithm, which then solves the applied problem. An evolutionary algorithm therefore encodes a method of how to come up with the best solution to an algorithmic problem.

Evolutionary algorithms set out to finding this best algorithmic solution through a *population-based* method. This means that EAs manage an arbitrary variety of algorithmic solutions - all of which differ and solve the applied problem with various grades of accuracy or fitness scoring. Each of these algorithmic solutions is called a *member* of the evolutionary algorithms' population and is potentially the best algorithmic solution - the best member in other words - that is eventually returned.

To determine the best member of the population does the evolutionary algorithm assign each member a *fitness score* after it is created. In the context of neuroevolution for example is this fitness score calculated by judging the accuracy of the artificial neural network or it is calculated by the fitness function in case of an environment embedded agent.

However, the question remains how the members of the population are created in a sensible way so that they may represent a reasonable algorithmic solution to the problem and eventually an optimized one. This is the point at which the EAs' aspect of *meta-heuristic optimization* comes into play. Each new member in a population is conceived by recombining and/or mutating a single or multiple existing members of the population. Presuming that additionally the single or multiple existing members that are recombined and/or mutated to create the new member are chosen to be the highest fitness scoring members (a.k.a. the *fittest* members), does the process constitute an optimization procedure resembling Darwinian evolution. Therefore can be said that evolutionary algorithms breed increasingly optimized algorithmic solution through evolutionary intercombination of existing algorithmic solutions - hence representing the mentioned optimization process.

The possible methods of intercombination between existing members are inspired by biological evolution, such as mutation, recombination and selection. For the sake of brevity will only the subsequent chapter II-C explain those intercombination methods in detail and how they apply to neuroevolution algorithms. The mentioned optimization process achieved through these intercombination methods is considered *meta-heuristic* because the optimization process is possible with incomplete or imperfect information or limited computational capacity. Thus even when the feedback - meaning the ability to determine a sensible fitness score through accuracy measurement or similar - of the applied problem is limited or sparse, is fruitful traversal across the search-space through e.g. a lucky mutation feasible.

Finally can be said that the evolutionary algorithms population-based methodology is considered *generic* because it does not dictate how the members of the population are encoded. Though an evolutionary algorithm needs to eventually return an algorithmic solution to the applied problem, does it not dictate that the members are algorithms in memory, nor that they even need to be in algorithmic form when they are given a fitness score. This is were genetic algorithms come into play.

B. Genetic Algorithms

Genetic algorithms (short form: GAs) are evolutionary algorithms whose members are not saved as algorithms in memory, but as genetic-like encodings which can then be translated into algorithms by a user-specified component of the genetic algorithm. The genetic-like encoded member in a GA is called *genotype*, whereas its corresponding translated algorithm is called *phenotype* To give an example, a genetic algorithm trying to find the best search algorithm would not represent a member such as the Quicksort algorithm itself in memory, but as a series of four different characters which would then in turn translate into the Quicksort algorithm. Its genotype could be a character-string such as "ACGCUG...", while its phenotype would be the explicit search algorithm in code.

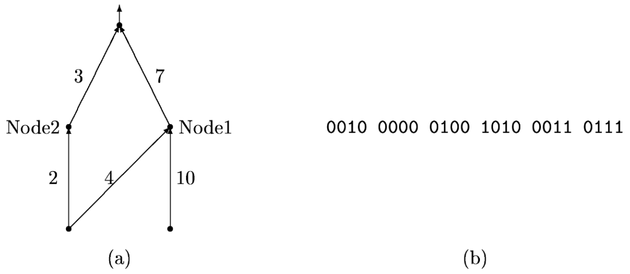

Fig. 1. Illustration of a binary encoding of an ANN. (Source: [1])

The defining advantage of genetic algorithms by representing members as a genetic-like encoding instead of an algorithm in memory is that intercombination methods like mutation and recombination - central aspects of evolutionary algorithms - are vastly easier and faster on relatively simple genetic encodings than on complex specified algorithms. Figure 2, though actually illustrating the 'Competing Conventions Problem', also illustrates well how such a complex algorithm as an artificial neural network is significantly simpler represented as a genetic encoding and how such a genetic encoding can be vastly easier recombined than the algorithm itself.

C. Neuroevolution

To come full circle is now a proper definition of Neuroevolution possible. Neuroevolution is the - possibly boundless - process in which by the means of a genetic algorithm its population of artificial neural networks is increasingly optimized in order to maximize the accuracy or fitness of the best ANN. Neuroevolution does so by continuously improving the members in its population through intercombination methods like mutation, recombination and selection.

