*Complex Systems* 4 (1990) 461–476

# Designing Neural Networks Using Genetic Algorithms with Graph Generation System

**Hiroaki Kitano**
*Center for Machine Translation, Carnegie Mellon University, Pittsburgh, PA 15213 USA*
and
*NEC Corporation, Tokyo, 108 Japan*

**Abstract.** We present a new method of designing neural networks using the genetic algorithm. Recently there have been several reports claiming attempts to design neural networks using genetic algorithms were successful. However, these methods have a problem in scalability, i.e., the convergence characteristic degrades significantly as the size of the network increases. This is because these methods employ direct mapping of chromosomes into network connectivities. As an alternative approach, we propose a graph grammatical encoding that will encode graph generation grammar to the chromosome so that it generates more regular connectivity patterns with shorter chromosome length. Experimental results support that our new scheme provides magnitude of speedup in convergence of neural network design and exhibits desirable scaling property.

## 1. Introduction

Use of genetic algorithms for neural network designing encompasses two major attractive features. First, it automatically discovers optimized network structures for given tasks in which researchers had to manually carry out trial-and-error processes to find near-optimal network architecture. Second, it is analogous to an actual biological process in that the blueprint is encoded in the chromosome, which can be changed through the evolutional process, and neural networks are created based on information encoded in the chromosome.

Recently there have been a few studies on the use of genetic algorithms for automatic designing of neural networks, including [9] and [15]. These reports claim effectiveness of genetic algorithms for neural network designing. Actually, their experimental results indicate that an optimized network can be generated through an evolutional process simulated by the genetic algorithm. There are two major problems in these approaches. First, none of

these studies carried out systematic experiments in terms of scalability and the speed of convergence, leaving applicability of the schemes for designing larger networks open to question. In fact, we will demonstrate in the next section that these methods have an undesirable scaling property. Second, these methods are biologically unfeasible, because they assume that connectivity information is encoded in the DNA in almost one-to-one correspondence. This leads to two major problems: (1) it cannot capture morphogenesis of neural systems, and (2) sufficient information cannot be encoded in DNA of the given length. Thus, existing methods do not take full advantage of using genetic algorithms for neural network designing.

The purpose of this paper is to present a new method of neural network designing using genetic algorithms that surpasses current methods in both efficiency and biological plausibility. We employ a completely different approach from the existing methods in encoding neural network architectures. Instead of directly encoding network configuration, as seen in the existing methods, our scheme encodes a graph generation grammar that defines the growth of graphs. The graph generation grammar is augmentation of Lindenmayer's L-system, which is designed to describe morphogenesis [13, 12]. Although at this moment we have no decisive evidence to prove the graph L-system to be the underlying mechanism of morphogenesis, the fact that the system has successfully described several morphogenesis gives us strong rationale to believe that our approach is biologically more plausible than the existing methods.

We believe that this difference of how to encode genetic information is a significant factor in the speed of convergence. Our paradigm is that the evolutional process operates upon the system that comprises the central dogma. In this framework, the major thread of processing is that the genetic information, called *genotype*, is interpreted to create an actual individual, called *phenotype*, and that the natural selection forwards evolution of the population. In this process, the fitness to the environment is evaluated for each individual — not DNA itself, and what is selected to the next generation is the DNA itself — not a copy of individuals that have undergone learning in the environment. There are two factors that largely affect the speed of evolution in this scheme:

1. representation of genetic information encoded in the DNA and its interpretation to create phenotype, and
2. how each individual is trained and evaluated until the end of its reproduction period.

Since the second factor can be homogeneous in neural network designing that we are considering in this paper, we will focus on the first factor. Our working hypothesis is that the representation and the interpretation scheme that is more likely to be used in the actual biological process is the most efficient one, since the scheme is actually selected through the evolution process.

In the next section, we will demonstrate that the current scheme of neural network design by genetic algorithms has undesirable scaling property.

We will then describe our scheme using the graph generation grammar, and experimental results will be shown.

