Designing Application-Specific Neural Networks using the Structured Genetic Algorithm.*

Dipankar Dasgupta and Douglas R. McGregor.
dasgupta@cs.strath.ac.uk and douglas@cs.strath.ac.uk
Department of Computer Science
University of Strathclyde
Glasgow G1 1XH
U. K.

Abstract

We present a different type of genetic algorithm called the Structured Genetic Algorithm (sGA) for the design of application-specific neural networks. The novelty of this new genetic approach is that it can determine the network structures and their weights solely by an evolutionary process. This is made possible for sGA primarily due to its redundant genetic material and a gene activation mechanism which in combination provide a multi-layered structure to the chromosome. In this paper, we focus on the use of this learning algorithm for automatic generation of a complete application specific neural network. With this approach, no a priori assumptions about topology are needed and the only information required is the input and output characteristics of the task. The empirical studies show that sGA can efficiently determine the network size and topology along with the optimal set of connection weights appropriate for desired tasks, without using back-propagation or any other learning algorithm.

1 Introduction

The use of genetic algorithms (GA)(Holland, 1975; Goldberg, 1989) for designing the neural networks encompasses three major features. (1) It automatically discovers (by global search) the optimized network architecture for performing given tasks in which researchers had otherwise manually carry out trial- and- error processes to find near-optimal network architecture. (2) It is similar to biological process in that the blue print is encoded in the

*In Proceedings of COGANN-92 (International Workshop on Combinations of Genetic Algorithms and Neural Networks), June 6, 1992, USA. Editors: L.D. Whitley and J.D. Schaffer. Publisher : IEEE Computer Society Press.

chromosome. This is modified through evolutionary process, and the expressed phenotype based on information encoded in the chromosome represents a neural network structure.
(3) Genetic algorithms make almost no assumptions about the problem space it is searching (e.g. it does not require any gradient information for neural network learning). The combination of genetic search and connectionist computation seems to be a very natural one, which has already become popular and exhibits some success in different applications. However, most of the genetic-neural work so far has usually taken one of the two forms:

- the overall designing of neural network architectures using genetic algorithms (Harp et al., 1989; Whitley et al., 1990; Harp et al., 1990; Miller et al., 1989; Maričič and Nikolov, 1990) that employ backpropagation(BP) learning algorithm for finding set of connection weights of the architecture to perform specific tasks.

- the determining of an optimal set of connection weights using genetic algorithms, namely, versions of GENITOR, GENESIS etc. (Whitley and Hanson, 1989; Whitley and Starkweather, 1990; Montana and Davis, 1989; Chalmers, 1990; de Garis, 1990) for fixed predetermined network structures (i.e. defined number of nodes and their connectivities). Moreover, Kitano used GA-BP(Kitano, 1990b) for neural network weight optimization. Specifically, he used GA to locate a point in weight-bias space which is a near-optimal solution; then the backpropagation algorithm was used to conduct an efficient local search for fine-tuning of weights and biases.

Recently there have been a few studies on the use of genetic algorithms for designing neural networks without using back propagation (Marti, 1992; Hintz and Spofford, 1990; Koza and Rice, 1991). Our method is an alternative approach with distinct following feature.

The Structured Genetic Algorithm (Dasgupta and McGregor, 1992d) defines the network configuration and its connection weights in its chromosome and both the sets of parameters are optimized simultaneously in single evolutionary process. So the Structured Genetic Algorithm, while searching for optimal topology, simultaneously searches for set of optimal connection weights in the population in every generation, resulting in smaller networks.

2 Structured Genetic Algorithm(sGA)

The central feature of sGA is its use of genetic redundancy and quasi-hierarchical structure in its genotype. The primary mechanism for eliminating the conflict of redundancy is through regulatory genes (as in biological systems) which act as switching operators to turn set of genes *on* and *off* respectively.

The sGA uses haploid genetic model, the chromosome is represented as a set of (binary) strings. The model also uses conventional genetic operators and the *survival of the fittest* criterion to evolve increasingly fit individual offspring. However, it differs considerably from Simple GAs in encoding genetic information in the chromosome, and in its phenotypic interpretation in following ways:

- sGA utilises chromosomes with a largely hierarchical directed graph genetic structure. As an example, sGA's having a two-level hierarchy directed graph structure of genes are shown in figure 1(a), and chromosomal representations of these structures are shown in figure 1(b).
- Genes at any level can be either active or passive .
- High level genes activate or deactivate sets of lower level genes. So the dynamic behavior of genes at any level (i.e whether they will be expressed phenotypically or not) is governed by higher leverage genes.