The intercombinatory method of *mutation* in the context of Neuroevolution means that the genotype of a chosen member is in some way modified by adding to, changing or removing from the genotype representation. The manner and probability in which this modification takes place is completely up to the implementation specifics of the respective neuroevolution algorithm. To give an example mutation for an ANN, is it possible to imagine a binary encoded genotype as seen in figure 1, which then has some bits flipped, added or removed possibly resulting in a new node or connection in its corresponding phenotype.

The intercombinatory method of *recombination* in the context of Neuroevolution means that the genotype of two or more arbitrary (though most often high performing) members are combined, forming a new member. This is done in the hopes that those parts of the genotype that encode member-distinct features combine into the newly created genotype and encode an ANN that performs even better than both *parent*-members in isolation. An example of such a recombination, though an impaired one as the figure actually represents the flawed process of the 'Competing Conventions Problem', can be seen in figure 2. Again is the manner and probability in which this modification is performed completely up to the implementation specifics of the respective neuroevolution algorithm.

Lastly is the intercombinatory method of *selection*. As previously defined does the process of neuroevolution work on populations; these however are often of fixed size in most neuroevolution algorithms. The purpose of this restriction is to force the neuroevolution process to remove low performing members from the population - and therefore the gene pool - from which possible parents for intercombination are chosen, by only allowing a limited number of members to exist. Once all members of the population have been assigned a fitness score, is this current state of the population considered a specific *generation*. The method of *selection* then removes certain members of the population while the intercombinatory methods of mutation and recombination add new members to the population and the whole population is evaluated again, marking the start of the next generation. The method of selection is also applicable in the case of an unrestricted population size, e.g. by removing members that score too far below the current best member.

All those Details of the neuroevolution process, e.g. how the encoding scheme specifies genotypes and their translation into the phenotype ANNs, how the initial population is created, which members are chosen as parents for intercombinatory methods or simply how exactly the intercombinatory methods are performed are all left to the specific neuroevolution algorithm.

D. Landmark Research in Neuroevolution

A short listing of landmark research in the field of neuroevolution will close this chapter and introduce some of the most important papers in recent years. This listing will start with the publication of NEAT, as the field of neuroevolution gained significant traction from there on, moving away from the evolution of fixed-topology networks to "Topology and Weight Evolving Neural Networks" (short form: TWEANNs).

a) Neuroevolution of Augmenting Topologies (NEAT): By Stanley&Miikkulainen in 2002 [cite]. First viable TWEANN neuroevolution algorithm. Introduced meaningful protection of topological innovation and solved the 'Competing Conventions Problem' and therefore enabled meaningful recombination of genes representing different neural networks. Spawned multiple variations, of which the most noteworty are rtNEAT and odNEAT.

b) HyperNEAT: By Risi&Stanley in 2009 [cite]. Move from *direct* to *indirect* encoding. Presented genotype does not represent the nodes and connections of ANN, but a function that will generate the ANN. Extended in 2011 with the introduction of ES-HyperNEAT by Risi&Stanley [cite].

c) Novelty Search: By Lehman&Stanley in 2011 [cite]. Abandons maxim to judge the members based on their respective fitness score and replaces it with judging members based upon the novelty of the search space they explore. Though not universally applicable, does a novelty-search algorithm often reach the convergence point faster than traditional algorithms.

d) CoDeepNEAT: By Miikkulainen, Liang, et al in 2017 [cite]. Applys NEAT to Deep Neural Network with genes representing layers instead of nodes and connections. Repeated Components (as applied in ResNET, GoogleNET, etc), topologies and hyperparameters are then evolved competitively. Around this time did the consensus emerge to use backpropagation to evolve weights and evolution to evolve topology, though a mixture of both is most often applied.

e) Large-Scale Evolution of Image Classifiers: By Real, Moore, et al in 2017 [cite].

f) EvoCNN: By Sun, Xue, et al in 2017 [cite].

g) Deep Neuroevolution: By Uber-Research in 2017 [cite].

h) Hierarchical Representations for Efficient Architecture Search: By Liu, Simonyan, et al in 2018 [cite]

i) Regularized Evolution for Image Classifier Architecture Search: By Real, Aggarwal, et al in 2019 [cite]

III. NeuroEvolution of Augmenting Topologies (NEAT)

¡Section Introduction¿

Neuroevolution of Augmenting Topologies (short form: NEAT) was first introduced in the paper "Evolving Neural Networks through Augmenting Topologies" by Kenneth O.Stanley and Risto Miikkulainen in the year 2002. [cite] It was finalized and shown to be superior to any preceding neuroevolution algorithm in Stanley's PhD thesis "Efficient Evolution of Neural Networks through Complexification" in 2004. [cite]

