Genetic Operators with Dynamic Biases that Operate on Attribute Grammar Representations of Neural Networks

Talib S. Hussain
Computing and Information Science Dept.
Queen's University
Kingston, ON, Canada. K7L 3N6
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

1 INTRODUCTION

Grammar-based representations of neural networks have shown promise in advancing the study of the evolutionary optimization of neural networks (Yao, 1993; Gruau, 1995; Hussain and Browse, 1998). Our research on the Network Generating Attribute Grammar Encoding (NGAGE) technique has demonstrated that attribute grammars may be used successfully in representing and exploring a space of neural networks (Browse, Hussain and Smillie, 1999). In addition to offering the capability of representing a wide variety of neural network models, NGAGE also offers the potential of designing meaningful dynamic genetic operators. In this paper, we present two reproduction operators that perform a biased offspring creation, and use knowledge of the grammar representation to adapt those biases in response to fitness measurements.

2 NGAGE PROPERTIES

An NGAGE system uses an attribute grammar (Knuth, 1968) to specify a class of neural networks. Attribute grammars consist 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 may be 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 distinct structural components and the manner in which they are organized within a neural network (see Figure 1). 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 may be used to provide constraints on the structures formed by the symbols of the right hand side of the production rule (see Figure 2). Synthesized attributes may be used to collect and store structural information about network components (see Figure 3).

<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

<HIDDEN> → <LAYER> <HIDDEN>

(inherited)

<LAYER>.max_size := $<HIDDEN>_1$.max_size

$<HIDDEN>_2$.max_layers :=
max(($<HIDDEN>_1$.max_layers - 1), 0)

$<HIDDEN>_2$.max_size :=
if $<HIDDEN>_2$.max_layers > 0
then $<HIDDEN>_1$.max_size
else 0

<HIDDEN> → <LAYER>

(inherited)

<LAYER>.max_size := <HIDDEN>.max_size

Figure 2: Example of Inherited Attributes

<HIDDEN> → <LAYER> <HIDDEN>

(synthesized)

<HIDDEN>$_1$.in_nodes := <LAYER>.all_nodes

<HIDDEN>$_1$.out_nodes :=

if non-empty(<HIDDEN>$_2$.out_nodes)

then <HIDDEN>$_2$.out_nodes

else <LAYER>.all_nodes

<HIDDEN>$_1$.all_nodes :=

<LAYER>.all_nodes ∪ <HIDDEN>$_2$.all_nodes) ∪
sat_nodes(<LAYER>.all_nodes,<HIDDEN>$_2$.in_nodes)

<HIDDEN>$_1$.connections :=

<HIDDEN>$_2$.connections ∪
sat_conns(<LAYER>.all_nodes,<HIDDEN>$_2$.in_nodes)

Figure 3: Example of Synthesized Attributes

Each parse tree generated from the grammar depicts an individual neural network (see Figures 4 and 5). 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. The interpreter may be called by a problem-dependent training paradigm in order to train and test the network on a particular set of data.

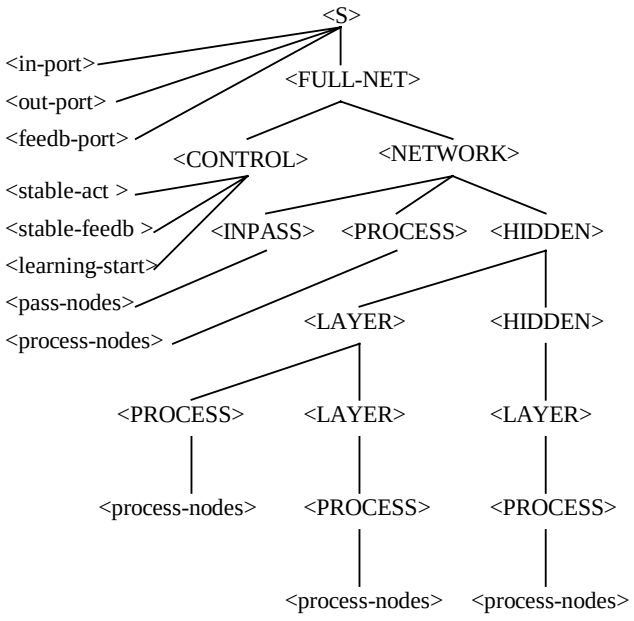

Figure 4: Sample NGAGE Parse Tree

Finally, the NGAGE system includes an evolutionary algorithm that performs a genetic search over the space of possible networks formed by the grammar. The context-free parse tree is used as the genotype, and the performance of the trained network is used as a fitness measure.