## 2. Problems of the direct encoding method

Current research in genetic algorithm-based neural network design methods uses an approach that directly encodes network configurations. A common feature of various versions of this approach is that these methods directly encode network connectivity onto the chromosome. We call this method a *direct encoding* method. Miller et al. [15] employs what they called a *strong specification* scheme, which represents connectivity matrix by a string of bits. Obviously, this scheme degrades its convergence performance as the size of network grows. Harp et al. [9] use clusters of areas that define the size of the area and projections going out. Since their methods do not strictly define each connection, they call it a *weak specification* scheme. The weak specification scheme, however, does not escape from the scalability problem when it needs to define large and complex networks, because each area has to encode connectivity in nearly one-to-one fashion, and there is no way to efficiently encode repeated patterns with complex internal structure. Therefore, although there may be some differences in the two methods, both require longer chromosome length as network size increases, and search space will be increased accordingly. Such schemes will degrade their convergence performance as the size of the networks grows. To support our speculation with regard to the scaling property of the direct encoding method, we have carried out a series of experiments designed to identify the scaling property.

The task we used is the 4-X-4 encoder/decoder problem. This is essentially the N-M-N encoder/decoder problem where the numbers of input and output nodes are four. Since we do not know how hidden layers can be configured, it is shown as "X." Experiments are conducted for networks of size 10, 20, and 30. The length of a chromosome are 100, 400, and 900, respectively.

We used a strong specification scheme as used in [15]. Figure 1 shows how we represent the 4-2-4 network in a connectivity matrix and in a chromosome with the strong specification direct encoding method. In the strong specification direct encoding method, each point of the chromosome directly corresponds to each grid of the matrix.

We used a proportional reproduction strategy in which reproduction probability is decided based upon the fitness of each chromosome. The elitist copying is introduced so that the best chromosome is copied into the next generation. Mutation is adaptive mutation with probability varying from 2% to 30%. Crossover probability is 50%. We used multipoint crossover as well as single- and two-point crossover, although no significant difference has been observed. Each position of the chromosome takes value either "1" or "0," and the value "1" means that the connection is established. Probability of the "1" to be generated at each position of the chromosome is set to 30%. Population size was 10. We also have changed population size to see if our result shown in figure 2 can be observed generally, and we confirmed that

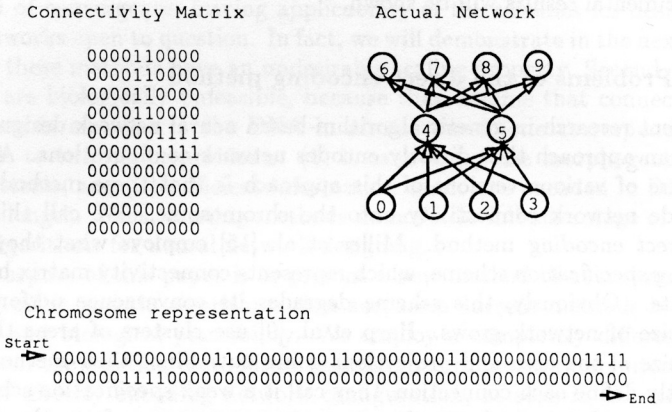



Figure 1: Representation of the 4-2-4 network.

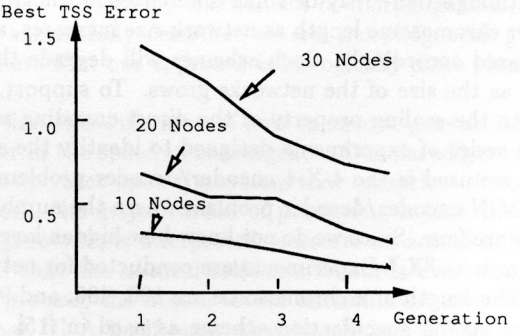



Figure 2: Scalability of the direct encoding method.

this trend is observed generally regardless of its population size. For an error measure, we used total sum square error (TSS error), and $1/\text{TSS}^2$ was used as a fitness measure. The TSS error actually used for fitness evaluation is a value after 10 epochs of backpropagation. For backpropagation, we used a standard backpropagation as described in [14] (the same parameter values are used). When the initialization is conducted for the network defined by the genetic algorithm part, weights and bias are randomly generated in range from $-1.0$ to 1.0. In backpropagation training and evaluation, only a upper-right triangle is used so that reverse connection is pruned out.

Experimental results show that the speed of convergence degrades as the network is scaled up (figure 2). The figure is plotted by the average TSS error of 20 trials for each size of the network. It should be noted that the

experiment has been carried out on a simple toy domain; in real domains, much larger networks with highly structured architecture are commonly demanded. It is questionable whether the direct encoding method is capable of generating network architecture with highly functional structures through reasonable evolution cycles. We need a faster convergence method capable of generating highly structured network architecture.

