Attribute Grammars for Genetic Representations of Neural Networks and Syntactic Constraints of Genetic Programming

Talib S. Hussain and Roger A. Browse
Queen's University, Kingston, Ontario

Presented at AIVIGI'98: Workshop on Evolutionary Computation (June 17, Vancouver, BC).

An attribute grammar (Knuth, 1968; Bochmann, 1976) is a context-free grammar augmented by the assignment of semantic attributes to the symbols of the grammar. A production rule specifies not only the replacement of symbols, but also the evaluation of the symbol's attributes. In our research, an attribute grammar is used to specify classes of neural network structures with explicit representation of their functional organization. These representations provide useful constraints upon a genetic optimization that guarantee the preservation of syntactically correct genetic trees with semantically meaningful sub-trees. In this paper, we give a broad overview of our research into attribute grammar representations, from the basic and known capabilities, to the current ideas being addressed, to the future directions of our research.

Attribute Grammars and Neural Networks

An attribute grammar may be applied as a technique for representing and generating classes of neural networks (Hussain and Browse, 1998a,b). Our approach involves two principles in the design of the context free syntax and attribute manipulations. First, we interpret every non-terminal and terminal symbol[1] in the grammar as representing neural network modules. Within the attributes of a symbol, a neural network module is defined to be a set of one or more neurons. From that set, input, output, and hidden neurons are designated along with the connections among them. Second, each production rule is designed to respect the internal integrity of the modules that are represented by the symbols on the right hand side of the production in composing the module that is represented by the left hand symbol. A production rule may create new connections only among the designated input and output nodes of the right hand modules and may not specify the input and output nodes of the left hand module from the hidden nodes of the right hand modules.

The process of generating a network with a given attribute grammar involves three steps. First, a parse tree is generated from the starting symbol through the application of the context-free production rules. This is termed a derivation tree. Second, the attributes of the symbols in the derivation tree are evaluated using the attribute manipulation rules associated with the grammar productions to generate a fully attributed parse tree. Third, a specification of a neural network structure is extracted from the attributes of the root symbol, and interpreted to produce a functional neural network. That neural network may then be randomly initialized and trained accordingly on the available data.

We have termed our approach the Network Generating Attribute Grammar Encoding (NGAGE). Figure 1 presents an NGAGE grammar which generates layered networks with p inputs and q outputs. The symbols x, o and n refer to neuron objects of different types which are created with a unique *id*. 'Neurons', Inputs', 'Outputs', 'Hidden' and 'Connections' are the attributes of the production rules. Each production rule specifies a syntactic rule and a set of attribute manipulations. For the Map, End1 and End2 rules, all the attributes of the left symbol are set equal to the corresponding attributes of the right symbol.

Start: $S \rightarrow B$ Neurons **of** S = [Neurons **of** B $\cup$ $x_1,...,x_p \cup o_1,...,o_q$]; Inputs **of** S = [*id*(x_1),...,*id*(x_p)]; Outputs **of** S = [*id*(o_1),...,*id*(o_q)]; Hidden **of** S = [Inputs **of** B $\cup$ Outputs **of** B $\cup$ Hidden **of** B]; Connections **of** S = [Connections **of** B $\cup$ *full_connect*(Inputs **of** S, Inputs **of** B) $\cup$ *full_connect*(Outputs **of** B, Outputs **of** S)];
Seq: $B_0 \rightarrow B_1\ B_2$ Neurons **of** B_0 = [Neurons **of** B_1 $\cup$ Neurons **of** B_2]; Inputs **of** B_0 = Inputs **of** B_1; Outputs **of** B_0 = Outputs **of** B_2; Hidden **of** B_0 = [Hidden **of** B_1 $\cup$ Outputs **of** B_1 $\cup$ Inputs **of** B_2 $\cup$ Hidden **of** B_2]; Connections **of** B_0 = [Connections **of** B_1 $\cup$ Connections **of** B_2 $\cup$ *full_connect*(Outputs **of** B_1, Inputs **of** B_2)];
Par: $C_0 \rightarrow C_1\ C_2$ Neurons **of** C_0 = [Neurons **of** C_1 $\cup$ Neurons **of** C_2]; Inputs **of** C_0 = [Inputs **of** C_1 $\cup$ Inputs **of** C_2]; Outputs **of** C_0 = [Outputs **of** C_1 $\cup$ Outputs **of** C_2]; Hidden **of** C_0 = [Hidden **of** C_1 $\cup$ Hidden **of** C_2]; Connections **of** C_0 = [Connections **of** C_1 $\cup$ Connections **of** C_2];
Map: $B \rightarrow C$
End1: $B \rightarrow$ node
End2: $C \rightarrow$ node
node: Neurons **of** node = [n]; Inputs **of** node := [*id*(n)]; Outputs **of** node := [*id*(n)]; Hidden **of** node := []; Connections **of** node := []