At time of envisioning of NEAT was Neuroevolution most promising learning approach. Still is powerful today (see rea17/19) "NE is a promising approach to learning behavioral policies and finds solutions faster than leading RL methods on many benchmark tasks (Gomez 2003; Moriarty and Miikkulainen 1997)" [5]

"In highly complex domains the heuristics for determining the appropriate size are not very useful, and it becomes increasingly difficult to solve such domains with fixed-length encodings." [5]

[See all notes write down about stanleys PhD thesis]

A. Key Aspects of NEAT and Differences to Preceding Neuroevolution

B. Performance of NEAT

C. Variants and Advancements of NEAT

1) *¡Variant 1¿:*
2) *¡Variant 2¿:*
3) *¡Variant 3¿:*

IV. Practical Applications of NEAT

A. ¡Application 1¿

B. ¡Application 2¿

C. ¡Application 3¿

V. Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra

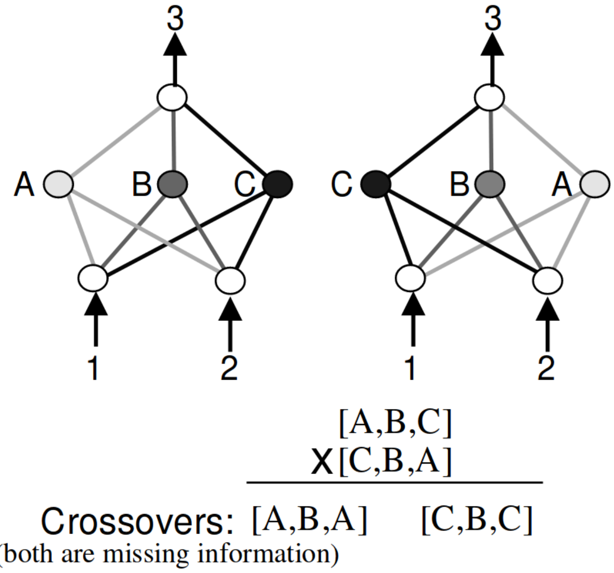

Fig. 2. Illustration of the 'Competing Conventions Problem'. (Source: [3])

sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

VI. Acknowledgments

I wish to thank Rezsa Farahani, from the Google Brain team, and Michael Adam, from the Technical University Munich, for their support and valuable input.

