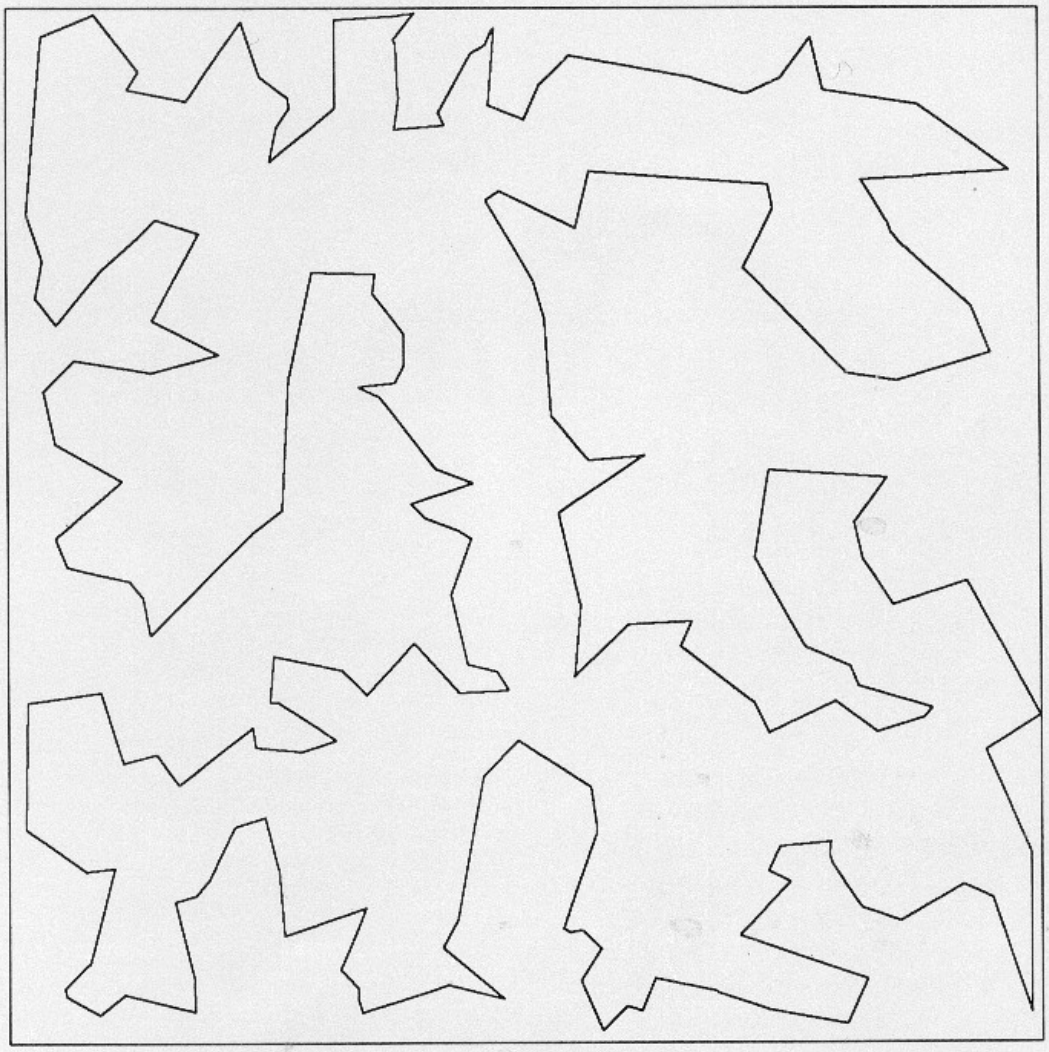
NEURAL NETWORK REVIEW

Volume 4, Number 1, 1990



LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS
Hillsdale, New Jersey Hove and London

MA 02138.

References

Fahlman, S. (1988). Faster-learning variations on back-propagation: An empirical study. *Proceedings of the 1988 Connectionist Models Summer School* (pp. 38-51). San Mateo, CA: Morgan Kaufmann.

Harp, S., Samad, T., & Guha, A. (1989). Towards the genetic synthesis of neural networks. *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 360-369). San Mateo, CA: Morgan Kaufmann. Fairfax, VA, June 4-7.

Mesard, W., & Davis, L., (1989). Backpropagation in a genetic search environment. Poster presentation at the conference on Neural Information Processing Systems. Denver, CO, November 27-30.