## 3. Graph generation system

We propose an alternative approach that acquires rules for graph L-system or graph generation system over generations. This approach, which we call the *grammar encoding method*, is different from previous methods in that network configuration is not directly encoded in the chromosome; instead, a set of rules that generate networks is encoded.

Graph L-system is an extension of the L-System [13, 12], which captures development of cells at early stage of development. The L-system has successfully described a development of such living organs as *Callithamnion roseum* and *Aster novae-angliae*. Graph L-system was proposed in [5] as an extension of the Lindenmayer's L-system to generate graphs, and it applied to describe development of *Halocynthia roretzi* (ascidian or sea squirt) and *Caenorhadbitis elegans (C. elegans)*.

The graph L-system or the graph generation system can be defined as L = (SN,SE,MN,ME,RN,RE,$\omega$) whereas SN, SE, MN, ME, RN, RE, and $\omega$ denotes a set of symbols representing a state of a node, a set of symbols representing a state of a edge, a set of $2 \times 2$ matrix with symbols for nodes assigned at diagonal elements and a symbol for an edge assigned at a nondiagonal element with an empty lower-left element, a set of $2 \times 2$ matrix with 0 and symbols for edge, rewriting rules for nodes, rewriting rules for edges, and an initial graph. With this definition, symbols on diagonal elements represents type of cells, and symbols on nondiagonal elements represent types of connections. In describing a real morphogenesis event, the system must be context-sensitive so that different generation rules are used depending upon location and neighbors of each cell. In this paper, we use a context-free and deterministic model so that effects and behaviors of this new approach are tractable.

In the experiment described in this paper, we have employed a modified version of the graph generation system because the graph generation system described above cannot generate graphs for recurrent network: it only generates an upper-right triangle matrix. Although we only need a upper-right triangle in our connectivity matrix in the experiment in this paper, because we only examine feed-forward networks, the graph generation system for neural network should be able to generate recurrent network, too. Thus, our graph generation system does not assume an empty lower-left element in matrices for MN, so that entire matrices can be generated.

In real development, context sensitivity, such as physical location of the cell and which types of cells are located next to the cell, would be an im-

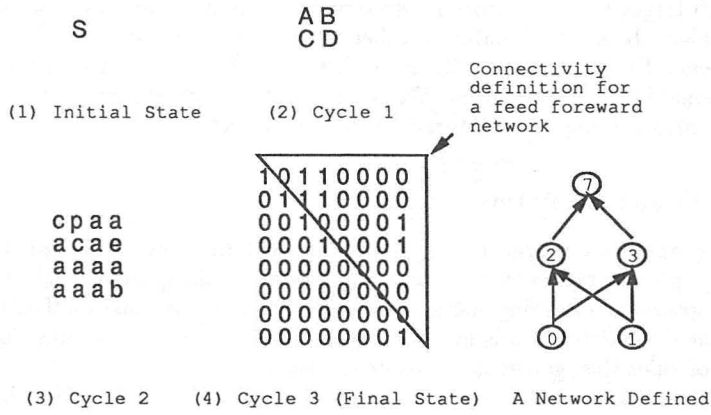



Figure 3: Development of the 2-2-1 XOR network.

$$S \rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad A \rightarrow \begin{bmatrix} c & p \\ a & c \end{bmatrix} \quad B \rightarrow \begin{bmatrix} a & a \\ a & e \end{bmatrix} \quad C \rightarrow \begin{bmatrix} a & a \\ a & a \end{bmatrix} \quad D \rightarrow \begin{bmatrix} a & a \\ a & d \end{bmatrix}$$

$$a \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad b \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad c \rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad e \rightarrow \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \quad p \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Figure 4: Graph generation rules used for generation of the 2-2-1 XOR network.

portant factor to determine actual cell division patterns and differentiation. However, we use a simplified model and employ a context-free model.

In figure 3, we illustrate how the standard XOR network can be generated using the graph generation system. Figure 4 shows rules involved in the development of the network. It starts from an initial state with a single symbol "S" (figure 3(1)). At the first cycle, the initial graph was rewritten by the first rule in figure 4. At the second cycle, each symbol is rewritten by the corresponding rewriting rule. The connectivity matrix grows to the predetermined size after three rewriting cycles (figure 3(4)). Symbols "1" and "0" denote existence and nonexistence of connection respectively. It should be noted, however, that symbols "1" and "0" are not terminal symbols. In the next cycle, they can be rewritten in a $2 \times 2$ matrix of full connections (all elements are "1") or nonconnections (all elements are "0").

