Neuron Reordering for Better Neuro-Genetic Hybrids

Jung-Hwan Kim and Byung-Ro Moon
School of Computer Science and Engineering
Seoul National University
Shilim-dong, Kwanak-gu, Seoul, 151-742 Korea
{aram,moon}@soar.snu.ac.kr

Abstract

We propose an approach to improve the performance of neuro-genetic hybrids. We used a two-dimensional genetic encoding to represent the network with less information loss and devised a careful restructuring of neural network so that the geographic contributions of neurons can be better reflected in the genetic process. We test handwritten character recognition problem to examine the performance of the proposed approach. Experimental results showed significant improvement by neuron reordering and two-dimensional encoding.

1 Introduction

Artificial neural networks (ANNs) have been intensively studied for Handwritten Character Recognition (HCR) systems [2, 18, 19, 25]. There are a variety of tuning methods for ANNs. Generally *backpropagation* is the most popular local gradient training technique. However, there is no guarantee with few exceptions that such methods find an optimal neural network. They tend to become stuck at local optima.

We propose a neuro-genetic hybrid approach for improving the performance of neural networks. Genetic algorithms (GAs) can provide diverse initial solutions to backpropagation and have recently become popular [9, 14, 17, 23]. To represent solutions, GAs for ANN optimization have been using linear encodings following the convention of the GA community [10, 14, 20]. Typically, every weight in an ANN takes a position in a linear chromosome of a GA. However, linear encodings have limited capability in reflecting the geographic linkages of genes [1, 7] and a number of two-dimensional (2D) encodings have been used for other problems [1, 4, 7, 15]. The weights of an ANN can be represented by a 2D matrix and thus they are intrinsically suitable for a 2D encoding.

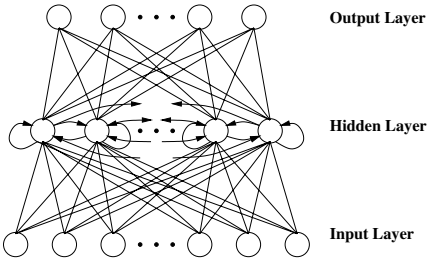


Figure 1: The recurrent neural network architecture we used

In this paper, we use a 2D encoding for genetic representation and employ 2D *geographic crossover* which has demonstrated good performance for the graph partitioning problem [15, 21, 22]. Once the structure of an ANN is fixed, the genotype depends on the order of neurons in the structure. Even with a 2D encoding, the geographic relationships between neurons may not be well reflected. We measure the relative relationships of neurons and propose a restructuring algorithm based on these relative relationships. This restructuring approach enables the hidden neurons with the relatively high relationship to be clustered.

The remainder of this paper is organized as follows. In the next section, we summarize the neural network architecture that we used. In Section 3, we describe our hybrid neuro-genetic approach and the reordering approach. We present our experimental results in Section 4 and state our conclusions in Section 5.

2 Neural Network Design

We use a recurrent neural network architecture based on Elman's recurrent neural network [8]. It consists

of two layers, excluding the input layer, as shown in Figure 1.

Each hidden unit is connected to itself and also fully connected to all the other units. These connections are updated in the propagation phase every Δt time interval. The intermediate output (ϕ_j) of the j^{th} hidden unit at cycle t is computed through the summation of two factors as in the following:

$$\begin{aligned} \phi_j(t) &= \sum_{i=1}^{n} w_{ji} y_i(t) + T \cdot \sum_{i=1}^{p} u_{ji} v_i(t-1) \\ v_j(t) &= \varphi(\phi_j(t)) \end{aligned}$$

where

- $v_j(t)$: the output value of the j^{th} hidden unit at cycle t
- $y_i(t)$: the value of the i^{th} input unit at cycle t
- w_{ji} : the weight between the j^{th} hidden unit and the i^{th} input unit
- u_{ji} : the recurrent weight between the j^{th} hidden unit and the i^{th} hidden unit
- n, p : the numbers of input units and hidden units, respectively
- T : the decay factor to control the reflection ratio of the recurrent factor
- φ : sigmoidal transfer function $\dfrac{1}{1+e^{-\lambda x}}$.

