4
5

Design of Artificial Neural Networks Using Genetic Algorithms: review and prospect

İbrahim Kuşçu and Chris Thornton

Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QN
Email: ibrahim@cogs.susx.ac.uk christ@cogs.susx.ac.uk

April 0, 1994

Abstract

The design of Artificial Neural Networks by Genetic Algorithm is useful in terms of

The APS contains some fields to describe the address and identification of the area, and the size (i.e., number of units) of it. In addition, 'dimension share' parameters determine the spatial organisation of the units. Since the representation used in Genesis does not assume a simple fully connected network structure, PSFs may be used to determine where a specific unit can make a connection. In PSFs the identity of the target area is coded either as an absolute (i.e., target ID itself) or relative address (i.e., position of the target area relative to current area) mode. Again, the dimension parameters allow connections only in that localised area. Finally, the degree of connectivity (between 30 to 100 percent) and learning rate parameter of back-propagation are also coded in PSFs.

A two point crossover is modified to allow identification of the points by referring to the markers in the blueprint. The variable length representation and the modified crossover seems to allow a much broader space of network architectures to be searched. This can result in more complex architectures.

This design strategy is used to solve two different problems: digit recognition and XOR problem. In both cases Genesis has produced reasonable networks and showed improvements over its initial random structures. But the over all results suggests that the representation used by Genesis is inadequate. More attention to representation of connectivity is needed. For example, a typical chromosome contains concatenation parameters describing the number of layers, size of the layers and how these layers are interconnected. Due to this abstraction, larger nets can be encoded with small chromosomes. However this is only true for some particular group of networks. This method would fail to encode some modular architectures with well defined and repeated groups of neurons.

2.2 The Innarvator System: strong representation

In the Innarvator System [31] a layered feed-forward network of N units is represented by a *connectivity constraint matrix* with dimensions $Nx(N+1)$. Each of the values of the matrix specified by (Column, Row) indices specifies the nature of the constraint of connection from one unit to another. The constraints can be either none (indicated by a zero), learnable (L), learnable but limited to positive values (L+) or learnable but limited to negative values (L-). The rows of the matrix are successively concatenated to form a bit string representation of a network. The following figure shows an example of a constraint matrix representing a 5 unit network (2 input and single output unit). The last column (i.e. the $N+1$th) in the matrix specifies the threshold biases of the units.

The representation used clearly defines the layers of the network and, thus, the translation from genotype to phenotype can be more easily interpreted. The crossover operator applied in this design strategy involves selecting a random row of the constraint matrix and swapping all the entries in that row between the parents. This is the simplest and the safest way of applying the crossover operator since any row contains a

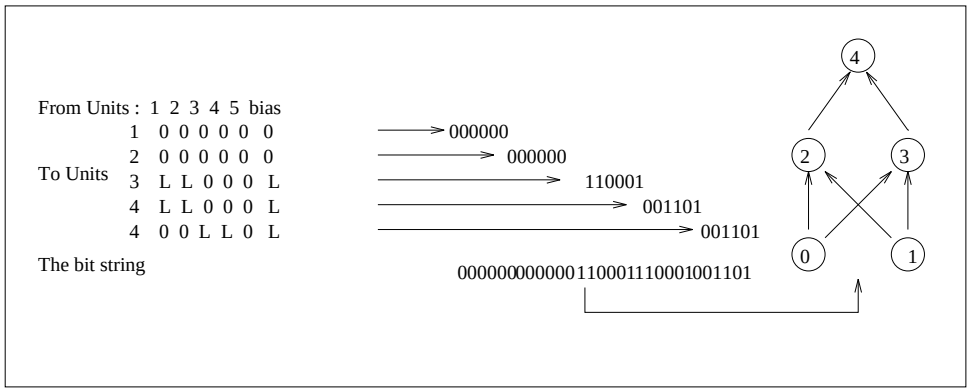

Figure 1: Encoding for XOR problem using strong representation (taken from [31]).

problem. The results have shown that such genetic based design can discover successful architectural solutions with faster learning of the tasks at hand. However, it is limited to encode a fixed number of neurons. It gives a chromosome of length n^2 for a network of n units. If the number of the units gets large the search space becomes too big. Moreover if the weights are also encoded then the crossover operation may result in non-functional off-springs: the structural/functional problem pointed out by [37].

3 Generative Strategies

3.1 Kitano's Grammar Encoding method

A system developed by Kitano [27] employs a different approach encoding ANN architectures. It uses a graph-generation grammar which can encode regular connectivity patterns with shorter chromosomes. Basically, it involves encoding a set of rules which can generate the ANN. Kitano argues that previous design strategies encode ANN configurations directly onto the chromosome and therefore require longer chromosome length and larger search space. As the size of the networks grows the time it takes to converge to a near-optimal configuration will increase. So, they are not suitable for designing large networks. Besides, these methods assume a rigid, one-to-one correspondence between the connectivity patterns and the generic information. This creates a substantial difficulty in encoding a network with repeated patterns and complex internal structure. Therefore, they are also biologically less plausible with respect to morphogenesis of the neural system.