Whitley, D., Starkweather, T., & Bogart, C. (in press). Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel Computing.*

Genetic Neural Networks Can Be Dynamic Too, You Know!

Optimizing small neural networks using a distributed genetic algorithm. Darrell Whitley and Timothy Starkweather. *Proceedings of the International Joint Conference on Neural Networks, 1,* 206-209. Hillsdale, NJ: Lawrence Erlbaum Associates. Washington, DC, January 15-19, 1990.

Reviewed by Hugo de Garis

Whitley and Starkweather's paper is one of a growing number in a new branch of neural networks that uses genetic algorithms to help design neural network architectures. A recent bibliography (Rudnick, 1990) on the genetic algorithm-neural network marriage has nearly 80 references. The recent neural network conference in Washington, DC in January, 1990 devoted a whole session to the evolutionary aspect of neural nets. Obviously, interest in genetic algorithms on the part of neural network researchers is growing. Unfortunately, since the branch is new, there is as yet no generally accepted name either for neural networks evolved with genetic algorithms, nor for the branch itself. Personally, I label a neural network constructed by means of a genetic algorithm a "GenNet," and the new branch (studying GenNets and how to put them together) "genetic programming" (de Garis, 1990).

Having read Whitley & Starkweather's paper several times, my overall impression was, "What's an essentially genetic algorithms paper doing in a neural network conference proceedings?" The essence of their message is that the parallelized version of their GENITOR variation of the genetic algorithm works better than the standard version (to be found in Goldberg, 1989, for example). (I recommend Goldberg's book as an excellent and readable introduction to the principles of genetic algorithms—the best in the literature.) Whitley and Hanson wrote a longer (6-page instead of 4-page) version of this paper for the Third International Conference on Genetic Algorithms (Whitley & Hanson, 1989) which says much the same things. I suppose that Whitley and Starkweather thought that if they applied their improved genetic algorithm to a neural network application (rather than to the usual, non-neural network optimization problems) it might get accepted at a neural network conference. Well, it did, but that doesn't change my opinion that their 4-page paper is a genetic algorithms paper in disguise. (Publish or perish, right?!) Still, it's not a bad paper. It is well written, clear, and reads easily. I had (I hope) no trouble with it (as a genetic algorithms paper!). What they are saying is that the standard genetic algorithm can be improved by replacing the traditional approach to crossover (i.e., creating two progeny by swapping portions of two parent chromosomes, which are destroyed in the process) by allowing the two parents to continue to exist in the population, and randomly choosing one of the two (crossed) progeny and using it to replace the lowest ranking chromosome in the population.

There is a second variation in the GENITOR approach. Instead of parents reproducing the next generation with a probability proportional to their fitness (i.e., a measure of the quality of the "solution" to the problem encoded in the chromosome, which is usually a binary string), they do so in proportion to their rank. Whitley & Starkweather make no mention of the usual technique of scaling as a means to prevent premature convergence. (Scaling is linearly transforming fitness scores such that the best (transformed) value is a (user specified) constant times larger than the average value.) Scaling (and ranking) prevents early high fliers from squeezing out other members of the population prematurely. Whitley & Starkweather's parallelized version of GENITOR (i.e. GENITOR II) uses quasi-isolated subpopulations, each with its own genetic algorithm. Occasionally, each subpopulation sends its best chromosome to another subpopulation (round robin style). The justification for all of the above is to retain genetic diversity in the whole population.

I was impressed by their results. They definitely got better evolutionary speeds and higher quality solutions than the traditional approach, and these results motivate me to want to try their ideas in my own work. However, their work raises two issues of importance. One involves

the usefulness of crossover. I use the genetic algorithm to find the signs and weights of fully connected networks (GenNets) such that their output behavior over time is maximized. In practice, I find that crossover so lowers the fitness values of chromosomes that I do not use it. I make do with mutation and selection. Since it is likely that many researchers will be getting into genetic programming over the next year or so and that the concept of crossover is fundamental to genetic algorithm theorists, the issue as to whether crossover helps or hinders the evolution of fully connected GenNets is one which needs to be settled. I have already discussed this issue with Ken DeJong of George Mason University in Fairfax, Virginia, one of the leading genetic algorithm experts, but so far, we do not agree.

