

# Evolving Neural Networks using Attribute Grammars

**Talib S. Hussain**

Computing and Information Science Dept,  
Queen's University,  
Kingston, ON, Canada. K7L3N6  
hussain@cs.queensu.ca

**Roger A. Browse**

Computing and Information Science Dept,  
Psychology Dept., Queen's University  
Kingston, ON, Canada. K7L 3N6  
browse@cs.queensu.ca

**Abstract-** The evolutionary optimization of neural networks involves two main design issues: how the neural network is represented genetically, and how that representation is manipulated through genetic operations. We have developed a genetic representation that uses an attribute grammar to encode both topological and architectural information about a neural network. We have defined genetic operators that are applied to the parse trees formed by the grammar. These operators provide the ability to introduce selection strategies that vary during the course of evolution.

## 1 Introduction

Grammar-based representations of neural networks are very useful for the evolutionary optimization of neural networks (Yao, 1993; Gruau, 1995; Hussain and Browse, 1998a, 1998b). We have developed the Network Generating Attribute Grammar Encoding (NGAGE) technique which permits attribute grammars to be used successfully in representing and exploring a space of neural networks (Browse, Hussain and Smillie, 1999). This approach offers the capability of representing a wide variety of neural network models, and also permits the design of useful dynamic genetic operators. In this paper, we present reproduction operators that perform biased offspring creation. These operators use knowledge of the grammar representation to adapt their biases in response to the values of fitness measures.

## 2 NGAGE Properties

NGAGE systems use attribute grammars (Knuth, 1968) to specify classes of neural networks. An attribute grammar consists of a context-free grammar base in which the productions are supplemented with the ability to compute values of both synthesized and inherited attributes which are associated with the terminal and non-terminal symbols. The use of such attributes extends the representational power of the context-free grammar. The context-free component of an NGAGE grammar specifies the structural

components and the manner in which they are organized within a neural network as processing nodes.

```
<S> → <in-port> <out-port> <feedb-port> <FULL-NET>
<FULL-NET> → <CONTROL> <NETWORK>
<CONTROL> → <stable-act> <stable-feedb> <learning-start>
<NETWORK> → <INPASS> <PROCESS> <HIDDEN>
<HIDDEN> → <LAYER> <HIDDEN>
           → <LAYER>
<LAYER> → <PROCESS> <LAYER>
           → <PROCESS>
<PROCESS> → <process-nodes>
<INPASS> → <pass-nodes>
```

Figure 1: Context-Free Portion of Attribute Grammar for Back-Propagation Networks

Figure 1 provides the context-free component for the class of back-propagation networks with arbitrary numbers of hidden layers and arbitrary numbers of processing nodes in each layer. In the grammar, lower case symbols represent collections of nodes, and upper case symbols are non-terminals that represent structural aspects of the network that will result. The nodes specified in the production from the root symbol <S> indicate the special layers of nodes that represent the input, output, and the error feedback for the output layer which are all fixed in size. The special nodes indicated in the production from the non-terminal <CONTROL> are the processing nodes that cycle the stages of either feeding forward or learning within the operating network that results. The non-terminal <INPASS> creates the fan-out layer for the input. The remaining productions are responsible for creating the hidden layers

Figure 2 shows an example of a parse-tree that might result from the application of the productions of the grammar. This parse tree will eventually result in the back-propagation network shown in figure 3. However, first the attributes must be computed. The values of the attributes that are computed within the parse tree encode the connections among nodes of the network along with the characteristics of the operation of the nodes. Inherited attributes are used to constrain the structures formed by the symbols of the right hand side of the production rule.

Synthesized attributes are used to collect and store structural information about network components.