## 4. Experimental settings

This section describes experimental settings of the controlled experiments designed to identify performance of the grammar encoding method over the direct encoding method. For the direct encoding method, we use the same parameters described in section 2. For the grammar encoding method, a

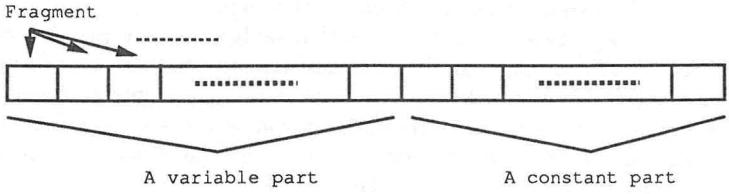



Figure 5: A variable and a constant part of a chromosome.

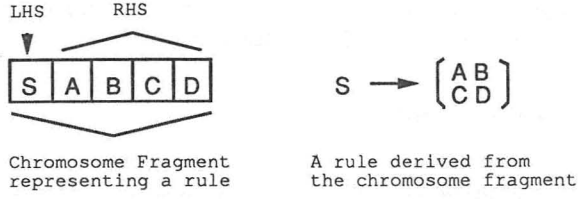



Figure 6: Grammar encoding on chromosome.

basic genetic algorithm is identical to what is used for the direct encoding method. The only differences are the content of chromosomes (how to represent networks) and chromosome length, due to the different representation scheme.

The chromosome consists of a variable and a constant part (figure 5.) The variable part is a part of the chromosome that actually acquires rules through a selection process. Genetic operations, such as mutations and crossovers, are performed on this region. The chromosome length in this method refers to the length of the variable part because the constant part is not involved in the recombination and mutation process. The constant part does not change, and 16 rules that rewrite symbols for nondifferentiated cells into symbols for differentiated cells are pre-encoded. Each rule is encoded as five allele positions (figure 6). For the variable region, symbols are generated for each position of the chromosome in range between "A" to "p." The first position of the chromosome, however, is fixed to be "S" (an initial graph), so that the rule for the first cell division can always be created. For the constant region, the left-hand side (LHS) has pre-assigned unique symbol between "a" to "p," corresponding to the 16 possible patterns for the right-hand side (RHS). The RHS of the rule consists of "1" and/or "0."

Each chromosome is interpreted to form the network to be trained. At the end of the entire rewriting cycle, if a certain position of the connectivity matrix is still a nondifferentiated cell (neither "1" nor "0"), it is considered to be dead and no connection is created. In this experiment, we use a simple version of the graph generation system so that we ignore cell types that could be represented in the diagonal elements and assume that the first N nodes function as input nodes and the last N nodes function as output nodes.

The network created by interpreting the chromosome will be trained using backpropagation. Of course, the same backpropagation parameters used for the direct encoding method are used here.

As tasks that we conducted in our experiments, we employed various sizes of the encoder/decoder problems. This is because (1) the encoder/decoder problem is easily scalable, and (2) it is easily accessible by other researchers in order to replicate our results. Of course, experiments with larger and real-world tasks are necessary to make decisive conclusions. However, at this moment, we need to examine the various properties of competing methods with more tractable and accessible tasks so that the community can explore new aspects of this new and unknown techniques.

## 5. Experimental results

We carried out several sets of experiments to compare the speed of convergence and scaling property of the direct encoding method and the grammar encoding method, and we confirmed that the grammar encoding method consistently outperformed the direct encoding method. For each experiment we ran 20 trials, and figures are drawn based on the average TSS error of the best chromosome in each trial.

The first experiment was conducted on the 4-X-4 encoder/decoder problem with networks of size 16 ($16 \times 16$ connectivity matrix). Figure 7 shows a result of the experiment. The TSS measure of the network created from the best chromosome is plotted aligned to the generation. The TSS measure is a value after training the network with 10 epochs. This is same for the rest of experiments.

In the second experiment, we have scaled up to the 8-X-8 encoder/decoder problem with networks of size 32 ($32 \times 32$ connectivity matrix). Similar to the previous experiment, superiority of the grammar encoding method was clearly demonstrated (figure 8). In this experiment, we ran a set of population size 10 and a set of population size 100. Results from the population size 10 are drawn by solid lines and results from the population size 100 are shown by dotted lines. The grammar encoding method seems to get more benefit from increased population size.