The first term is related to the feedforward connections between the input and hidden layer and the second term reflects the recurrent factor.

We represent the set of weights by a matrix $M = [m_{ij}]$ where $m_{ji} = w_{ji}$ for $1 \le i \le n$, and $m_{ji} = u_{j,i-n}$ for $n < i \le n+p$. We will create a chromosome based on this matrix. In the m_{ji}, i is the index of the source unit and j is the index of the target unit. Then, in the weight matrix, values on the same row are with the same target unit and values on the same column are with the same source unit.

The weight correction of the above recurrent neural network during error backpropagation is determined as follows:

$$\begin{aligned} \frac{\partial E}{\partial w_{ji}} &= \sum_{k=1}^{q} \{-(t_k - o_k)\varphi'(\psi_k)\pi_{kj}\}\,\varphi'(\phi_j) y_i \\ \frac{\partial E}{\partial u_{ji}} &= T \sum_{k=1}^{q} \{-(t_k - o_k)\varphi'(\psi_k)\pi_{kj}\}\,\varphi'(\phi_j) v_i \end{aligned}$$

where

- E : the error value in the training phase calculated using the mean square error
- t_k : the target output of the k^{th} output unit
- o_k : the real output of the k^{th} output unit
- π_{kj} : the weight between the k^{th} output unit and the j^{th} hidden unit
- q : the number of output units
- $\psi_k = \sum_{j=1}^{p} \pi_{kj} v_j$.

We use 28×28 images of digits as input data of the neural network. If the network has an input unit for every one of the $28 \times 28 = 784$ pixels, the problem space becomes very large and the backpropagation learning speed is considerably reduced. Kohara and Nakamura [16] suggested an alternative approach that counts the density (number) of black pixels on each row and provides the vector of the counted numbers. The horizontal density distributions (HDDs) – the spectrum of black-pixel numbers in the horizontal rows – are visibly different digit by digit. This approach can significantly decrease the number of units in the neural networks. They reported an 85.3% recognition rate for the handwritten digit recognition. When this approach is applied to our problem, the number of input units reduces from 784 to 28. This speeds up the convergence of network tuning at the cost of lower recognition quality. We find a compromise in between the two extremes. In addition to the number of black pixels in each row, we also count the number of black pixels in each column (we call this the vertical density distribution (VDD)); thus we have 56 input units. In addition, we shift every image upwards and to the left so that both the leftmost column and the topmost row have at least one black pixel. All our training patterns were fitted to a 23×22 grid. Thus the number of input units becomes 45. We used 20 units in the hidden layer; the output layer naturally has 10 units corresponding to the 10 digits. If any test pattern does not fit to the 23×22 grid, the rightmost columns and bottommost rows are thrown away. We use the *1-of-N* approach as output encoding. Each output neuron corresponds to one of the 10 digits. For each input vector, only one of 10 output neurons has value 1 and the others have value 0.

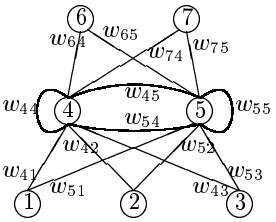
3 The Proposed Neuro-Genetic Hybrid

3.1 The GA Structure

The template of the proposed genetic algorithm is shown in Figure 2. It is a typical steady-state hybrid genetic algorithm. When a genetic algorithm is hybridized with a local improvement heuristic, it called

```
Create initial population of fixed size;
do {
    choose parent1 and parent2 from population;
    offspring = g2d_xover(parent1, parent2);
    mutation(offspring);
    backpropagation(offspring);
    reordering(offspring);
    if suited(offspring)
    then replace(population, offspring);
} until (stopping condition);
Return the best solution;
```

Figure 2: The template of the hybrid GA



(a) NN structure

	input units			hidden units		output units	
	1	2	3	4	5	6	7
4	w_{41}	w_{42}	w_{43}	w_{44}	w_{45}	w_{64}	w_{74}
5	w_{51}	w_{52}	w_{53}	w_{54}	w_{55}	w_{65}	w_{75}

(b) Weight Matrix

Figure 3: A 2D encoding for a neural network.