Figure 1: NGAGE Grammar for Layered Networks

We have derived representations for several networks, including recurrent networks, the Kohonen network, arbitrarily layered networks, and the classic back-propagation network. Our research is continuing to extend the NGAGE representations along several fronts. An interesting consequence of this research is that as more neural network architectures are modeled as NGAGE grammars, the similarities and differences between them are emphasized within a common framework. In addition, the grammatical description of network architectures suggests

[1] We will use the terms non-terminals and terminals to refer to the symbols of the grammars discussed in this paper, and the terms internal nodes and leaves to refer to the placement of elements in the genetic tree. This should clarify differences with genetic programming terminology.

techniques for integrating multiple architectures to form hybrid classes.

Attribute Grammars and Genetic Programming

An attribute grammar may be used in the optimization of neural networks using genetic programming in two ways. Firstly, an attribute grammar may be used to form a genetic encoding of a class of neural networks. Secondly, an attribute grammar provides several properties that may be exploited in the design and application of the operators of the genetic programming technique.

An NGAGE grammar provides three different representations of a specific neural network structure: the derivation tree, the attributed parse tree, and the final functioning neural network. Thus, we have available not only a specification of a neural network's structure, but also information on the functional organization of that structure. As a consequence of the two grammar design principles described in the previous section, each subtree in any generated derivation tree corresponds to a meaningful structural component of the network (i.e., a module). The derivation tree therefore provides a compact, indirect encoding of a neural network with which intelligent genetic manipulations may be made.

In the design of genetic programming, genetic operators must be used which ensure that the offspring they produce are always valid. In the encoding provided by the derivation tree, this requires that the genetic operators form offspring whose derivation trees could be generated from scratch through the appropriate application of the attribute grammar productions. Our solution is to constrain the genetic operators to operate only upon entire subtrees and to ensure that a subtree rooted by a given non-terminal symbol will only be replaced with a subtree rooted by the same non-terminal symbol. This solution is related to work on structure-preserving crossover (Koza, 1994) and strongly-typed genetic programming (Montana 1993; Haynes et al., 1996). Figure 2 demonstrates an example of constrained crossover on two derivation trees (a) and (b) formed from the grammar in Figure 1. The boxed subtree of (a) is replaced with the boxed subtree of (b) to form the new derivation tree (c), which is syntactically valid according to the grammar.

In genetic programming, it is also important that the genetic manipulations be semantically meaningful. If changes to a genetic tree always have entirely unpredictable results on the functionality of the solution, then the genetic search is no better than a random walk. In NGAGE grammars, the property that subtrees in the derivation tree correspond to structural modules ensures that genetic operations will correspond to semantically meaningful changes. A substitution of one subtree with another rooted by the same symbol is effectively the substitution of one modular structure with another, and the only adverse effect on the remaining neural network structure concerns the connections that will be made to the input and output nodes of the new module.

