Directed Study – A1

Jéssica Pauli de C. Bonson (B00617515)

Survey the NEAT algorithm

NeuroEvolution of Augmenting Topologies (NEAT) [1] is a type of Topology and Weight Evolving Artificial Neural Networks (TWEANNs), which is based on three key ideas: historical marking, speciation, and complexification.

NEAT uses direct encoding, where genomes contains a list of connection genes, each connection gene specify the nodes being connected, the weight, if the connection is enabled, and an innovation number. Innovations numbers are assigned to new genes, through a global counter, they are used to define if a gene has the same historical origin that another, meaning that they represent the same structure. The innovation numbers also are used to identify homological genes during the crossover, allowing to crossover even between genomes of different configurations, the crossover can happen intra or inter species.

Those historical markings are also used for speciation in NEAT, where the number of excess and disjoint genes, along with the average weight differences, are used to define the compatibility between genes with little computation cost. According to a compatibility threshold the algorithm is then able to speciate the genomes, it protects topological innovations, which usually would be discarded because of an initial lack of fitness. Genomes just compete against other members of their specie and the fitness function is specific for their niche. An advantage of this model is that the algorithm has a lower chance of getting stuck in local minimas, because it evolves diverse topologies at the same time [1].

The third key idea behind NEAT is that it starts minimally, and then increasingly complexify the network, expanding the search space when its beneficial, at the same time that optimizes the solution. The initial network topology contains zero hidden nodes, and all inputs are connect with all outputs. Doing it NEAT searches through fewer dimensions of the search space than TWEANNs and fixed-topology NE systems, having a performance advantage over them [1]. Since the algorithm develops the network topology from scratch, it isn't necessary previous knowledge about the network's shape or search space's dimensions.

Mutations in NEAT can affect either weights or structures. Weights have a 90% chance of being uniformly perturbed and a 10% chance of receiving a new random value. Structural mutations can add connections or nodes to a network. Because of speciation, the algorithm is tolerant to frequent mutations.

NEAT is able to develop increasingly more sophisticated strategies, because the dominant strategies can be elaborated over time by adding new structures, which results in a continual coevolution. Because of this feature the algorithm is powerful in open-ended domains, like competitive coevolution, and it was demonstrated in [2] that it can generate pretty sophisticated strategies for the robot duel domain.

According to [3] NEAT has a great performance at complex control tasks, like double pole balancing and robotic strategic-learning, and it has the best performance regarding problems that require reactive control, especially when they have small solutions. The studies in [1] shows that NEAT outperforms the algorithms SANE [4] and ESP [5] in the domain of double pole balancing with velocity information, achieving faster solutions with minimal topologies. It also has a much better performance than the algorithms CE [6] and ESP in the domain of double pole balancing without velocity information.

Besides these advantages, [3] states that since the NEAT algorithm is limited to small and incremental changes in the network structure it has problems to solve tasks that require complicated or repeated internal structure. The authors classify the tasks that NEAT has difficult solving as fractured problems, which are high-level strategic problems where the optimal actions change repeatedly and discontinuously as the agent moves through states. Some examples of such problems are concentric spirals classification, multiplexer, and high-level decision making in general.

Adaptations in the algorithm can improve its performance in relation to this sort of problem. [3] describes three of such algorithms: RBF-NEAT, Cascade-NEAT and SNAP-NEAT. RBF-NEAT modifies the original algorithm by adding an “add RBF node” topological mutation, this new node is activated by an axis-parallel Gaussian with variable center and size, and is connected to all input and output nodes. With this modification the algorithm bias the networks toward local-processing structures, improving its performance in solving fractured problems.

In Cascade-NEAT the main idea is to constrain the search process to topologies that have a cascaded structure. To achieve it there is a new structural mutation, “add cascade node”, which adds a new node that is connected to all inputs, outputs and all previous hidden nodes. When a node is added it freezes the previous network structure, focusing the search on the connections of the newest added node.

According to [3], both RBF-NEAT and Cascade-NEAT have a higher performance in fractured problems than NEAT, but NEAT still performs better in low-level control tasks. The SNAP-NEAT algorithms goal is to be a general approach that can be applied to a variety of problems. It is realized by creating an algorithm that is able to choose between the mutation operators of the three previous algorithms when they are most effective, the selection algorithm is a modified version of the adaptive pursuit algorithm with an initial estimation period. The results at [3] show that SNAP-NEAT is able to intelligently select the best operators for the problem it faces, but for the non-Markov double pole-balancing problem the original NEAT algorithm is still the best one.

References:

[1] K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” Evol. Comput., vol. 10, no. 2, pp. 99–127, 2002.
[2] K. O. Stanley and R. Miikkulainen, “Competitive coevolution through evolutionary complexification,” J. Artif. Intell. Res., vol. 21, no. 1, pp.
63–100, 2004.
[3] N. Kohl and R. Miikkulainen, “An Integrated Neuroevolutionary Approach to Reactive Control and High-Level Strategy”, 2012.
[4] D. E. Moriarty and R. Miikkulainen. “Efficient reinforcement learning through symbiotic evolution”. Machine Learning, 22:11–32, 1996.
[5] F. Gomez and R. Miikkulainen, “Solving non-Markovian control tasks with neuroevolution.” In Dean, T., editor, Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pages 1356–1361, Morgan Kaufmann, San Francisco, California, 1999
[6] F. Gruau, D. Whitley and L. Pyeatt, “A comparison between cellular encoding and direct encoding for genetic neural networks.” In Koza, J. R. et al., editors, Genetic Programming 1996: Proceedings of the First Annual Conference, pages 81–89, MIT Press, Cambridge, Massachusetts, 1996.