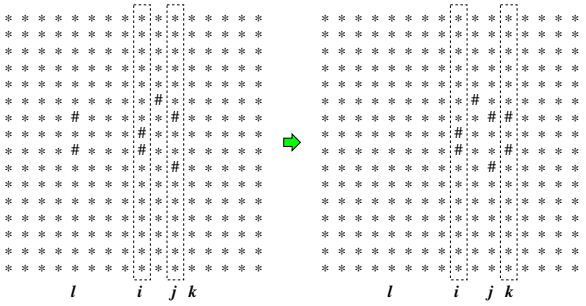


Figure 4: Examples of 2D schemata

a hybrid GA. We set the population size to be 50. Two parents are selected according to probabilities that are proportional to their fitness values. The probability that the best solution is chosen was given as four times that of the worst solution. The offspring is produced through geographic 2D crossover. The "g2d_xover" indicates the geographic 2D crossover in Figure 2. The geographic 2D crossover is described in Section 3.4. The offspring is then modified by a *mutation* operator and locally optimized by backpropagation. The backpropagation process helps the GA fine-tune around local optima. From another perspective, the GA provides diverse initial solutions to the backpropagation routine. At this point, the offspring is modified by reordering the hidden units. Of course, the modification does not affect the quality of the offspring. Then the offspring replaces a solution in the population by the following rule [5]: the more similar parent to the offspring is replaced if the offspring is better; otherwise, the other parent is replaced if the offspring is better; if not again, the worst chromosome in the population is replaced. The rational behind this is to maintain the population diversity to the extent that not too much time is wasted. For stopping, we use the fixed generation.

3.2 Problem Encoding

Most GAs for neural-network optimization encode a solution (a set of weights) with a linear string following the convention [10, 14, 20]. However, linear encodings are known to be a factor limiting GAs' performance in other problems [1, 7]. Two-dimensional encodings have been proven to perform favorably [1, 4, 7, 21, 22]. Instead of transforming into a linear string, we represent a solution by a weight matrix. In the matrix, each row corresponds to a hidden unit and each column corresponds to an input unit, a hidden unit, or an output unit. Figure 3 shows an example of such encoding. The relationships among the edges in the network can be better reflected in this 2D representation, which will be experimentally supported in Section 4. In addition, we suspect that some units have stronger relationships with one another than with the others. We further modify the chromosome by reordering the units according to their relative relationships (See Section 3.3). In summary, a chromosome is represented by a $p \times (p+n+q)$ matrix with columns and rows reordered, where p, n, and q are the numbers of hidden units, input units, and output units, respectively.

3.3 Neuron Reordering (Network Restructuring)

3.3.1 Schemata

A schema is a similarity template describing a subset of strings with similarities at certain string positions inside chromosomes [13]. In a linear encoding of m genes, a schema is defined to be an m-tuple $\langle s_1 s_2 \cdots s_m \rangle$ where $s_i \in U \cup \{*\}$, given a set of alphabet U. In a schema, the symbol '$*$' represents "don't-care" locations and the other symbols represent *specific symbols* which specify the pattern. In the study, a chromo-

```
Calculate d_ij (i, j = 1, 2, ..., p);
Choose the pair (u_m, u_n) (m ≠ n) s.t. d_mn is maximum over all d_ij's;
S = u_m u_n;
U = {u_1, u_2, ..., u_p} − {u_m, u_n};
while (U ≠ φ ) {
    Choose u_l ∈ U s.t. L(u_l, S) is maximum over all L(u_i, S)'s;
    Choose u_r ∈ U s.t. R(u_r, S) is maximum over all R(u_i, S)'s;
    if (L(u_l, S) < R(u_r, S)) {
        S = S · u_r;            // concatenation
        U = U − {u_r};
    } else {
        S = u_l · S;            // concatenation
        U = U − {u_l};
    }
}
```