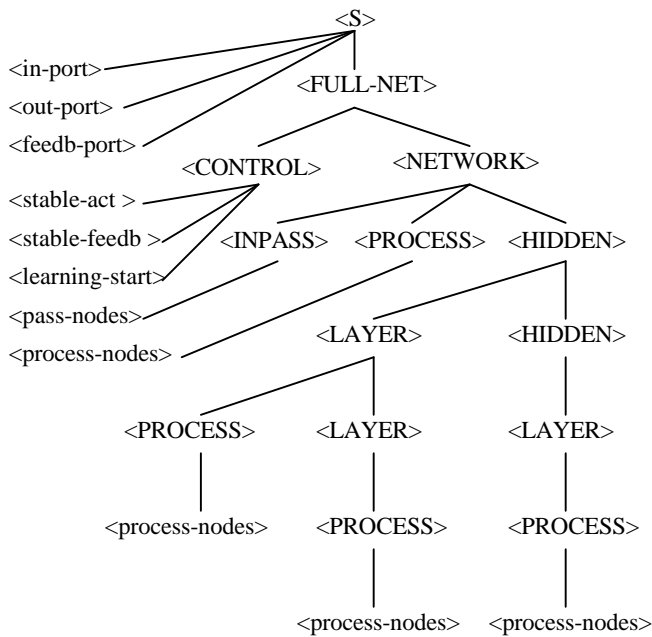


Figure 2: Sample NGAGE Parse Tree

$\langle \text{HIDDEN} \rangle_1 \rightarrow \langle \text{LAYER} \rangle \langle \text{HIDDEN} \rangle_2$ <i>(inherited)</i> $\langle \text{LAYER} \rangle.\text{max\_size} := \langle \text{HIDDEN} \rangle_1.\text{max\_size}$ $\langle \text{HIDDEN} \rangle_2.\text{max\_layers} := \max((\langle \text{HIDDEN} \rangle_1.\text{max\_layers} - 1), 0)$ $\langle \text{HIDDEN} \rangle_2.\text{max\_size} :=$ if $\langle \text{HIDDEN} \rangle_2.\text{max\_layers} > 0$ then $\langle \text{HIDDEN} \rangle_1.\text{max\_size}$ else 0
$\langle \text{HIDDEN} \rangle \rightarrow \langle \text{LAYER} \rangle$ <i>(inherited)</i> $\langle \text{LAYER} \rangle.\text{max\_size} := \langle \text{HIDDEN} \rangle.\text{max\_size}$

Figure 3: Network Depicted by Parse Tree

The values of the attributes that are computed within the parse tree encode the connections among nodes of the network along with the characteristics of the operation of the nodes. Inherited attributes are used to constrain the structures formed by the symbols of the right hand side of the production rule. Synthesized attributes are used to collect and store structural information about network components.