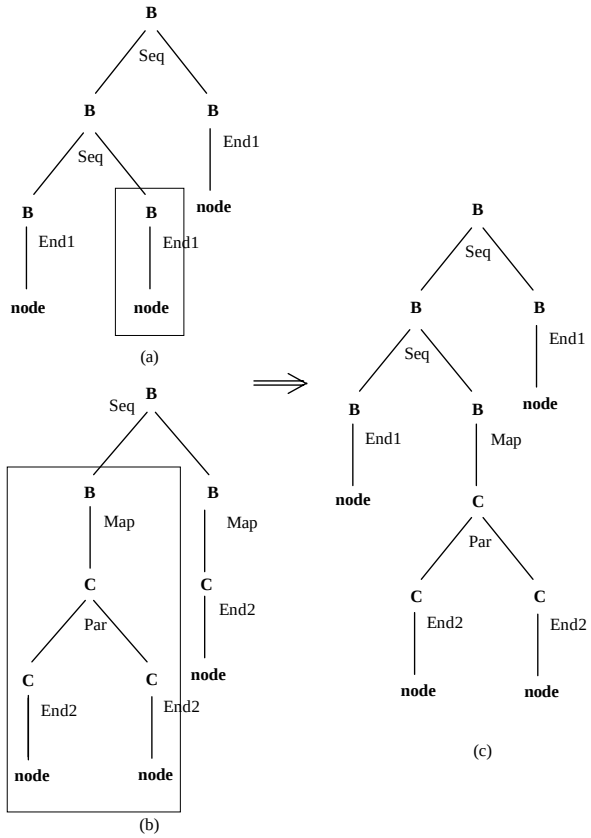

Figure 2: Example of Constrained Crossover

The genetic operators may also be designed to have access to the context-free rules of the grammar. For instance, the mutation operator can identify the non-terminal symbol present at a selected point in the tree and then generate a new subtree by applying the grammar rules using that non-terminal symbol as the initial symbol.

The use of attribute grammars has two additional benefits in the search for good neural network solutions. Firstly, changes to the attribute grammar itself can produce significant and meaningful changes to the space of architectures that are to be explored and require no changes to the genetic programming design. Secondly, the random application of the production rules in creating a population results in a randomization of both the coarse and fine structure of neural network solutions. Since an attribute grammar provides the ability to specify classes of neural networks at the basic structural level, some random choices of productions will reflect large architectural characteristics while others will reflect smaller structural details.

Our approach to the evolutionary optimization of neural networks integrates a variety of principles into a single framework, including the genetic representation of modular neural networks (Happel & Murre, 1994); the use of grammars as a base for genetic encoding of neural networks (Kitano, 1990; Boers et al., 1993; Jacob & Rehder, 1993; Gruau, 1995); and syntactic constraints upon genetic programming (Montana, 1993; Gruau, 1996; Haynes et al., 1996).

Current Work

As a neural network representation, the attribute grammar can be made slightly more interesting through the assignment of probabilities to the various production rules. During the random generation of an individual, the rules applicable to a given non-terminal under expansion may be selected probabilistically. This has the effect of biasing the generation process to encouraging certain structural forms in the final neural network. Note that the class of networks represented by the grammar is not affected, only the frequency with which certain networks are generated. In population-based genetic search, this has the effect of biasing the relative frequency of certain genes in the population. As well, these same probabilities may be used in combination with mutation to naturally constrain certain types of mutations from occurring too often. In this way, the frequency of extreme structures in the population will remain low over the course of evolution unless and until certain extreme cases exhibit high fitness.

The application of a genetic operator such as mutation or crossover requires the selection of an internal node within the derivation trees of the neural networks that make up the current population. If such internal node is represented by a non-terminal that is close to the starting symbol of the grammar, there will be a large structural modification. The selection of a node that corresponds to a non-terminal that is close to the terminals of the grammar will result in a small structural change. Within our current research, each genetic operator selects these non-terminals according to probabilities, thereby providing the ability to bias the scale of structural changes. We are investigating a dynamic set of probabilities that shifts the scale of structural changes as the genetic algorithm proceeds, and we are investigating the use of a reinforcement learning strategy to adjust these probabilities, where the reinforcement signal is derived from the fitness measure for the newly generated network. Our conjecture is that the effect of such dynamic manipulation of the probabilities will have the consequence of the genetic program seeking coarse structure in the early generations and focusing upon the finer structure in the later generations. Such a result could be compared to similar findings of Poli and Langdon (1997) using standard genetic programming.