Figure 5: Reordering algorithm

some is a two-dimensional matrix of gene values. Here, a schema can be represented by a $p \times (p+n+q)$ matrix S where $s_{ij} \in U \cup \{*\}$, $U = [-0.5, 0.5]$. Although a GA handles the chromosomes, it also implicitly handles schemata. From one point of view, the process of a GA is a struggle among schemata. Crossover emergently creates new larger schemata from smaller schemata. To construct a larger schema, corresponding smaller schemata must be preserved through the crossover. In the case of linear chromosomes, schemata with short defining lengths[1] or those with clustered specific-symbol distribution turned out to be advantageous in survival [6, 13]. It is intuitively clear that, in 2D chromosomes too, schemata with clustered specific-symbol distributions would have an advantage. Figure 4 shows two examples of 2D schemata. For convenience, we use '#' to represent the positions of specific symbols. In the left-hand schema, the specific symbols on the column l depart from the specific symbols between columns i and j. If the columns l and k are swapped, the specific symbols are clustered and the schema would be better preserved through crossover. This swap corresponds to a swap between two hidden nodes in the neural network structure. This is done in the process of reordering which is described in the next section.

3.3.2 Reordering

In the hidden layer, a unit inhibits or activates other units. We believe that edges connecting units with strong relationships would have stronger patterns than a random set of edges. If units with a high relative contribution can stay close to one another in the chromosome, schemata related to those units would highly probably survive better through crossover. The larger the weight from unit u_i to u_j, the more strongly unit u_i activates unit u_j. On the other hand, smaller negative weight means stronger inhibition. If the unit u_i strongly activates or inhibits unit u_j, we consider unit u_i to have a high relative contribution to unit u_j.

The reordering algorithm is shown in Figure 5. In the algorithm, the function R, L computes the relative contribution of unit u_i on the string $S = u_a u_{a+1} \cdots u_b$ as follows:

$$\begin{aligned} R(u_i, S) &= \alpha \cdot d_{i,b} + (1-\alpha) \cdot d_{i,b-1} \\ L(u_i, S) &= \alpha \cdot d_{i,a} + (1-\alpha) \cdot d_{i,a+1} \end{aligned}$$

where

$$\begin{aligned} &d_{ij} = z_{ij} + z_{ji}, \\ &z_{ij} = \ \mid (w_{ij} - m_i)/\sigma_i \mid, \\ &w_{ij} : \text{the weight from hidden unit } u_j \text{ to } u_i, \\ &m_i = (\textstyle\sum_{j=1}^{p} w_{ij})/p, \text{ and} \\ &\sigma_i = \sqrt{(\textstyle\sum_{j=1}^{p} (w_{ij} - m_i)^2)/p}\ . \end{aligned}$$

In the expression, d_{ij} is the relative contribution between u_i and u_j. The reordering algorithm helps units with high relative contribution stay close together in the array of neurons. This helps their relevant genes (weights) stay close together in the 2D chromosomes, too. Figure 6 shows an example of such unit reordering. By neuron reordering, schemata are transformed to different ones. The schema in Figure 6 was transformed to a seemingly better one to survive. The effectiveness of the reordering will be examined in Section 4.

[1] The length between the leftmost specific symbol and the rightmost specific symbol.

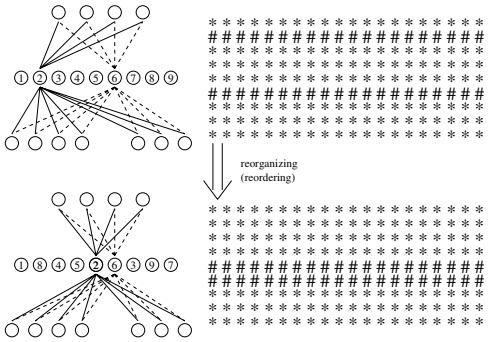


Figure 6: Example of unit reordering

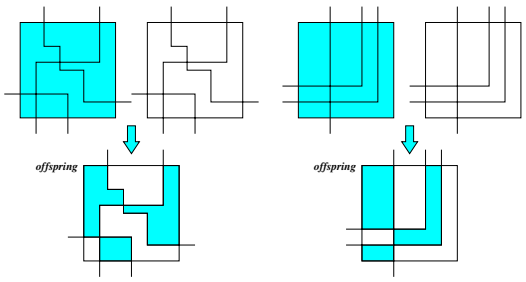


Figure 7: Example of 2D geographic crossover

Bui and Moon [3] proposed a static reordering technique where the genes are reordered just once before the GA starts. The reordering here is different from [3] in that it reorders the neurons every time an offspring is created.