References

[1] Yao - Evolving Artificial Neural Networks; 1999; http://avellano.fis.usal.es/~lalonso/compt_soft/articulos/yao99evolving.pdf
[2] Stanley, Miikkulainen - Efficient Evolution of Neural Network Topologies; 2002; http://nn.cs.utexas.edu/downloads/papers/stanley.cec02.pdf
[3] Stanley, Miikkulainen - Evolving Neural Networks through Augmented Topologies; 2002; http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf
[4] Geard, Wiles - Structure and Dynamics of a Gene Network Model Incorporating Small RNAs; Dec 2003; https://ieeexplore.ieee.org/document/1299575
[5] Stanley - Efficient Evolution of Neural Networks through Complexification; Aug 2004; http://nn.cs.utexas.edu/downloads/papers/stanley.phd04.pdf
[6] Reisinger, Miikkulainen - Acquiring Evolvability through Adaptive Representations; Jul 2007; http://nn.cs.utexas.edu/downloads/papers/reisinger.gecco07.pdf
[7] Mattiussi, Duerr, et al - Center of Mass Encoding: A self-adaptive representation with adjustable redundancy for real-valued parameters; Jul 2007; https://infoscience.epfl.ch/record/101405
[8] Floreano, Duerr, et al - Neuroevolution: From Architectures to Learning; Jan 2008; http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.182.1567
[9] Mattiussi, Marbach, et al - The Age of Analog Networks; Sep 2008; https://www.aaai.org/ojs/index.php/aimagazine/article/view/2156
[10] Stanley, D'Ambrosio, et al - A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks; 2009; http://axon.cs.byu.edu/~dan/778/papers/NeuroEvolution/stanley3**.pdf
[11] Risi, Stanley - Enhancing ES-HyperNEAT to Evolve More Complex Regular Neural Networks; Jul 2011; http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.365.4332
[12] Lehman, Stanley - Novelty Search and the Problem with Objectives; Oct 2011; https://www.cs.ucf.edu/eplex/papers/lehman_gptp11.pdf
[13] Woergoetter, Porr - Scholarpedia Article on 'Reinforcement Learning'; Sep 2012; http://www.scholarpedia.org/article/Reinforcement_learning
[14] Holland - Scholarpedia Article on 'Genetic Algorithms'; Oct 2012; http://www.scholarpedia.org/article/Genetic_algorithms
[15] Fogel, Fogel, et al - Scholarpedia Article on 'Evolutionary Programming'; Oct 2013; http://www.scholarpedia.org/article/Evolutionary_programming
[16] Lehman, Miikkulainen - Scholarpedia Article on 'Neuroevolution'; Oct 2013; http://www.scholarpedia.org/article/Neuroevolution
[17] Pascanu, Ganguli, et al - On the Saddle Point for Non-Convex Optimization; May 2014; https://www.researchgate.net/publication/262452520
[18] Kim, Rigazio - Deep Clustered Convolutional Kernels; Mar 2015; https://arxiv.org/abs/1503.01824
[19] Fernando, Banarse, et al - Convolution by Evolution; Jun 2016; https://arxiv.org/abs/1606.02580
[20] Miikkulainen, Liang, et al - Evolving Deep Neural Networks; Mar 2017; https://arxiv.org/abs/1703.00548
[21] Xie, Yuille - Genetic CNN; Mar 2017; https://arxiv.org/abs/1703.01513
[22] Negrinho, Gordon - DeepArchitect: Automatically Designing and Training Deep Architectures; Apr 2017; https://arxiv.org/abs/1704.08792
[23] Real, Moore, et al - Large-scale Evolution of Image Classifiers; Jun 2017; https://arxiv.org/abs/1703.01041
[24] Stanley - Neuroevolution: A Different Kind of Deep Learning; Jul 2017; https://www.oreilly.com/ideas/neuroevolution-a-different-kind-of-deep-learning
[25] Brock, Lim, et al - SMASH: One-Shot Model Architecture Search through HyperNetworks; Aug 2017; https://arxiv.org/abs/1708.05344
[26] Salimans, Ho - Evolution Strategies as a Scalable Alternative to Reinforcement Learning; Sep 2017; https://arxiv.org/abs/1703.03864
[27] Jaderberg, Dalibard, et al - Population Based Training of Neural Networks; Nov 2017; https://arxiv.org/abs/1711.09846
[28] Zhang, Clune, et al - On the Relationship Between the OpenAI Evolution Strategy and Stochastic Gradient Descent; Dec 2017; https://arxiv.org/abs/1712.06564
[29] Stanley, Clune - Welcoming the Era of Deep Neuroevolution; Dec 2017; https://eng.uber.com/deep-neuroevolution/
[30] Liu, Simonyan, et al - Hierarchical Representation for Efficient Architecture Search; Feb 2018; https://arxiv.org/abs/1711.00436
[31] Such, Stanley, et al - Accelerating Deep Neuroevolution: Train Atari in Hours on a Single Personal Computer; Apr 2018; https://eng.uber.com/accelerated-neuroevolution/
[32] Such, Madhavan, et al - Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning; Apr 2018; https://arxiv.org/abs/1712.06567
[33] Zoph, Vasudevan, et al - Learning Transferable Architectures or Scalable Image Recognition; Apr 2018; https://arxiv.org/abs/1707.07012
[34] Lehman, Chen, et al - ES Is More Than Just a Traditional Finite-Difference Approximator; May 2018; https://arxiv.org/abs/1712.06568
[35] Lehman, Chen, et al - Safe Mutations for Deep and Recurrent Neural Networks through Output Gradients; May 2018; https://arxiv.org/abs/1712.06563
[36] Zhong, Yan, et al - Practical Block-Wise Neural Network Architecture Generation; May 2018; https://arxiv.org/abs/1708.05552
[37] Rawal, Miikkulainen - From Nodes to Networks: Evolving Recurrent Neural Networks; Jun 2018; https://arxiv.org/abs/1803.04439
[38] Conti, Madhavan, et al - Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty Seeking Agents; Oct 2018; https://arxiv.org/abs/1712.06560
[39] Frolov - Neuroevolution: A Primer on Evolving Artificial Neural Networks; Oct 2018; https://www.inovex.de/blog/neuroevolution/
[40] Real, Aggarwal, et al - Regularized Evolution for Image Classifier Architecture Search; Feb 2019; https://arxiv.org/abs/1802.01548
[41] Sun, Xue, et al - Evolving Deep Convolutional Neural Networks for Image Classification; Mar 2019; https://arxiv.org/abs/1710.10741