In the next set of experiments, we investigated the scaling property of the direct encoding method and the grammar encoding method. We have scaled up the network size from 16 to 64 using the 4-X-4 encoder/decoder for region from 16 to 32 and the 8-X-8 encoder/decoder for region from 32 to 64. Figure 9 shows results of this experiment. Although the direct encoding

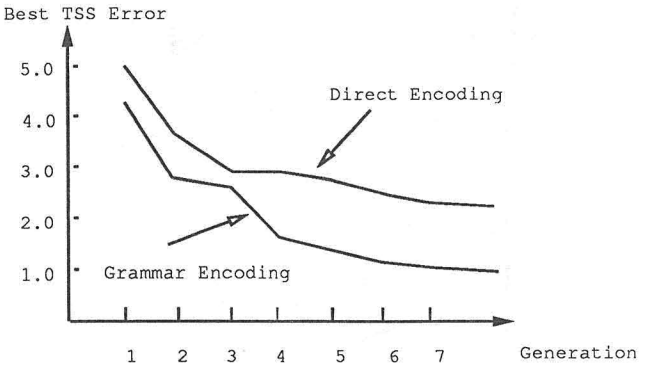



Figure 7: Convergence in the 4-X-4 problem.

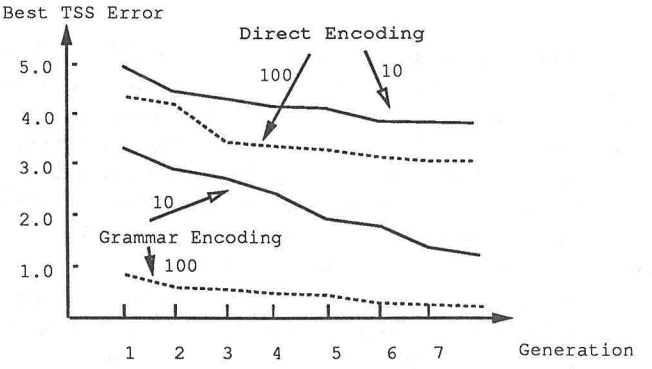



Figure 8: Convergence in the 8-X-8 problem

method suffered from scaling up the network size, the grammar encoding method does not seem to be affected much.

## 6. Discussions and future studies

### 6.1 Speed of convergence and scaling property

The superior convergence and scaling property of the grammar encoding method over the direct encoding method has been clearly demonstrated. In any experiment, the grammar encoding method outperformed the direct en-

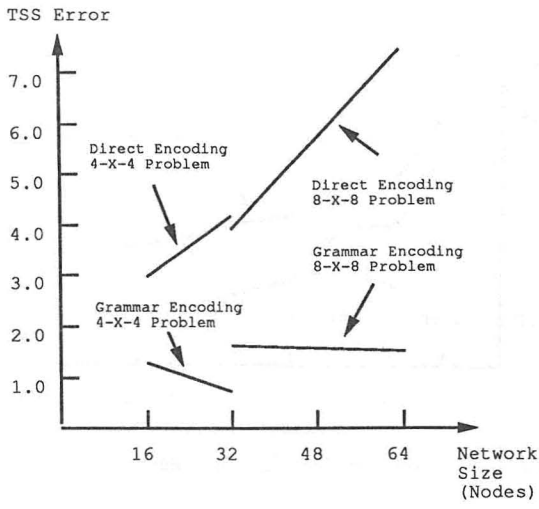
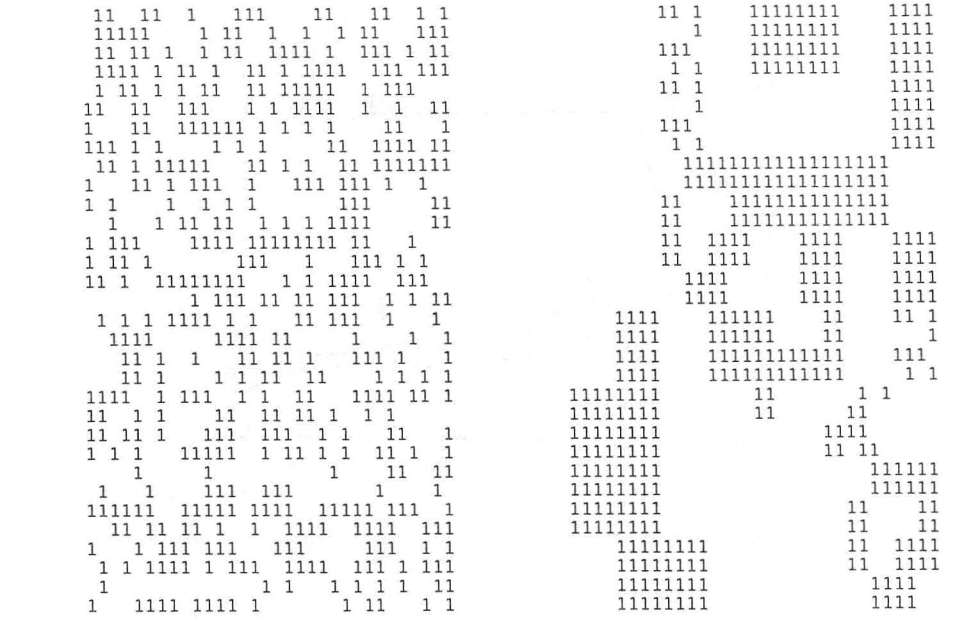