In his design strategy Kitano's grammar generates a family of matrices of the size 2^k. The elements contained in these matrices are some characters of a finite alphabet. A larger matrix is developed using rewrite rules corresponding to these characters. This is translated to a connectivity matrix which describes the structure of an ANN.

Kitano's *grammar encoding method* is different from the previous methods where the structure of the network is not directly encoded in the chromosome. Rather, this method uses a set of re-write rules encoded in the chromosome to generate networks. It is based on Graph L-system which is an extension of Lindenmayer's L-system [28] [29]. Figure 2 shows the generation of a typical XOR network using Kitano's graph generation system.

The followings are the rules used in developing the connectivity matrix for the XOR network. Starting from the initial state "S" the graph is developed by rule-matching in

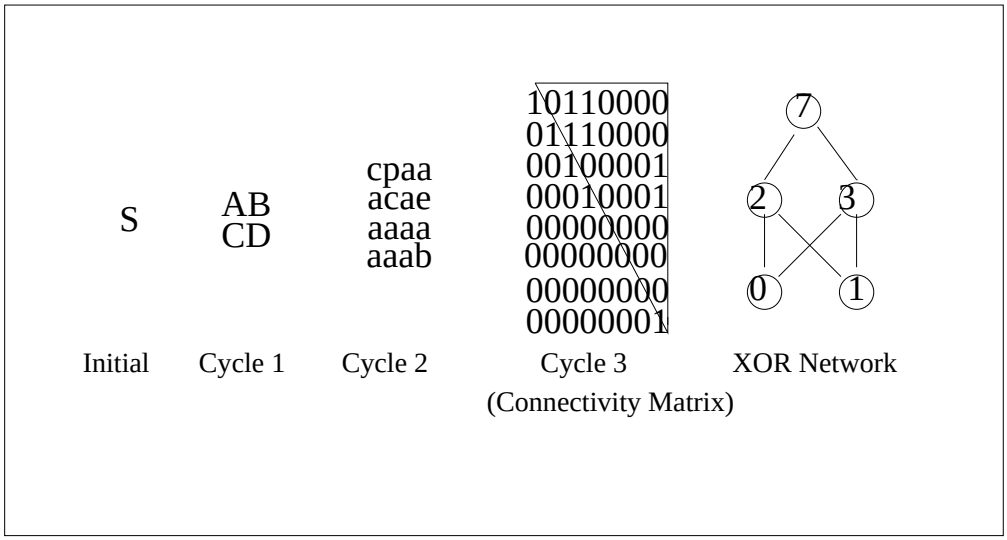

Figure 2: Generation of XOR graph (taken from [27].

each cycle.

```
        A B
S -->   C D

        c p          a a          a a          a a
A -->   a c    B --> a e    C --> a a    D --> a d

        0 0          0 0          1 0          0 1          1 1
a -->   0 0    b --> 0 1    c --> 0 0    e --> 0 1    p -->   1 1
```

For example, in the first cycle start symbol 'S' is re-written using the relevant rule. There after, for every symbol at the right hand side of first rule, the relevant rule is processed. At the end the connectivity matrix is developed which shows the existence of a connection between two units by a "1" and the non-existence of a connection with a "0".

A typical chromosome representing a network has two parts: a variable and a constant part. The constant part does not change and contains a set of static rules that are used to re-write symbols. The genetic algorithm is only applied to the variable part to acquire rules through a selection process. Since the constant part is not involved in the recombination and mutation processes, the length of the variable part constitutes the chromosome length. Each of these parts are divided into some fragments. Each fragment is made up of five bits representing a rule, in which the first bit represents the left-hand-side of a rule and the rest represent the right-hand-side of a rule. For example, the first rule

```
            AB
S -->       CD
```

(among the rules above) would be represented as follows:

In order to ensure that a cell division will always take place, the beginning of the chromosomes will always contain "S"; the initial state. The variable part contains symbol-generating rules in the range between "A" to "p" and the constant part contains pre-encoded re-write rules of symbols from "a" to "p".

The grammar encoding system is tested on XOR, 4-X-4 and 8-X-8 encoding problems using the back-propagation learning rule on feed forward networks.

The results of several experiments showed that the grammar encoding method converges much faster than direct encoding methods. Also it creates more regular network connections than direct encoding methods would normally do. This means that the grammar-encoding system shows a better scaling property and ability to generate more complex networks. Finally, it is biologically more plausible since the connectivity information is encoded in the chromosome in a more flexible manner.

Although with the grammar encoding method the same abstraction of the rules can

This strategy is biologically more plausible and efficient than matrices. The language used to describe network structures is more elegant and compact and suitable for the genetic algorithms. It also allows for coding of the weights. Various properties of this strategy have been formalised by Gruau in [14]. Some of these can be summarised as follows:

1. Completeness: any network can be encoded using the CE strategy.
2. Compactness: the ANN representations created by CE and manipulated by the GA are of minimal size. This ensures a reduction in the genetic search space.
3. Closure: the process of CE is closed under GA. It always produces meaningful structures, for either acyclic or recurrent neural networks, via the reproduction process.
4. Modularity: for larger decomposable networks, the code of the network is the concatenation of the codes of subnetworks. This result in formation of the building blocks which can be used in several different places in a typical ANN structure. It also implies more regular ANN structures.
5. Scalability: the complexity of the problem is not reflected in the representation schema. A family of ANNs can be encoded with a fixed size code.
6. Power of expression: the CE strategy can be used to encode both the architecture and the weights.