3.4 Geographic 2D Crossover

Two-dimensional encoding can preserve more geographical relationships among the genes [4, 15]. However, when traditional straight-line-based cutting strategies are used, the power of new-schema creation is far below that of crossovers on linear encodings [15]. Geographic crossover was suggested to resolve this problem [15]. In the case of a 2D encoding, it chooses a number of lines, divides the chromosomal domain into two equivalence classes, and alternately copies the genes from the two parent chromosomes. Figure 7 shows two example geographic crossover operators. We used geographic crossover in this work. By combining two-dimensional representation, unit reordering, and 2D geographic crossover, we are pursuing both reduced information loss in the stage of encoding and the power of new-schema creation.

Figure 8: Sample input data

4 Experimental Results

In this section, we examine the performance of the proposed hybrid neuro-genetic algorithm. We used the MNIST database[2] for input patterns of 28×28 grids. We used 800 samples for training, 200 samples for the validation set, and 2000 samples for the test set. The samples are evenly classified to ten digits. Figure 8 shows some samples. The digit images were written by 500 different writers. Simard *et al.* [24] experimented with the same data and used a 3-layer neural network. In their experiment, the recognition rate of the test data was 97.05%. However, they used one input unit for each pixel (i.e., $28 \times 28 = 784$ input units in total) and 650 hidden units. We wish to obtain comparable results to those of [24] using a much simpler model. We used 45 input units and 20 hidden units.[3]

First, we examine the recognition power of the method with HDD of the input images. The weights are tuned by backpropagation only. The initial weights of the network are initialized at random between -0.5 and 0.5. The learning rate is 0.8 and the momentum factor is used. All the other experiments in this paper are also based on this scenario. This method with only HDD input showed 86.00% recognition rate. This recognition rate is consistent with the HDD model of Kohara and Nakamura's (85.3%) [16]. Also, we examine the recognition power of the method with both HDD and VDD information. The recognition power was notably better than the HDD-only model. The addition of VDD information was realized at negligible cost. We use the HDD+VDD model in the hybrid GAs.

Next, we examine the performance of the neuro-genetic hybrids. We tested four versions which are classified according to the utilization of reordering and

[2] http://www.research.att.com/~yann/exdb/mnist

[3] Since we used a recurrent model, the number of edges is greater than the feed-forward model with the same number of units. However, the total number of edges is 1,295, which is much smaller than the 516,100 edges of Simard *et al.*'s study.

Table 1: Results of Neuro-Genetic Hybrid with Given Ordering

	0	1	2	3	4	5	6	7	8	9
	Linear Xover (94.55%)									
0	**192**	2	0	0	0	2	0	0	1	3
1	1	**194**	0	0	0	2	3	0	0	0
2	0	0	**194**	0	0	1	1	3	1	0
3	0	0	2	**185**	0	1	0	0	12	0
4	0	0	0	4	**186**	1	0	1	4	4
5	2	3	0	1	4	**188**	0	0	2	0
6	2	2	0	0	0	4	**192**	0	0	0
7	0	0	0	0	0	2	2	**189**	6	1
8	0	3	1	5	1	0	0	2	**187**	1
9	4	0	0	1	3	2	0	2	4	**184**
	Geographic 2D Xover (95.20%)									
0	**190**	0	0	0	0	3	0	0	1	6
1	0	**195**	0	0	0	2	2	1	0	0
2	1	4	**193**	0	0	0	1	1	0	0
3	0	2	9	**183**	0	0	0	0	6	0
4	0	0	0	3	**186**	0	0	1	1	9
5	4	0	0	0	4	**189**	0	0	1	2
6	1	1	0	0	1	2	**195**	0	0	0
7	0	0	1	0	0	0	3	**195**	0	1
8	4	0	0	3	0	0	0	1	**191**	1
9	4	0	0	1	3	0	0	1	4	**187**

