Braincraft: Neuroevolution for the Masses

Undergraduate Researchers: Christopher Donahue, Prad Nelluru, Minwoo Bae, Leah Hudson, Taylor Rose, Kristi Park, Katie Park
Research Educators: Uli Grasemann, Igor Karpov
Faculty Lead: Risto Miikkulainen
Department of Computer Science, College of Natural Sciences, The University of Texas at Austin

Summary

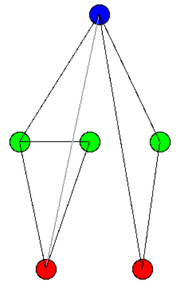
Chris Donahue

Neuroevolution is the process of iteratively constructing neural networks to find a solution network for a given problem. Braincraft is an approachable Java library for experimentation with neuroevolution techniques. Users can run an evolution experiment on the Braincraft platform with minimal knowledge of the library's inner workings. The algorithms behind Braincraft are implemented in a generic way so that they can be adapted to any problem. Users receive detailed feedback from the library and can even save, view, and test any of the produced neural networks. Braincraft grants users access to powerful neuroevolution techniques regardless of skill level.

Using Braincraft

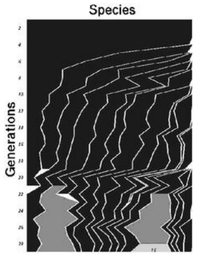
How does one use Braincraft?

Braincraft is designed to be extremely simple to use. To run an experiment, the user tells the library the number of inputs (sensors) and outputs (actions) for the neural networks they wish to test. The library will create a population of neural networks based on these specifications. After testing a network, the user reports a fitness value back to Braincraft which it will eventually use to create the next generation of networks. Several examples come bundled with the library to demonstrate this workflow. The process is simple and intuitive, giving it an advantage over approaches taken in the past which were often too intimidating to users less educated in neuroevolution.



Visualizing a neural network

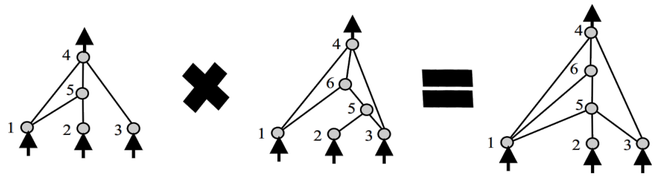
The NEAT Algorithm



Visualizing Species development in NEAT

What is NEAT?

NEAT is a neuroevolution algorithm written by Ken Stanley and published in 2002. Like most neuroevolution algorithms it produces a generation of neural networks, accepts fitness values for those networks from the user and then selectively breeds them to create the next generation. However, NEAT partitions the neural networks in a given generation into different "species", an abstraction designed to protect new neural innovations. The genetic algorithm used to cross two neural networks in NEAT utilizes historical innovation markers created whenever a new structural mutation occurs.



Crossing two neural networks with NEAT

How does it apply to Braincraft?

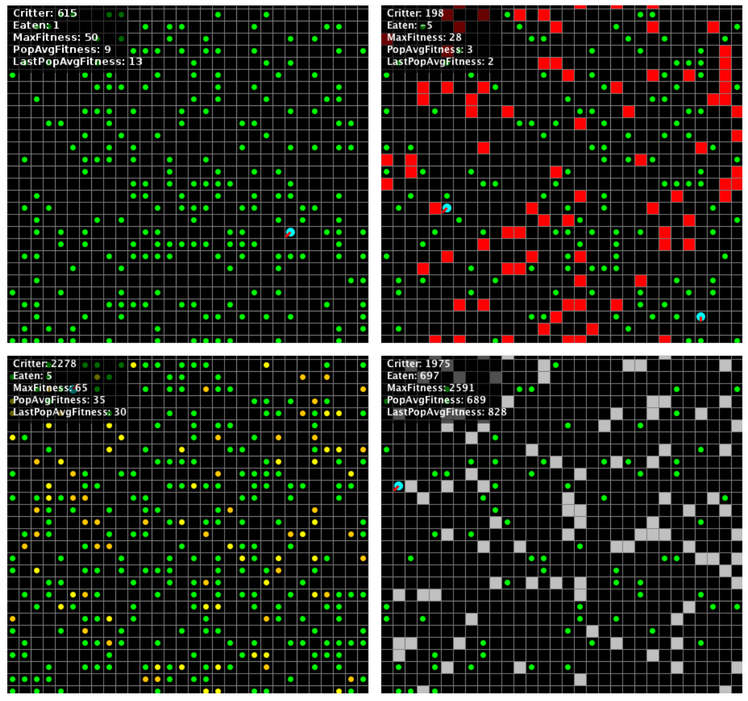
Braincraft owes a great deal to the NEAT algorithm. The platform began its life as a Java implementation of NEAT but then itself evolved. Now, the user can use NEAT or choose to write their own method of population control and test it on any existing Braincraft experiment. This allows for powerful codevelopment of a neuroevolution experiment on both the frontend and backend. Braincraft continues to utilize NEAT-style innovation markers to cross networks.