Future Directions

We expect that the attribute grammar system will lead us to a consideration of cultural evolution (Donald, 1991; Bankes, 1995; Spector & Luke, 1996). In any evolutionary computation of complex learning components such as neural networks, the largest proportion of compute time is dedicated to the evaluation of the fitness of individuals. To evaluate the fitness of a neural network, it must first be instantiated from the genetic code and then it must undergo learning in its environment. At the end of these two processes, the neural network contains a great deal of useful acquired information. However, typically none of that information is transmitted to new individuals, whose training is started from scratch. The conjecture of a cultural evolution process is two-fold. On the one hand, if some information that has been learned by prior individuals can be transmitted relatively quickly to new individuals, then the evaluation of their fitness will be sped up. On the other hand, perhaps the individuals will learn the problem better because they have had a better starting point.

The transmission of cultural information may involve the direct transfer of neural network structure or the indirect transfer of, usually, behavioral information. With neural networks, direct transmission is difficult to perform since it is generally difficult to identify components in two individuals which share identical structure (i.e., sub-graph isomorphism), and even more difficult to support the claim that such components are performing similar functions. In NGAGE, we suspect that the intermediate representation, the attributed parse tree, may be used to quickly identify matching structural components between individuals (e.g., between parent and offspring) and permit direct transfer of structural information (e.g., weights). Two identical components formed from subtrees in similar locations in the genetic tree may also be likely to be performing similar functions in the final network.

References

Bankes, S. (1995) "Evolving social structure in communities of agents through meme evolution," *Proceedings of the Fourth Annual Conference on Evolutionary Programming*. Cambridge, Mass: The MIT Press, p. 333-352.

Bochmann, G.V. (1976) "Semantic evaluation from left to right," *Communications of the ACM*, *19*, p. 55-62.

Boers, E.J.W., Kuiper, H., Happel, B.L.M & Sprinkhuizen-Kuyper, I.G. (1993) "Designing modular artificial neural networks," Leiden University Tech Report #93-24.

Donald, M. (1991) *Origins of the Modern Mind*. Cambridge, Mass: Harvard University Press.

Gruau, F. (1995) "Automatic definition of modular neural networks," *Adaptive Behavior*, *3*, p. 151-183.

Gruau, F. (1996) "On using syntactic constraints with genetic programming," Chapter 19 in K.E. Kinnear, Jr. and P.J. Angeline (Eds.), *Advances in Genetic Programming 2*. Cambridge, Mass: MIT Press.

Happel, B.L.M. & Murre, J.M.J (1994) "Design and evolution of modular neural network architectures," *Neural Networks*, *7*, p. 985-1004.

Haynes, T.D., Schoenefeld, D.A. & 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.

Hussain, T.S. & Browse, R.A. (1998a) "Network generating attribute grammar encoding," *1998 IEEE International Joint Conference on Neural Networks*, May 4-9, 1998 in Anchorage, Alaska, p. 431-436.

Hussain, T.S. & Browse, R.A.(1998b) "Basic properties of attribute grammar encoding," *Third Annual Genetic Programming Conference: Ph.D. Student Workshop*, July 21, 1998 in Madison, Wisconsin.

Jacob, C. & Rehder, J. (1993) "Evolution of neural net architectures by a hierarchical grammar-based genetic system," *ANNGA '93*, p. 72-79.

Kitano, H. (1990) "Designing a neural network using genetic algorithm with graph generation system," *Complex Systems*, *4*, p.461-476.

Knuth, D. E. (1968) "The semantics of context-free languages," *Mathematical Systems Theory*, *2*, p.127-145.

Koza, J.R. (1994) *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, Mass: MIT Press.

Montana, D.J. (1993) "Strongly Typed Genetic Programming," *BBN Tech Report #7866*.

Poli, R. & Langdon, W.B. (1997) "An experimental analysis of schema creation, propagation and disruption in genetic programming," *Proceedings of the Seventh International Conference on Genetic Algorithms*, p.18-25.

Spector, L. & Luke, S. (1996) "Cultural Transmission of information in genetic programming," *Proceedings of the First Annual Genetic Programming Conference*.