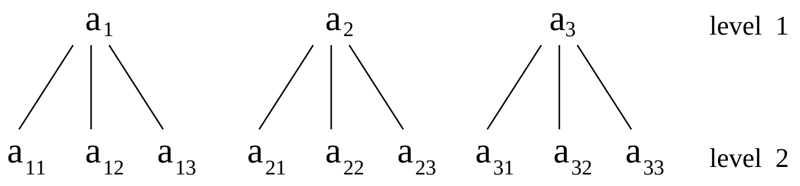
a_1
a_2
a_3
level 1
a_{11} a_{12} a_{13} a_{21} a_{22} a_{23} a_{31} a_{32} a_{33}
level 2

Figure (a). A 2-level structure of sGA.

$(a_1\ a_2\ a_3\quad a_{11}\ a_{12}\ a_{13}\ a_{21}\ a_{22}\ a_{23}\ a_{31}\ a_{32}\ a_{33})$ -a chromosome

Figure (b). A string formation of sGA.

Figure 1: A Representation of the Structured Genetic Algorithm.

Thus a single change at higher level represents multiple changes at lower levels in terms of genes which are active and it produces an effect on the phenotype that could only be achieved in simple GA by a sequence of many random changes. However, genes which are not active (passive genes) do not disappear, they remain in the chromosome structure and are carried in a neutral and apparently redundant form to subsequent generations with the individual's string of genes for future potential use. So genetic operations (crossover and mutation) altering high-level genes result in changes in the active elements of the directed graphs and hence control the development of fitness of the phenotype. Thus the Structured Genetic model allows large variations in the phenotype while maintaining high viability. It introduces a new type of intra-chromosome operation which allows coordinated multiple bit changes in the active genes in a single generation. It is therefore able to function well in complex environments.

Further information on sGA and a simple mathematical model with some empirical results may be found elsewhere (Dasgupta and McGregor, 1992d; Dasgupta and McGregor, 1992c).

3 Network design using sGA

Figure 2 shows the working principle of sGA for designing an application specific neural network architectures (Dasgupta and McGregor, 1992a).

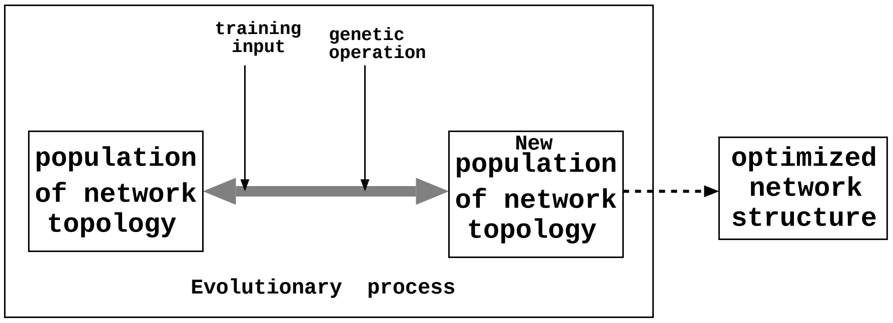
training input
genetic operation
population of network topology
New population of network topology
optimized network structure
Evolutionary process

Figure 2: Designing process of neural network.

For the empirical studies here we have considered a two-level Structured Genetic Algorithm. Each individual is composed of two-segments of strings which represent the two-level of genomic structure; the higher level defines the connectivity, the lower encodes the connection weights and biases. The high level of sGA searches the connectivity space of N units (to evolve a minimal network structure), while the low-level searches for optimized connection weights of the network. The fitness of each individual is determined by the combination of its two components, i.e., each individual is treated as a single complete network. A set of individuals (population) is generated randomly to initialize the evolution-learning process.