Aegis

Prad Nelluru, Chris Donahue

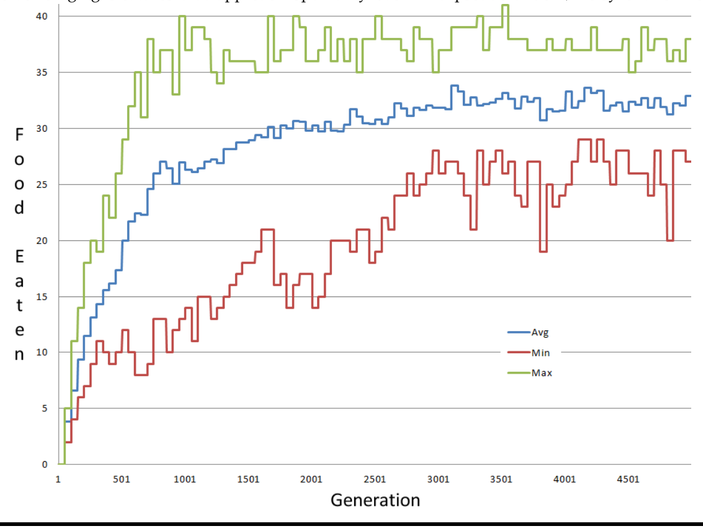
What is Aegis?

Aegis is a simple foraging experiment written on the Braincraft platform and designed to be a teaching tool for introducing neuroevolution. Aegis was given to the students in the Spring 2012 *Computational Intelligence in Game Design* research stream. Students were asked to extend the experiment by adding additional challenges for the evolving foragers.



Four different experiments running in Aegis. From top-left and clockwise: default foraging experiment, lava and simultaneous foragers (Kristi Park, Katie Park), adding walls (Kristi Park, Katie Park), different types of food (Leah Hudson, Taylor Rose)

The core foraging experiment of Aegis was run on the *UTCS Mastodon High Throughput Computing Cluster*. The results (charted below) suggest that Braincraft running the NEAT algorithm can evolve foraging creatures that approach optimality even in a pseudorandom, noisy environment.

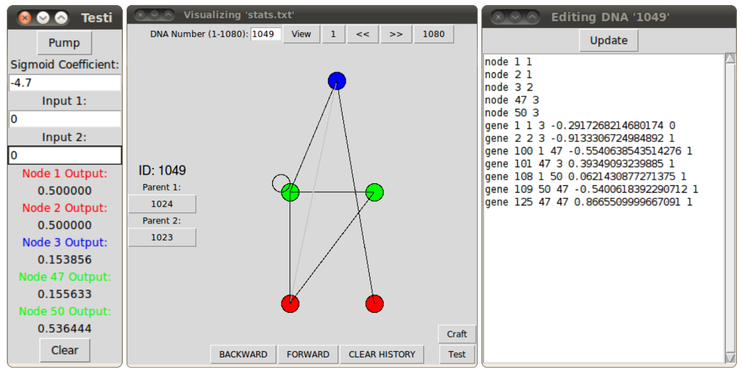


BRAINCRAFT

Network Visualizer

Chris Donahue

Braincraft comes bundled with a visualizer for the neural networks it produces. The tool allows the user to see, test and edit the networks that the library produced. An edited network can then be used by the library as a seed structure.



Testing, visualizing and editing a network that approximates a two-bit AND gate

Potential Applications

Braincraft is a flexible library with a wide variety of potential uses. Because it is written in Java, Braincraft lends itself nicely to the development of software for the Android operating system. It is also being employed for experimentation with signal processing. A plugin is currently being developed to create audio filters and synthesizers using Braincraft and the Steinberg VST format.

Related Publications

D. E. Goldberg, and J. Richardson (1987). Genetic Algorithms With Sharing for Multimodal Function Optimization. *Proceedings of the Second International Conference on Genetic Algorithms*, 148-154, 1987.
F. Gruau (1993). Genetic Synthesis of Modular Nueral Networks. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 318-325, 1993.
K. O. Stanley, and R. Miikkulainen (2002). Evolving Neural Networks Through Augmenting Topologies. *Evolutionary Computation*, 10(2):99-127, 2002.
K. O. Stanley, B. D. Bryant, and R. Miikkulainen (2005). Real-Time Neuroevolution in the NERO Video Game. *IEEE Transactions on Evolutionary Computation*, 9(6):653-668, 2005.

Acknowledgements

This research is supported in part by the Microsoft summer research fellowship and by the College of Natural Sciences at the University of Texas at Austin. Thanks to Igor Karpov for the LaTeX poster template.

Updated April 6, 2012