Figure 9: Scalability of direct encoding and of grammar encoding method.

coding method with a significant margin. The comparison of the convergence with different size of networks (figure 9) shows dramatic difference between the grammar encoding method and the direct encoding method. While the direct encoding method suffers from degraded convergence speed, the convergence speed of the network generated by the grammar encoding method does not seem significantly affected by the size of the network created. The superior performance of the grammar encoding method over the direct encoding method can be attributed to several major reasons.

First, the grammar encoding method generates more regular patterns than the direct encoding method. Figure 10 shows the connectivity matrix generated from the best chromosome at the 10th generation. Apparently, the grammar encoding method creates far more regular connectivity patterns than the direct encoding method. In the direct encoding method, probability of creating a cluster of connection of a size $4 \times 8$, as seen in the upper center of figure 10(B), would be of the order of $1/2^{32}$, whereas in the grammar encoding method the figure would be in the order of $1/16^2$. This is due to the fact that the direct encoding method does not entail intrinsic mechanism to generate regular network connection patterns, whereas the grammar encoding method ensures regular patterns emerge from its acquired grammar. Generation capability of regular patterns is important because in the complex tasks, we need several neural modules with similar functionalities but which are tuned for different input patterns and reactions. For example, *time delay neural network* (TDNN) [18] assumes several multilayered feed-forward networks, each of which recognizes certain phonemes from sound spectrum patterns.



(A) Direct Encoding Method (B) Graph Generation Grammar Method

Figure 10: Final networks for the 8-X-8 problem.

Also, it is biologically attractive because the existence of regular repetitive patterns are observed in many regions of the brain such as the hypercolumn clusters in the visual cortex.

Second, the grammar encoding method better preserves and is capable of copying the meaningful subcircuits discovered. This is because these patterns are preserved in grammar rules that can be localized in the chromosome so that the rules are less likely to be disturbed by crossover. In addition, reuse of the discovered local circuit is easy because each of the local circuits is likely to be under a symbol that can be further rewritten to generate the circuit. If other parts of the rule include this specific symbol, the same pattern will appear in different parts of the network. However, the direct encoding method is not capable of preserving local circuits because connectivities of these local circuits are represented distributively in the chromosome, so that these local patterns are likely to be lost by the recombination process. Also, the direct encoding does not allow discovered local circuits to be used in various different parts of the network, because there is no means to reuse or copy its circuit topology.

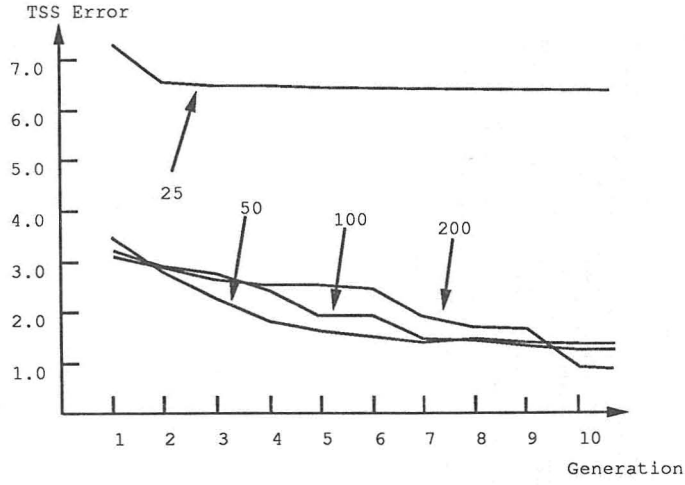



Figure 11: Chromosome scalability of grammar encoding method.