Table 2: Results of Neuro-Genetic Hybrid with Reordering

	0	1	2	3	4	5	6	7	8	9
	Linear Xover (93.85%)									
0	**187**	0	0	0	0	8	2	0	1	2
1	0	**193**	2	0	0	0	1	3	1	0
2	0	4	**192**	1	1	0	0	1	1	0
3	0	0	8	**183**	1	1	0	2	5	0
4	1	0	0	0	**185**	0	0	0	0	14
5	6	2	0	0	1	**188**	3	0	0	0
6	0	3	0	0	1	2	**194**	0	0	0
7	0	1	1	1	0	1	2	**190**	4	0
8	1	0	0	6	1	4	0	3	**183**	2
9	3	2	0	0	7	2	0	2	2	**182**
	Geographic 2D Xover (97.10%)									
0	**193**	0	0	0	0	5	1	0	0	1
1	0	**196**	0	0	0	0	4	0	0	0
2	0	1	**193**	0	1	0	2	2	1	0
3	0	0	2	**196**	1	0	0	0	1	0
4	1	0	0	0	**196**	0	0	0	0	3
5	4	0	0	0	3	**191**	2	0	0	0
6	0	1	0	0	1	1	**196**	0	1	0
7	0	0	1	1	0	0	2	**196**	0	0
8	0	0	1	4	0	1	0	1	**193**	0
9	2	0	0	1	3	0	1	0	1	**192**

2D geographic crossover. The framework of the neuro-genetic hybrids was described in Figure 2. In this framework, the backpropagation adopted the stopped learning method [11, 12].

Table 1 represents the results without reordering. The first group represents the recognition results with the traditional multi-point crossover under the linear encoding. The second group represents the results with the geographic crossover under the 2D encoding. They were both superior to the neural network with only backpropagation. The 2D encoding showed slight improvement over the linear encoding. Table 2 represents the results with reordering. The effectiveness of the 2D encoding and the 2D geographic crossover was notable.

Table 3 shows a summary of the experimental results. The mean and best results are derived from 500 trials for each version. BP1 represents the experimental results with HDD (Kohara – Nakamura version). BP2 indicates the version with HDD + VDD. BP1 and BP2 used only backpropagation. The versions GA1 through GA4 are hybrid GAs with backpropagation and correspond to Table 1 and Table 2. The effectiveness of the geographic 2D crossover was consistent in both non-reordered and reordered cases. The reordering significantly improved the results with the 2D encoding. However, it was not helpful with the linear encoding. This phenomenon appears to occur because the reordering is performed not with the edges but with the neurons (hidden nodes); the order of columns and rows in the 2D encoding are determined by the reordering of neurons. The combination of the 2D crossover and reordering showed strong synergy. Note

Table 3: Summarization of the Result over 500 Trials

	Xover	Reordering	Mean (%)	Best (%)
BP1	—	—	85.35	86.00
BP2	—	—	90.28	93.45
GA1	1D	N	93.38	94.55
GA2	2D	N	94.08	95.20
GA3	1D	Y	93.49	93.85
GA4	2D	Y	96.52	97.10

that the recognition rate 97.10% of GA4 (with 75 units and 1,295 edges in total) is comparable to the value of 97.05% of Simard *et al.*'s study [24] which used a far more complex structure (with 1,444 units and 516,100 edges in total).

5 Concluding Remarks

We proposed a neuro-genetic hybrid approach for handwritten character recognition. To reduce the size of ANNs, we used two vectors representing horizontal density distributions and vertical density distributions for input. This model provides a dimensional reduction of the problem and its actual performance was comparable to the one-unit-per-pixel model.

We devised a reordering algorithm of neurons to effectively reflect the neurons' geographic relationship in the genetic search. We also used a 2D encoding and 2D geographic crossover. The 2D encoding/crossover and reordering showed strong synergy. It is notable that we used an ANN model with 1,295 edges and obtained results that are comparable to those achieved with a huge model with 516,100 edges.