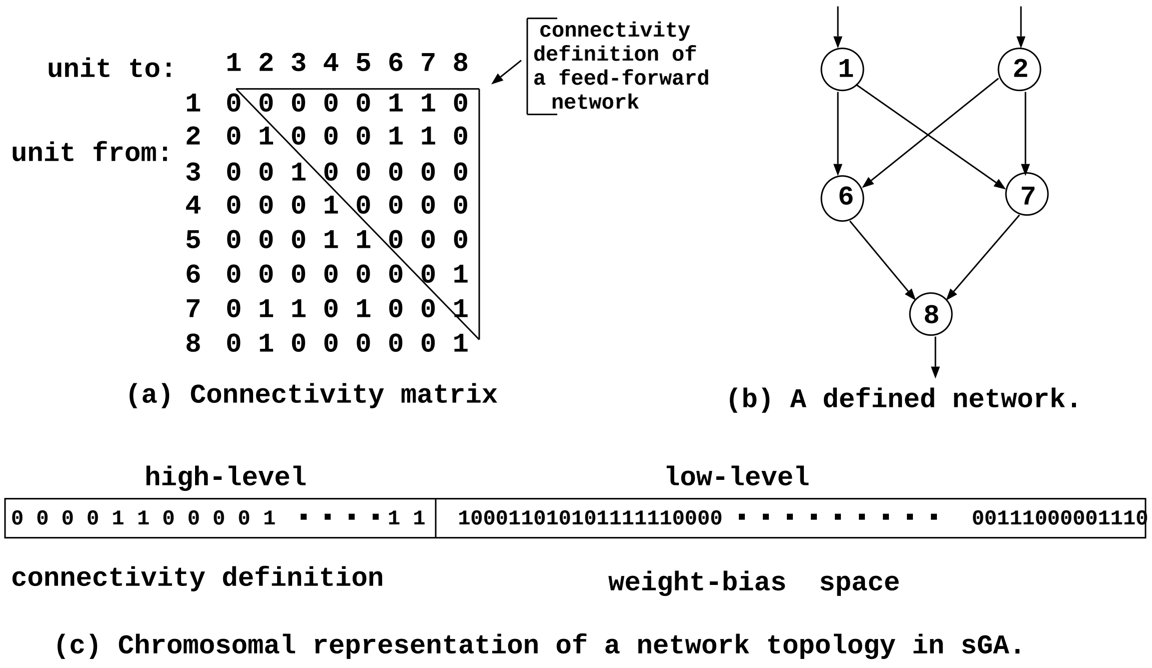

Figure 3: A two-level sGA representating neural network.

For representing high level portion of genotype, we have used a connectivity matrix representation (Kitano, 1990a; Miller et al., 1989) as shown in figure 3, and encoded genes are arranged in row-major fashion. Each bit in the high level of the chromosome represents an individual connection which can take the value '1' or '0', the value '1' meaning that the connection is established. Accordingly column j represents the *fan-in* of connections to unit

j and row i represents *fan-out* of connections from unit i.

A feedforward network is one whose topology has no closed paths. Its input nodes are the ones with no arcs to them and its output nodes have no arcs away from them. Since we are examining here only feed-forward networks, we need only a upper-right triangle of the connectivity matrix (Kitano, 1990a) as shown in figure 3, but the complete matrix would be required to generate recurrent networks.

The definition of neural network performance depends on the application. If the application requires good generalization capabilities, the results of testing on (appropriately chosen) non-training data are important. If a network capable of real-time learning is required, the learning rate must be maximized. Low connectivity is similarly beneficial. In most applications several such criteria must be considered. In the sGA approach, these important aspects of application-specific network design may be included in the evaluation function.

4 Fitness function and reward scheme.

In every generation, each chromosome is decoded into its phenotype (a network structure with its weights), and its fitness is evaluated by taking into account the feasibility of the structure and its ability to learn a set of training signals. More specifically, since sGA is used to find both an optimal architecture and the synapse weights, the evaluation function must include not only a measure of sum-square-error (e.g., MSE), but also a feasibility measure of network structure and its complexity (i.e., number of nodes and their connectivities). The higher the fitness of an individual, the more stable the topology (no high-level changes) and greater the probability of its being selected as parent in the subsequent generations. In a randomly-generated initial population there are likely to be a large number of individuals which show poor performance due to two reasons: first, they have a infeasible network structure i.e., improper connectivity pattern; and second, arbitrary values of weight-bias parameters, may be far from optimum (even though the structure is feasible). A network structure is infeasible if there exists no path from input nodes, and/or to output nodes, if there is fan-in to a hidden node but no fan-out or vice versa, if any unreachable substructure etc. The infeasibility measures quantifies the amount by which an individual structure is having deformation (congenital defects). In our fitness criterion, if an individual decodes to a feasible structure, it is rewarded such that its high-level portion remains stable during subsequent generations, while only the weight-bias space is explored. Also the feasible individuals which have fewer nodes and links get a selection advantage for reproduction relative to the competing feasible individuals with more complex structures. Since we are rewarding only the feasible structures, there is no chance of an individual structure getting reward by pruning all its connections and nodes so as to become infeasible. This means that networks with fewer nodes and links are given more learning opportunities over other feasible structures, similar to that used in (Whitley et al., 1990). The individuals decoding to infeasible structures are penalized according to their deformation and undergo a higher mutation probability in their high-level, the structural portion of their genome when selected for reproduction. They thus also have the chance to reproduce by changing their connectivity pattern (which may result in feasible offspring) and become stable members of the population. Exploration of new feasible structures and evolution of weights of the existing stable networks continues until the optimized network architecture evolves or whole population converges to a feasible network architecture. We have used a ranking selection scheme (Baker, 1985) in which a viable individual receives an expected number of offspring which is based on the rank of performance and not on its magnitude.