A second issue is a global one that involves the whole of the genetic programming community (as far as it exists). If one goes to the trouble of creating a holy alliance between genetic algorithms and neural networks, why restrict oneself to neural network applications which are boring old (static, feedforward) vector mappings when one can just as easily employ genetic algorithms on fully connected GenNets which are dynamic, i.e., they behave over time? Genetic algorithms are indifferent to what the fitness value actually measures (whether it is the quality of a static vector mapping or of a time-dependent output behavior). To a genetic algorithm, the fitness (or ranking) is merely a guide to reproductive probability. By using a genetic algorithm on fully connected GenNets, I have been able to evolve GenNet modules with specific behaviors over time. For example, imagine you wanted to evolve a GenNet output which oscillates with a given amplitude and a period of 50 cycles. You let the GenNet run for 50 cycles (with given initial conditions) and measure the output at each cycle. The fitness value might be the inverse of the sum of the squares of the differences between the desired sinusoidal value and the actual output value at each cycle. After a few hundred generations (mutating the bits in the chromosomes representing the signs and weights of the connections) you will get an actual output curve which is very close to the desired sinusoid.

Since my GenNets are fully connected, where each connection can be either excitatory or inhibitory and can take any weight value between (let us say) +1 and −1, there is a huge number of degrees of freedom to play with, especially over hundreds of cycles. The power of genetic algorithms is that they can provide solutions to problems which are too complex to analyze with other neural network methods. If one can evolve one time-dependent GenNet, one can evolve many of them, each with its own specific behavior over time. For example, I have evolved GenNets which are frequency generators, frequency detectors, leg controllers, motion controllers, antenna rotators, signal strength detectors, production rule implementors, etc.—all with the aim of showing that GenNets can be put together to build artificial nervous systems. Building artificial nervous systems (brain building) is a challenge far more worthy of the holy genetic algorithm-neural network alliance than constructing static vector mappings. A related challenge will be to put these ideas into robotics, and thus we shall probably see genetically programmed robots within one to two years.

However, the future of genetic programming is even more glorious. Within a human generation, future technologies such as wafer scale integration, molecular electronics, nanotechnology, and quantum computing will provide the neural network community with machines with millions, billions, and trillions of processors. These processors will have to self-organize (you can not program them all), but how do you know they have self-organized "well"? What does well mean? In the future, the only way to proceed may be by taking the evolutionary path by putting populations of computer architectures through a Darwinian "survival-of-the-fittest" gauntlet. Genetic algorithms may be implemented directly in hardware, thus creating the concept of the Darwin Machine.

I would like to conclude by predicting that within two years, no self-respecting neural network researcher will dare divulge to his or her colleagues that he or she is still ignorant of the basic principles of genetic algorithms. I believe that the evolutionary approach not only will become very important for neural networks, but will probably become dominant, and not just for neural networks, but for computer science as a whole. In this light, it is a pity that the organizing committee for the International Joint Conference on Neural Networks in June, 1990 in San Diego decided in its wisdom (or ignorance of genetic algorithms?) not to have a special session on genetic programming at its summer conference.

Hugo de Garis is at the CADEPS AI Research Unit, University of Brussels, Universite Libre de Bruxelles, Ave F.D. Roosevelt 50, C.P. 194/7, B-1050, Brussels, Belgium.

References

de Garis, H. (1990). Genetic programming: Modular neural evolution for Darwin machines. *Proceedings of the International Joint Conference on Neural Networks, 1,* 194-197. Hillsdale, NJ: Lawrence Erlbaum Associates. Washington, DC, January 15-19. Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison Wesley Publishing Co.

Rudnick, M. (1990). *A bibliography of the intersection of genetic search and artificial neural networks* (Tech. Rep. CS/E 90-001). Corvallis, OR: Oregon Graduate Institute, Department of Computer Science and Engineering.

Whitley, D., & Hanson, T. (1989). Optimizing neural networks using faster, more accurate genetic search. *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 391-396) San Mateo, CA: Morgan Kaufman. George Mason University, Fairfax, VA, June 4-7.

The Author Responds

The review of my paper offers several criticisms that I find hard to understand. First, there is similarity between the paper published in the *Proceedings of the Third International Genetic Algorithm Conference* (Whitley & Hanson, 1989) and the paper published in the *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, but there were also major differences. In the earlier paper, we were not able to reliably solve relatively small neural net optimization problems. In the IJCNN paper, we had solved this problem for small nets (e.g., up to 50 connections) by using a distributed genetic algorithm that shares information between subpopulations. The application was the same, but the approach was new and the results were new.

Second, I fail to see how it is that a report on using a genetic algorithm to optimize small neural networks is a genetic algorithm paper and not a neural network paper.