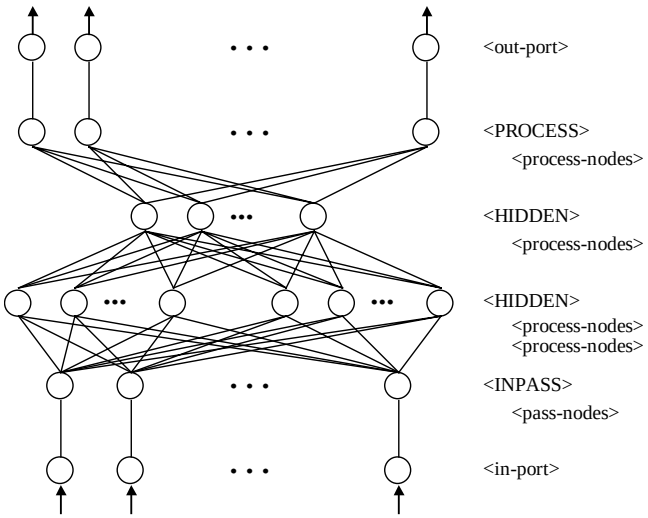

Figure 5: Network Depicted by Parse Tree

A more complete description of the NGAGE grammar for back-propagation networks illustrated in Figures 1 through 5 is provided in Browse et al (1999).

3 GENETIC OPERATOR DESIGN

There are several important aspects of the design principles behind the attribute grammar representation of neural networks that affect the development of appropriate genetic operators.

(a) *Any of the neural networks generated by the grammar are encoded as simple context-free parse trees.*

This suggests the use of tree-based genetic operators, such as the sub-tree crossover and sub-tree mutation operators of genetic programming.

(b) *An NGAGE grammar is hierarchical.*

Unlike most genetic programming representations, an NGAGE grammar contains non-terminal symbols that are not interchangeable. Proper evaluation of the attributes requires a valid context-free parse tree. The genetic operators used must therefore preserve type in order to ensure viable offspring (Haynes, Schoenefeld and Wainwright, 1996).

(c) *Different non-terminal symbols in an NGAGE grammar may represent very different functional components.*

Previous grammar representations of neural networks (Gruau, 1995) have focused on using the grammar to specify the topology of a set of identical nodes. In NGAGE, a variety of nodes are possible and different non-terminal symbols may refer to components that perform functionally distinct operations. Further, two different non-terminal symbols may differ significantly in their functional importance to the resultant network. This suggests that genetic operators which are biased towards selecting certain symbols as points of mutation or crossover may focus

the genetic search on exploring interesting areas of the search space.

(d) *Multiple productions may share the same left-hand non-terminal symbol and may expand that symbol in very different ways.*

This is a natural result of exploiting the expressive power of a grammar. A grammar is defined to represent the space of all the parse trees that may be formed from all possible expansions of the root symbol. However, in searching that space algorithmically, the likelihood of randomly visiting a particular parse tree will depend upon the relative frequency with which the productions are applied. If all productions are equally likely, certain parse trees may have an extremely low likelihood of being generated randomly. In the context of a genetic search, this has strong implications for the creation of the initial population and the application of subtree mutation operators. Different biases in the relative likelihood of applying productions will focus the genetic search on different areas of the search space.

4 GENETIC OPERATOR DEFINITION

The genetic operators used in the NGAGE system include typed subtree mutation and typed subtree crossover, as suggested in cases (a) and (b) from the previous section. Research on strongly typed genetic programming (Haynes et al., 1996) provides a basis for the creation of typed operators. However, there is no accepted mechanism whereby crossover points in the trees are selected. For instance, Haynes (1998) uses a crossover operator that continues to randomly select two nodes, one from each parent tree, until the symbols at those nodes match. The subtrees rooted at those symbols are then swapped. This has the drawback of potentially never finding a match. Montana (1993) uses a crossover operator in which a node in one tree is selected randomly, the second tree is analyzed to extract every node with a matching symbol and then one of those nodes is randomly selected. This has the potential of not finding a match if the first symbol was poorly selected.

In defining our genetic operators, we use a third approach that takes into account the points raised in cases (c) and (d) from the previous section. Consider the addition of two components to our NGAGE system. The first is a set of probabilities associated with each production in the grammar. In the normal creation of a parse tree, these values will affect the frequency with which a rule is applied relative to all other rules that share the same left-hand symbol. The initial population created in the evolutionary algorithm will be thus be biased. In the creation of an offspring by reproduction operators, these probabilities will affect operators such as typed sub-tree mutation.

The second new component is a set of probabilities associated with each non-terminal symbol in 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.

Given these two components, 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

A final 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 the researcher to tune the application 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 point in the evolution and small-scale changes more important at a different point. 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. Preliminary results, some of which have been reported in Browse et al. (1999), suggest that genetic operators with dynamic biases have a beneficial effect on the genetic search of NGAGE grammars.

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. Future work will involve further testing of the new operators and their effects on genetic search. Two potential problems that will be addressed are the 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.

References

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,” *Applications of Artificial Neural Networks in Image Processing IV*.

Haynes, T.D., Schoenefeld, D.A. and Wainwright, R.L. (1996). “Type inheritance in strongly typed genetic programming,” Chapter 18 in K.E. Kinnear, 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. (1998). “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.*

Yao, X. (1993) “Evolutionary artificial neural networks,” *International Journal of Neural Systems*, *4*, p. 203-222.