5 Experiments

As a preliminary study, we have conducted the following two experiments often used for testing and benchmarking network design techniques. The first problem is the well-known exclusive-or (XOR) function which is not linearly separable and requires the use of hidden units to learn the task. The second problem is a 4 by 4 encoder/decoder problem. For these application-specific network designs, we used the training set that contains all possible input patterns, and also defined a number of input/output nodes (for first problem 2-X-1 and second problem 4-X-4, where X represents number of hidden nodes which is also a determining factor). sGA starts with a random initial population of network configurations and reliably discovers a combination of connections of optimal network structures by exploring the space of possible connectivities (removing or introducing connections through genetic operations) that enhance learning ability. We used the logistic transfer function in all the nodes, except the output nodes, where sigmoid function is used, as it is useful that the error on the output nodes be continuous (Whitley et al., 1990), not just discrete, since it allows the genetic algorithm to better discriminate between the performance of different strings representing different solutions. Our experiments have the following parameters. The population size was 100 and each weight-bias space is encoded with 10 bits in the range of -2 to +2. Two different size (8 and 16) of connectivity matrix were used for the problems. Different GA parameter sets (e.g population size, crossover and mutation probabilities) and selection schemes were also tested. The experiments here used a two_point crossover opera-

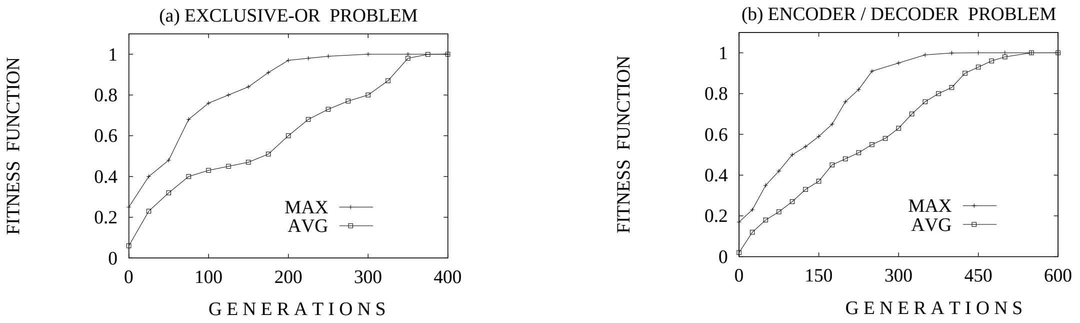

Figure 4: Convergence of population resulting in the evolution of neural networks.

tor with probability of 70%; both varying and adaptive mutations (Kitano, 1990a; Whitley and Hanson, 1989; Whitley et al., 1990) are used in two-levels of sGA respectively. The

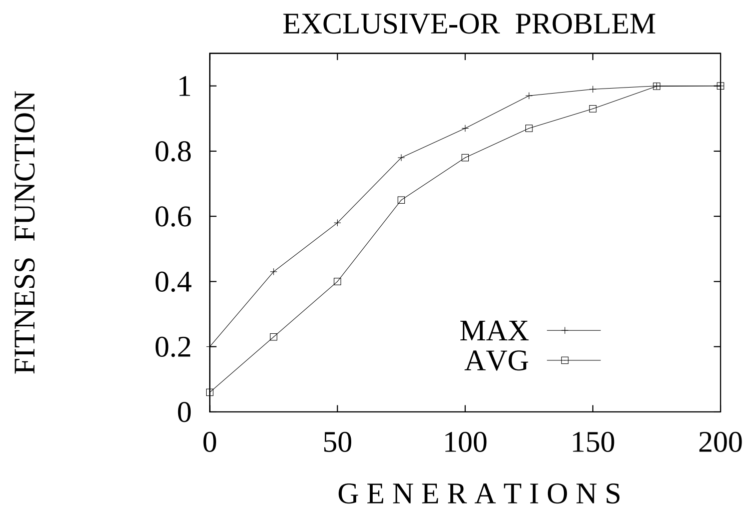

Figure 5: Using Mixed encoding in sGA for solving XOR problem.

results reported here averaged over 10 independent runs. The performance of the sGA in designing optimal neural networks for solving the two problems are shown in figure 4(a) and 4(b) respectively. For both the problems, feasible structures were evolved (fitness of 0.5) within 100 (7000 recombinations) generations and an optimal solution was produced for XOR in 200 generations and for encoder problem in 350 generations. However, some more generations were required to converge the whole population.