Third, the grammar encoding method requires shorter chromosome length than the direct encoding method. In most experiments described in this paper, the chromosome length of the grammar encoding method was 100, allowing 20 rules to be acquired. The direct encoding method required chromosome length ranging from 256 (for 16-node networks) to 4096 (for 64-nodes networks). Theoretically, the grammar encoding method does not cover the whole search space with the chromosome of the length 100. However, our experimental results indicate no significant problem due to the short chromosome length. To confirm this point, we ran the 8-X-8 encoder/decoder problem with network size of 32 with varying chromosome length (25, 50, 100, and 200). The result is shown in figure 11. An experiment with different chromosome length (figure 11) shows that, except with a very short chromosome (25 elements), the grammar encoding method outperforms the direct encoding method (compare with figure 8), and similar convergence curvatures have been observed. Of course, there are some effects due to the chromosome length. While the shorter chromosome (50) converges quickly at the initial few generations but slows down drastically later, the longer chromosome (200) converges a bit slowly but it converges down to lesser error measures after the 10th generation. This indicates that though the longer chromosome works better in the long run, the shorter chromosome does just fine as far as it is longer than a certain length. From this observation, we can generally assume that the grammar encoding method requires shorter chromosome length than the direct encoding method without significantly undermining its final convergence level.

### 6.2 Incorporating neural plasticity

Although the grammar encoding method generates a much more regular connectivity matrix, the generated network is not guaranteed to be a fully optimized network. In fact, there are some redundant connections that can be pruned out. Also, there may be cases in which the addition of some nodes or connections improves the speed of convergence and the final performance. Theoretically, even such a detailed fine-tuning can be done with sufficient numbers of generation and with a larger population. However, in the genetic algorithm, the convergence speed for detailed fine tuning tends to degrade compared to the initial speed of convergence. Thus, it would be more efficient to configure networks with genetic algorithms, just enough to determine overall architecture, and use different mechanisms to make detailed adjustments on creation and elimination of nodes and connections. Fortunately, there are some investigations available on dynamic node creation and eliminations [7, 1, 11]. Since these node creation and elimination methods assume rather simple networks to start with, it is a subject of further investigation to discover an architecture that can be effectively applied to more complex initial network configurations.

This cascaded combination of genetic algorithm-based network design and dynamic creation and elimination of nodes and links is a physiologically plausible approach. It is well acknowledged that genetic information cannot fully configure connectivity of neurons in the brain. While estimated numbers of neurons are of the order of $2 \times 10^{10}$ and the numbers of synapses may be around thirty thousand times higher, the numbers of gene coding for protein is roughly estimated to be in the range of $10^4$ to $10^5$ [8]. Considering that not all genes encode neural connectivity, we can reasonably assume that genetic information determines only a framework of neural circuits and detailed circuits configuration are left with the capability of neural circuits to change their local network under external stimuli (*neural plasticity*). In fact, there is ample evidence supporting this hypothesis (for a good overview of such evidences, see [17]). At a certain stage of the development, there are too many neurons and many of their connections are eliminated later on [3]. However, which cells and connections are eliminated differs depending on how external stimuli are given to the neural system. The death of cells is programmed in DNA (programmed cell death), but it is not fully deterministic because it is affected by external stimuli. This can be simulated by our scheme by (1) introducing eliminatable nodes and connections in addition to current "1" and "0" states, and (2) incorporating dynamic nodes and connection elimination mechanisms similar to the optimal brain damage model [11]. Although some cells and connections are eliminated, there is a significant increase in synaptic connections following the programmed cell death [4]. A mechanism that enables dynamic creation of new connections should be incorporated into our scheme in order to simulate this phenomena.

Sperry [16] and Jacobson [10] report that until a certain stage of the development, detailed correspondence on the projection of nervous fibers are

not determined, and it is only after a certain stage of the development that the rigid projection is determined. This level of dynamism in neural plasticity may be useful if we cannot fully grasp a set of complex tasks with which neural networks have to deal. Whether our scheme is capable of incorporating such a mechanism can be examined by training networks in such a way that the presentation of tasks are flipped at a certain point of the chromosome interpretation cycle. To do this, we need to extend our scheme to allow presentation of external stimuli even during interpretation of the chromosome so that external stimuli affect each stage of network development. It is a subject of further research to incorporate some findings and theories of neurogenesis such as the theory of selective stabilization [2] and the theory of neural group selection [6].