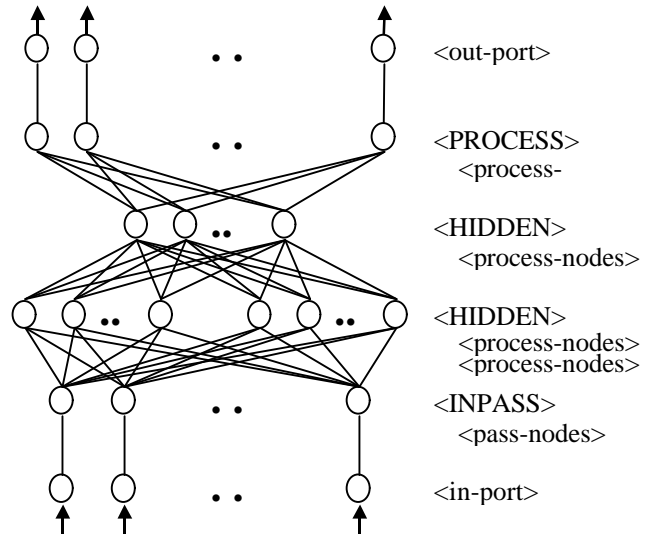


Figure 4: Example of Inherited Attributes

Figure 4 illustrates the use of inherited attributes. The production  $\langle \text{HIDDEN} \rangle \rightarrow \langle \text{LAYER} \rangle \langle \text{HIDDEN} \rangle$  adds a hidden layer of processing nodes. In the application of that production, an inherited attribute *max\_size* specifies the maximum number of processing nodes permitted at any layer in the network. This attribute has become associated with the left-hand side non-terminal  $\langle \text{HIDDEN} \rangle$  through a series of inheritances from the root symbol. This value is inherited by the non-terminal  $\langle \text{LAYER} \rangle$  in the parse tree. The other inherited attribute, *max\_layers*, works the same way, but it specifies the maximum number of hidden layers that are allowed in the network. It is inherited by the right hand side  $\langle \text{HIDDEN} \rangle$  with the value decremented, and constrained to not go below zero. The right hand side  $\langle \text{HIDDEN} \rangle$  also inherits the *max\_size* attribute with the provision that if *max\_layers* has been exceeded, the *max\_size* that is inherited will be held to zero.

Figure 5 shows the synthesized attributes for the same production of the grammar. The information necessary to form connections within the network is accumulated from the lower levels in the parse tree through attributes of the right-hand symbols. That information is then combined into attributes that become associated with the left-hand side non-terminals. Through a continuation of this same process, the information about the connections within the network is eventually recorded in attributes associated with the root symbol of the parse tree.

The synthesized attributes of the root symbol of the tree form a concise neural network specification. The NGAGE system's neural interpreter is able to accept this specification and carry out the functions of the network.



made, there may be no available matching symbol in the second tree.

In defining our genetic operators, we use a third approach that takes into account the points raised in cases (c) and (d). Our genetic operators have access to two sets of probabilities, one that influences the points of selection for crossover and mutation, and one that influences the creation of new individuals and mutations.

#### 4 Probabilistic Generation of Individuals

The assignment to grammar productions of probability values that influence genetic operators and the adaptation of those values over evolution has been considered by Salustowicz and Schmidhuber (1997). In their research, they evolve parse trees generated from an untyped grammar. During evolution, a *prototype parse tree* is maintained. This prototype tree is always kept at least as large as the largest parse tree in the population. Each node in the prototype tree has associated with it a probability value for every possible grammar production. This prototype tree is used to influence the creation of new individual parse trees. At a given position in the new parse tree, the decision of which production to apply next is determined by a random selection that is weighted by the probability values stored at the exact corresponding location in the prototype tree. Expansion of a branch ceases when a terminal is reached, and thus, for that new individual, the probability values of the corresponding descendants in the prototype tree are not used.

The evolution process followed by Salustowicz and Schmidhuber is unusual. At the beginning of each generation, a completely new population is created, with each individual influenced by the prototype tree as above. At the end of fitness evaluation, the best individual in that generation is identified. The prototype tree is then modified slightly so that any new individuals that are subsequently created will be more similar to that best individual than would have been the case previously. The final step for a given generation is to perform slight mutations on the probabilities stored at nodes in the prototype tree.

In NGAGE, we also assign a set of probabilities that is used in the selection of productions during the course of generating parse trees. However, we do not tie those probability values to specific locations in the parse tree, and store a single probability value to each production. In order to have the limited recursive mechanism needed to specify such characteristics as the number of processing nodes per layer, it is necessary to have production rules with common left-hand side non-terminals. The usual approach is to randomly select among alternative productions during generation. While this is random with respect to the use of productions, it does not provide a uniform random

distribution of the network structures. For example, for a grammar with the two productions  $\{P \rightarrow Pa, P \rightarrow a\}$ , each sentence  $a^n$  has a probability  $P(a^n) = 1/2^n$ . It is unlikely that the needs of the genetic algorithm would be best served with this default distribution. Thus we have incorporated probabilities that are associated with each of the choices in the generation process. All rules that share the same left-hand symbol are assigned probability values such that their total is 1. Where  $x$  is the probability of the recursive production of the pair shown above (i.e.,  $P \rightarrow Pa$ , with probability  $x$ , and  $P \rightarrow a$  with probability  $1-x$ ), the probability of generating sentences becomes the more general form  $P(a^n) = x^{n-1}(1-x)$ .