Despite solving the problems with binary encoding of both search spaces, it could not scale up properly as the network size increases (Whitley et al., 1990).

We then used a mixed encoding technique, where the high level portion of the chromosome is a binary-coded representing connectivity and the low level is a real-valued space encoding weight-bias (Menczer and Parisi, 1992; Hintz and Spofford, 1990; Montana and Davis, 1989). Each weight-bias space is represented by a single real-value and recombination can only occur between weights. A higher level of mutation is used (Whitley et al., 1990; Whitley and Bogart, 1990; Davis, 1991) and the mutation is such that a random value (range -0.1 to +0.1 here) is added to the existing weight rather than replacing it. This mixed encoding approach exhibits rapid convergence and improved results (shown in figure 5).

The algorithm is implemented on a *sparc workstation* in unix environment. The results demonstrate the effectiveness of using the structured genetic algorithm in evolving optimal neural networks in comparison to the simple genetic algorithms and other optimization techniques.

6 Discussion and further work

We are currently investigating a *three-level sGA*, where the top level defines the number of nodes, the middle level defines connectivity, and low level defines weight-bias space.

In dealing with larger network sizes, encoding a long chromosomal string, a genetic search finds difficulty, since the genetic algorithm exploits schemata representing hyperplanes: increasing the size of the encoding increases the exploration of conflicting schemata and representations in order to find good schemata (Whitley and Hanson, 1989). The success of GA in neural network application depends on an ability to scale up from small networks and low-dimensional 'toy' problems to networks of thousands or millions of nodes of high-dimensional real-world problems (Rogers, 1990).

For solving small neural network problems, direct encoding of connectivity information in the higher level of chromosome proves efficient. But in dealing with the larger network architectures (currently under investigation) it may be useful to use a graph grammatical encoding as mentioned by (Kitano, 1990a) for determining optimal connectivity. Since sGA simultaneously explores structure and weight-space, a dynamic parameter encoding technique(DPE) (Schraudolph and Belew, 1992) may be useful to focus the search on those regions with least variability for fine tuning of connection weights. We have not yet explored the above modifications. However, there may be other possible ways of implementing the Structured Genetic Model, such as incorporating the idea of Cellular Encoding (Gruau, 1992).

The Structured Genetic Approach offers the following advantages:

- It requires neither the assumption of any fixed architecture nor any gradient information technique (like backpropagation) at any level of its network evolving process.
- It is able to evolve reasonably small application specific network architectures.
- The mixed encoding scheme ensures better scalability and speed of convergence.
- For large problems, it may be able to learn quickly (compared to existing methods) when evolve in highly parallel machines e.g. *Super Node of Transputers* (Bessiere, 1991).
- Most modifications applicable to simple genetic algorithms may be equally applicable to sGA (in its different layers) for further enhancing its performance.
- Biological plausibility is one of the most attractive points of this model.

Of course, much work remains to be done to draw any firm conclusion. Our previous experimental results of sGA for non-stationary (Dasgupta and McGregor, 1992c) and multimodal (Dasgupta and McGregor, 1992b) function optimizations exhibited that it is superior when compared with simple genetic algorithms. We feel that the application of sGA to neural network training and design is a much more open area of research.

Acknowledgements

The first author gratefully acknowledges the support given by the Government of Assam (India) for awarding the State Overseas Scholarship. We also acknowledge the help of our colleagues in the Intelligent Knowledge Base System (IKBS) Research Group.

References

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In J.J.Grefenstette., editor, *Proceedings of an International Conference on Genetic Algorithms and their Applications.*, pages 101–111, Carnegie-Mellon University,Pittsburgh.

Bessiere, P. (1991). Genetic algorithms applied to formal neural networks. Imag-lgi/lasco3, Institute of IMAG-Lab. of Cognative Science, France.