The bulk of de Garis' review has more to do with his own IJCNN paper (de Garis, 1990) and his "vision" of merging genetic algorithms and neural networks than with the subject of my paper. I hesitate to engage in a diatribe, but certain issues need to be clarified.

First, I think genetic algorithms have the potential of making a significant contribution to the field of neural networks. Second, I also agree that merely applying genetic algorithms to feedforward networks is not the ultimate goal of researchers interested in merging genetic algorithms and neural networks, but I see nothing wrong with using this as a starting point. My co-author and I have indicated in a recent article reviewing the application of genetic algorithms to neural network optimization problems (Whitley et al., in press), that the real attraction of genetic algorithms is that they make so few assumptions about the problem being optimized, and thus can potentially be applied in many areas. They have the potential to be used for optimizing recurrent networks, networks that use other than sigmoid transfer functions, or networks that use some form of reinforcement learning (where the behavior of the system is evaluated to obtain an error term for feedback) as opposed to supervised learning. There has also been work using genetic algorithms to define the connectivity of neural networks (Harp et al., 1989; Miller et al., 1989; Whitley & Bogart, 1990).

On a technical point, there is very good reason why Ken DeJong disagrees with de Garis concerning the role of mutation versus crossover in genetic algorithms. Using only mutation, the strings in the population cannot share information. Furthermore, the use of mutation alone is inconsistent with the fundamental theory of genetic algorithms (Holland, 1975), which shows how reproduction and recombination (crossover) alter the representation of hyperplane samples in the population. It is reproduction and recombination that allow a genetic algorithm to produce a global search of a problem space. Without crossover, one has a fancy stochastic hillclimber doing an unguided local search from several different points in the search space. This is not to say that mutation alone does not do the job; rather, it says that the problem space is not so complex as to require the full strength of a genetic search. We have recently obtained very good results using a search that also relies predominantly on mutation, for optimizing large feedforward nets (hundreds of connections). We also have good theoretical explanations, however, for why the search is effective, and in fact, we have shown why crossover can indeed sometimes cause problems when optimizing neural networks. The problem is that recombining strings that directly encode the network's weights results in the recombination of functionally dissimilar networks. This recombination results in a high variance in the string evaluation, which in turn means that the genetic algorithm is receiving inconsistent feedback about what are good regions in the search space. Solving this problem will clear the way for using genetic algorithms (with crossover) to globally optimize various types of neural networks (Whitley & Bogart, 1990). Finally, I think a more objective, less euphoric vision of the future than we find in de Garis' review is needed if genetic algorithms are to be taken seriously. Genetic algorithm researchers need to show that they indeed have something to offer which is tangible and effective; we also need to demonstrate that genetic algorithms fill a significant void in neural network research.

I do not think that this has been demonstrated yet. I do agree that this contribution is most likely to be associated with networks which cannot be trained using backpropagation. But given the results that we have obtained and the results reported by other researchers (including de Garis), I think that one short-term contribution may be an accumulation of data which indicates that global genetic search is not required to optimize these problems. This evidence coupled with the theory of genetic algorithms suggests that simple stochastic hillclimbing methods exist that will solve these problems using a local search.

Darrell Whitley is at the Department of Computer Science, Colorado State University, Fort Collins, CO 80523.

de Garis, H. (1990). Genetic programming: Modular neural evolution for Darwin machines. *Proceedings of the International Joint Conference on Neural Networks, 1,* 194-197. Hillsdale, NJ: Lawrence Erlbaum Associates. Washington, DC, January 15-19.

Harp, S., Samad, T., & Guha, A. (1989). Towards the genetic synthesis of neural networks. *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 360-369). San Mateo, CA: Morgan Kaufmann. George Mason University, Fairfax, VA, June 4-7.

Holland, J. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Miller, G., Todd, P., & Hegde, S. (1989). Designing neural networks using genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 379-384). San Mateo, CA: Morgan Kaufmann. George Mason University, Fairfax, VA, June 4-7.

Whitley, D., & Hanson, T. (1989). Optimizing neural networks using faster, more accurate genetic search. *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 391-396). San Mateo, CA: Morgan Kaufmann. George Mason University, Fairfax, VA, June 4-7.

Whitley, D., & Bogart, C. (1990). The evolution of connectivity: Pruning neural networks using genetic algorithms. *Proceedings of the International Joint Conference on Neural Networks, 1,* 134-137. Hillsdale, NJ: Lawrence Erlbaum Associates. Washington, DC, January 15-19.

Whitley, D., Starkweather, T., & Bogart, C. (in press). Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel Computing.*