In NGAGE, a second set of probabilities is associated with the non-terminal symbols of the grammar. In the creation of an offspring by reproduction operators, these probabilities will have an effect on which nodes in the tree(s) are selected for mutation and/or crossover.

We define two new reproduction operators that exploit these probability values. A *biased typed subtree crossover* operator is defined as follows. Given two parent parse trees, all the non-terminal symbols that are shared by both trees are extracted. Those symbols that have a non-zero probability of selection are considered in a random, weighted selection. The result is the selection of a non-terminal symbol from the grammar that exists in both trees and is a valid crossover point. Then, for each tree, all the nodes that match the selected symbol are identified, and a uniform random selection of one of those nodes is made. The subtrees rooted by the chosen node in each tree are swapped to produce the offspring. Since crossover can only occur at subtrees that have the same root symbol, this crossover operator guarantees that the two newly created examples could have been generated from the defining grammar. The operator is somewhat expensive computationally, but is guaranteed to find a match if one exists.

A *biased typed subtree mutation* operator is defined as follows. Given a single parent tree, a node is selected using the same procedure described above of probabilistically selecting a non-terminal symbol first and randomly choosing a matching node next. The subtree rooted at the selected node is then replaced by a new tree randomly generated using the grammar, but with the selected non-terminal as the start symbol. This guarantees that the mutated network falls within the class of networks that is described by the grammar. In this generation process, the probabilities associated with the productions may influence the mutation.

#### 5 Dynamic Biases

Another issue to be addressed in the design and application of our genetic operators concerns the probability values that

are used. On an arbitrary problem and with an arbitrary neural network, it is difficult to provide a strong rationale for any particular set of probability values. For instance, consider a grammar in which one non-terminal symbol reflects a large-scale change to the network structure and a second reflects a small change. Setting different probability values to the two symbols allows tuning of the genetic algorithm to favor either large or small-scale modifications. However, the values that are chosen may have highly detrimental consequences for the genetic search. Ensuring a good genetic search may require an additional search for a good set of probability values. As well, there may be no one fixed set of values that is appropriate. For instance, the scale of the ideal modifications may vary through the course of the genetic search, with large-scale changes being more important at one time in the evolution and small-scale changes more important at a different time. Both of these points suggest that a mechanism should be used whereby the evolutionary algorithm can adapt the probability values.

During the course of an evolutionary algorithm, operations involving one particular non-terminal symbol may be more effective than operations on other symbols in generating offspring with improved fitness. It is possible to keep track of the results of applying genetic operators to each of the non-terminal symbols, and use a reinforcement learning scheme to influence the probability of applying future operations to that symbol. For example, if a particular symbol is selected in applying crossover and the two offspring have higher fitness values than the original parents, the probability of selecting that symbol can be increased slightly using a reinforcement learning algorithm.

Similarly, it is possible to keep track of the production rules used in each mutation and adapt the probabilities associated with those productions depending upon the comparative fitness of the parent and offspring. We therefore extend our evolutionary algorithm to include reinforcement learning on both symbol probabilities and production probabilities. The resulting evolutionary process is summarized below.

Initially, a random set of probability values are assigned to the grammar productions and another set to the non-terminal grammar symbols. An initial population of parse trees is created, as influenced by the production probabilities. The fitness of each individual is then evaluated. A random decision of whether to perform crossover or mutation is made, and a fitness-proportional selection of one or two individuals (as appropriate) is made.