Chalmers, D. J. (1990). The evolution of learning: An experiment in genetic connectionism. In *Proceedings of the 1990 Connectionist Models Summer School*. Morgan Kaufmann.

Dasgupta, D. and McGregor, D. R. (1992a). Designing Neural Networks using the Structured Genetic Algorithm. In *Proceedings of the International Conference on Artificial Neural Networks(ICANN)*, Brighton, U K.

Dasgupta, D. and McGregor, D. R. (1992b). Engineering Optimizations using the Structured Genetic Algorithm. In *Proceedings of European Conference on Artificial Intelligence(ECAI)*, Vienna, Austria.

Dasgupta, D. and McGregor, D. R. (1992c). Nonstationary function optimization using the Structured Genetic Algorithm. In *In* Proceedings of Parallel Problem Solving From Nature *(PPSN-2), Brussels, 28-30 September.*

Dasgupta, D. and McGregor, D. R. (1992d). A Structured Genetic Algorithm: The model and the first results. *Presented at AISB PG-Workshop*, January. (Tech Report NO. IKBS-2-91).

Davis, L. (1991). *Handbook of Genetic Algorithms*. Von Nostrand Reinhold, New York., first edition.

de Garis, H. (1990). Genetic programming: Modular neural evolution for darwin machines. In *Intenational Joint conference on Neural Network(IJCNN).*, pages 194–197.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley., first edition.

Gruau, F. C. (1992). Cellular encoding of genetic neural networks. Technical Report 92-21, Institut IMAG, Grenoble, France.

Harp, S. A., Samad, T., and Guha, A. (1989). Towards the genetic synthesis of neural networks. In Schaffer, J. D., editor, *Proceedings of third International conference on Genetic Algorithms*, pages 360–369, George Mason University, USA. Morgan Kaufmann.

Harp, S. A., Samad, T., and Guha, A. (1990). Designing application-specific neural networks using the genetic algorithm. *Advances in Neural Information Processing Systems*, 2:447–454.

Hintz, K. J. and Spofford, J. J. (1990). Evolving a neural network. In et. al., A. M., editor, *Proceedings 5th IEEE International symposium on Intelligent Control*, pages 479–484, Los Alamitos California, U S A. IEEE Computer Society Press. Volume-I.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan press, Ann Arbor.

Kitano, H. (1990a). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476.

Kitano, H. (1990b). Empirical studies on the speed of convergence of neural network training using genetic algorithms. In *Proceedings of AAAI.*

Koza, J. R. and Rice, J. P. (1991). Genetic generation of both the weights and architecture for a neural network. In *Intenational Joint conference on Neural Network(IJCNN).*

Mariči̇c, B. and Nikolov, Z. (1990). GENNET - system for computer aided neural network design using genetic algorithm. *IJCNN*, I:102–105.

Marti, L. (1992). Genetically generated neural networks II: Searching for an optimal representation. CAS/CNS-TR-92 015, Boston University, Center for Adaptive Systems.

Menczer, F. and Parisi, D. (1992). Evidence of hyperplanes in the genetic learning of neural networks. *Biological Cybernetics*, 66(3):283–289.

Miller, G. F., Todd, P. M., and Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In Schaffer, J. D., editor, *Proceedings of third International conference on Genetic Algorithms*, pages 379–384.

Montana, D. J. and Davis, L. (1989). Training feed forward networks using genetic algorithms. In *Proceedings of International Joint Conference on Artificial Intelligence.*

Rogers, D. (1990). Predicting Weather Using a Genetic Memory: a combination of Kanerva's sparse distributed memory with Holland's Genetic Algorithms. *Advances in Neural Information Processing Systems*, 2:455–464.

Schraudolph, N. N. and Belew, R. K. (1992). Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 9(1):9–22.

Whitley, D. and Bogart, C. (1990). The evolution of connectivity: Pruning neural networks using genetic algorithms. In *Proceedings of International Joint Conference on Neural Networks*, pages 134–137.

Whitley, D. and Hanson, T. (1989). Optimizing neural networks using faster, more accurate genetic search. In Schaffer, J. D., editor, *Proceedings of third International conference on Genetic Algorithms*, pages 391–396.

Whitley, D. and Starkweather, T. (1990). Optimizing small neural networks using a disributed genetic algorithm. In *Intenational Joint conference on Neural Network(IJCNN).*, pages 206–209.

Whitley, D., Starkweather, T., and Bogart, C. (1990). Genetic algorithms and neural networks: optimizating connections and connectivity. *Parallel Computing*, 14:347–361.