## 7. Conclusion

In this paper, we proposed a new scheme of neural network designing using genetic algorithms. The new scheme uses the graph generation grammar to encode network configuration. The most current scheme of neural network designing employs the direct encoding method, which directly maps network connectivity to the chromosome. However, the direct encoding method was discovered to have an undesirable scaling property so that it is not suitable for designing larger networks. The new scheme, which we call the grammar encoding method, encodes a graph generation grammar that defines development patterns of neural cells. The grammar is to be interpreted for a few cycles, similar to the L-system, and a network of the desired size is generated.

Benefits of the proposed scheme can be described as follows:

The grammar encoding method converges much faster than the direct encoding method. A series of experiments demonstrated that the grammar encoding method consistently outperformed the direct encoding method in speed of convergence. It also exhibits a significantly better scaling property than the direct encoding method.

The grammar encoding method generates far more regular network connectivity patterns than that created by the direct encoding method. This ensures better scalablilty and the possibility to generate very complex networks for more difficult real-world tasks.

Biological plausibility is one of the most attractive points of the scheme. Since the L-system and the graph L-system have been successfully applied to the description of morphogenesis, it is plausible that an actual network can be formed in a similar manner. Of course, we do not believe that the actual network development directly corresponds to the development patterns incorporated in our scheme, but it is reasonable to assume that there may be substantial similarity at an abstract level.

The fact that the grammar encoding method exhibited desirable convergence and scaling properties implied that the grammar encoding

method may actually be used in the real DNA encoding. Obviously, we do not have any decisive evidence to claim this hypothesis at this moment; however, it is conceivable that this computationally efficient method is selected in nature after a long evolution process.

## References

[1] T. Ash, *Dynamic Node Creation in Backpropagation Networks*, ICS Report 8901, The Institute for Cognitive Science, University of California, San Diego (Saiensu-sha, 1988).

[2] J. Changeux and A. Danchin, "Selective stabilization of developing synapses as a mechanism for the specification of neuronal networks," *Nature*, **264** (1976) 705–712.

[3] W. Cowan, J. Fawcett, D. O'Leary, and B. Stanfield, "Regressive events in neurogenesis," *Science*, **225** (1981) 1258–1265.

[4] B. Cragg, "The development of synapses in the visual system of the cat," *J. Comp. Neurol.*, **160** (1975) 147–166.

[5] Y. Doi, *Morphogenesis of Life Forms* (Saiensu-sha, 1988).

[6] G. Edelman, *Neural Darwinism: The Theory of Neuronal Group Selection* (Basic Books, New York, 1987).

[7] S. Fahlman and C. Lebiere, *The Cascade-Correlation Learning Architecture*, Carnegie Mellon University Technical Report, CMU-CS-90-100, 1990.

[8] A. Gierer, "Spatial organization and genetic information in brain development," *Biol. Cybern.*, **59** (1988) 13–21.

[9] S. Harp, T. Samad, and A. Guha, "Towards the genetic synthesis of neural networks," *Proceedings of the International Conference on Genetic Algorithms*, 1989.

[10] M. Jacobson, "Development of neuronal specificity in retinal ganglion cells of xenopus," *Develop. Biol.*, **17** (1968) 202–218.

[11] Y. Le Cun, J. Denker, and S. Solla, "Optimal brain damage," *Advances in Neural Information Processing System 2* (Morgan Kaufmann, 1990).

[12] A. Lindenmayer, "Mathematical models for cellular interactions in development," *J. Theor. Biol.*, **18** (1968) 280–299.

[13] A. Lindenmayer, "Developmental systems without cellular interactions, their languages and grammars," *J. Theor. Biol.*, **30** (1971) 455–484.

[14] J. McClelland and D. Rumelhart, *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises* (MIT Press, 1988).

[15] G. Miller, P. Todd, and S. Hedge, "Designing neural networks using genetic algorithms," *Proceedings of the International Conference on Genetic Algorithms* 1989.

[16] R. Sperry, "Chemoaffinity in the orderly growth of nerve fiber patterns of connections," *Proc. of Nat. Acad. Sci. U.S.A.*, **50** (1963) 703–710.

[17] T. Tsumoto, *Brain and Development* (Asakura-Shoten, Tokyo, 1986) (in Japanese).

[18] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, May 1989.