If crossover is chosen, then all shared non-terminal symbols in the two selected individuals are extracted and a random choice is made, as weighted by the symbol probabilities. Let  $s$  be the symbol that was chosen. Two appropriate offspring are generated. The fitness of each

offspring is then evaluated and a family competition takes place, in which the worst two individuals are removed from the population. If an offspring survives to the next generation, then we consider that a beneficial genetic manipulation has taken place. The symbol probability associated with  $s$  is thus increased by a factor proportional to the fitness difference between the best offspring and the best discarded parent. If neither offspring survives, then we consider that a detrimental genetic manipulation has taken place and we decrease the probability of  $s$  by a factor proportional to that fitness difference between the best offspring and the worst parent.

If mutation is chosen, then all non-terminal symbols in the selected individual are extracted and a random choice is made, as weighted by the symbol probabilities. Let  $s$  be the symbol that was chosen. An appropriate offspring is generated by choosing a matching node  $n$  and creating a new random subtree. Let  $r$  be the production rule that was chosen to expand  $n$ . A family competition takes place, and if the offspring survives, the symbol probability of  $s$  and the production probability of  $r$  are increased by factors proportional to the fitness difference between the offspring and its discarded parent. The values are similarly decreased if the offspring does not survive.

The resulting system differs significantly from the approach of Salustowicz and Schmidhuber. The evolutionary process of NGAGE follows a more traditional approach of applying crossover and mutation operators to individuals in creating a new generation. The symbol and production probabilities influence the new individuals that are created, and the mechanism whereby they are changed over time is dependent upon comparisons to immediate parents and not to the best individual in each generation.

## 6 Conclusions

The NGAGE system has been extended to include probabilistic selection of grammar symbols and production rules in the application of biased typed tree-based genetic operators, as well as a reinforcement learning mechanism that adapts the values of those probabilities dynamically through the course of evolution. Two potential problems that will be addressed in future work are the possible introduction of hill-climbing and low genetic diversity into the genetic search. Both of these may result if the reinforcement mechanism tends to produce probability values that strongly favor a small number of non-terminal symbols. In this situation, the genetic adaptations required to lead the genetic search away from a local optimum will have an extremely low likelihood of occurrence, and the operators will tend to propagate highly similar individuals.

## Acknowledgments

The research reported in this paper was conducted with financial support from the Natural Science and Engineering Research Council of Canada.

## Bibliography

- Gruau, F. (1995) "Automatic definition of modular neural networks," *Adaptive Behavior*, 3, p. 151-183.
- Browse, R.A., Hussain, T.S., and Smillie, M.B. (1999) "Using attribute grammars for the genetic selection of backpropagation networks for character recognition," *Proceedings of SPIE: Applications of Artificial Neural Networks in Image Processing IV*, p. 26-34.
- Haynes, T.D., Schoenefeld, D.A. and Wainwright, R.L. (1996) "Type inheritance in strongly typed genetic programming," Chapter 18 in K.E. Kinneer, Jr. and P.J. Angeline (Eds.), *Advances in Genetic Programming 2*. Cambridge, Mass: MIT Press.
- Haynes, T.D. (1998) "Strongly typed genetic programming," Tutorial presented at *1998 Genetic Programming Conference*.
- Hussain, T.S. and Browse, R.A. (1998a) "Network generating attribute grammar encoding," *1998 IEEE International Joint Conference on Neural Networks*, p. 431-436.
- Hussain, T.S. and Browse, R.A. (1998b) "Including control architecture in attribute grammar specifications of feedforward neural networks," *1998 Joint Conference on Information Sciences 2*, p. 432-436.
- Knuth, D. E. (1968) "The semantics of context-free languages," *Mathematical Systems Theory*, 2, p.127-145.
- Montana, D.J. (1993) "Strongly typed genetic programming," *BBN Tech Report #7866*.
- Salustowicz, R. and Schmidhuber, J. (1997) "Probabilistic incremental program evolution," *Evolutionary Computation*, 5, p. 123-141.
- Yao, X. (1993) "Evolutionary artificial neural networks," *International Journal of Neural Systems*, 4, p. 203-222.