3.3 More Generative Methods

In [5] a combination of L-systems, production rules and the GA is used to design modular ANNs. The system uses L-systems as a basis for re-writing production rules which constitute string representation of the network topologies. Thus, a chromosome encodes an ANN structure in the form of production rules. The GA is used to evolve a population of these representations. The fitness of each population member is determined by the residual error after a certain amount of back-propagation training.

Another system, which is also inspired by the Kitano's work is presented by Voigt *et al.* in [16] [17]. In this approach a *stochastic* L-system is used. Although the basic algorithm involves a grammar-encoding schema similar to that of Kitano's, the production rules used are probability-dependent. This aspect has been shown to be useful in preventing the generation of large numbers of redundant production rules.

After the network structure is generated in the same way as it is in Kitano's system, the sub-networks are iteratively and randomly modified. Sub-networks are chosen in a probabilistic manner. This corresponds to an individual development process. The GA is applied to a population of individuals who have passed through this process. This strategy also uses feed-forward network structures with the back-propagation learning rule. The fitness criterion applied is interesting: it is determined by mixing learning-error, classification-error, number of training iterations, number of connections, and minimal and maximal path-length in the network structure.

Finally, another generative method presented in [45, 43, 44] uses emergent modelling to construct ANNs within an incremental, comprehensive and biologically plausible life cycle of development, plasticity, natural selection and genetic changes. The ANNs are represented by a set of production rules describing the local behaviors. They are organised hierarchically. At the lowest level are cells with their connections; next comes the individual with its behavior and, at the top level ,the environment encloses all. Similar to Gruau's approach, starting from a single cell, the system controls the division of the cells and the growth of the connections. The system works in a similar way to

the knowledge based systems. This strategy can encode feed-forward networks as well as recurrent networks. However, the use of GAs for rule-generation and recombination is limited.

4 Other Designs

There are quite a number of works in which ANNs are created and evolved as part of specific research interest. These either adapt the design strategies mentioned in this paper or create their own. For example, in [7] and [50]

[18] S. A. Harp, T. Samad, and Aloke Guha. Toward a genetic synthesis of neural networks. In J.D. Schaffer, editor, *Proceedings of Third International Conference on Genetic Algorithms*, 1989.

[19] I. Harvey. Adding species adaptation genetic algorithms:a basis for a conitinuing saga.

[40] J.D. Schaffer, D. Whitley, and L. Eshelman. Introduction. In D. Whitley and J.D. Schaffer, editors, *COGAN-92, Combination of Genetic Algorithms and Neural Network*. IEEE Computer Society Press, 1992.

[41] P. Spiessens and J. Toreek. Massively parallel evolution of recurrent networks: an approach to temporal processing. In Varela and Bourgie, editors, *First European conference on Artificial Life*, 1992.

[42] P. M. Todd and G. F. Miller. Exploring adaptive agency ii: simulating the evolution of asociative learning. In Meyer et al., editor, *From animals to animats: simulation of adaptive behavior*, 1991.

[43] Jari Vaario. *An Emergent Modeling Method for Artificial Neural Networks*. PhD thesis, The University of Tokyo, 1993.

[44] Jari Vaario and Setsuo Ohsuga. Adaptive neural architectures through growth control. In Cihan H. Dagli, Soudar R.T. Kumara, and Shin, editors, *Intelligent Engineering Systems through Artificial Neural Networks*, pages 11–16. ASM Press, New York, 1991.

[45] Jari Vaario and Setsuo Ohsuga. An emergent construction of adaptive neural architectures. *Heuristics - The Journal of Knowledge Engineering*, 5(2), 1992.

[46] D. Whitley and T. Hanson. Optimizing neural networks using faster, more accurate genetic search. In J.D. Schaffer, editor, *Proceedings of Third iInternational Conference on Genetic Algorithms*, 1989.

[47] D. Whitley and J.D. Schaffer. *COGAN-92, Combination of Genetic Algorithms and Neural Network*. IEEE Computer Society Press, 1992.

[48] D. Whitley, T. Starkweather, and C. Bogart. Genetic algorithms and neural networks: optimising connections and connectivity. *Parallel Computing*, 14:347–361, 1990.

[49] L.D. Whitley, S. Dominic, and R.Das. Genetic reinforcement learning with multilayer neural networks. In Belew and Booker, editors, *Proceedings of Fourth International Conference on Genetic Algorithms*, 1991.

[50] A.P. Wieland. Evolving controls for unstable systems. In Touretzky et al, editor, *Connectionist Models*. Morgan Kaufmann, 1990.