The ideas we have presented here are applicable not just to handwritten character recognition. Future studies will include extending the experiments to other problems.

Acknowledgments

This research was supported in part by KOSEF through Statistical Research Center for Complex Systems at Seoul National University and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

References

[1] C. A. Anderson, K. F. Jones, and J. Ryan. A two-dimensional genetic algorithm for the Ising problem. *Complex Systems*, 5:327–333, 1991.

[2] S. Behnke, M. Pfister, and R. Rojas. Recognition of handwritten digit using structural information. In *International Conference on Neural Networks (ICNN'97)*, volume 3, pages 1391–1396, 1997.

[3] T. N. Bui and B. R. Moon. Hyperplane synthesis for genetic algorithms. In *International Conference on Genetic Algorithms*, pages 102–109, 1993.

[4] T. N. Bui and B. R. Moon. On multi-dimensional encoding/crossover. In *International Conference on Genetic Algorithms*, pages 49–55, 1995.

[5] T. N. Bui and B. R. Moon. Genetic algorithm and graph partitioning. *IEEE Trans. on Computers*, 45(7):841–855, 1996.

[6] T. N. Bui and B. R. Moon. GRCA: A hybrid genetic algorithm for circuit ratio-cut partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(3):193–204, 1998.

[7] J. P. Cohoon and W. Paris. Genetic placement. In *IEEE International Conference on Computer-Aided Design*, pages 422–425, 1986.

[8] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[9] A. Grauel and F. Berk. Mapping of dynamical systems by recurrent neural networks in an evolutionary algroithm approach. In *European Congress on Intelligent Techniques and Soft Computing*, volume 1, pages 470–476, 1998.

[10] S. A. Harp, T. Samad, and A. Guha. Towards the genetic synthesis of neural networks. In *International Conference on Genetic Algorithms*, pages 360–369, 1989.

[11] M. H. Hassoun, P. B. Watta, and R. Shringarpure. Cross-validation without a validation set in bp-trained neural nets. In *IEEE International Conference Neural Networks*, volume 1, pages 369–372, 1995.

[12] S. Haykin. *Neural Networks, A Comprehensive Foundation*. Prentice Hall, 1999.

[13] J. H. Holland. *Adaptation in natural and artificial systems*. The University of Michigan Press, 1975.

[14] R. Jeff and V. B. Ciesielski. An evolutionary approach to training feed-forward and recurrent neural networks. In *International Conference on Knowledge-Based Intelligent Electronics Systems*, pages 596–602, 1998.

[15] A. B. Kahng and B. R. Moon. Toward more powerful recombinations. In *International Conference on Genetic Algorithms*, pages 96–103, 1995.

[16] K. Kohara and Y. Nakamura. Modifying desired outputs to improve pattern recognition by combining subfeature-input neural networks. *Transactions of the Institute of Electrical Engineers of Japan*, 117-c(6):805–813, 1997.

[17] T. Kumagai, M. Wada, S. Mikami, and R. Hashimoto. Structured learning in recurrent neural network using genetic algorithm with internal copy operator. In *IEEE International Intermag '97 Magnetics Conference*, pages 651–656, 1997.

[18] S. J. Lee and H. L. Tsai. Pattern fusion in feature recognition neural networks for handwritten character recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 28(4):612–617, 1998.

[19] S. W. Lee and H. H. Song. A new recurrent neural-network architecture for visual pattern recognition. *IEEE Transactions on Neural Networks*, 8(2):331–340, 1997.

[20] C. T. Lin and C. P. Jou. Controlling chaos by GA-based reinforcement learning neural network. *IEEE Transactions on Neural Networks*, 10(4):846–869, 1999.

[21] B. R. Moon and H. N. Kim. Effective genetic encoding with a two-dimensional embedding

heuristic. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 3(2):113–120, 1999.

[22] B. R. Moon, Y. S. Lee, and C. K. Kim. GEORG: VLSI circuit partitioner with a new genetic algorithm framework. *Journal of Intelligent Manufacturing*, 9(5):401–412, 1998.

[23] V. Patridis, E. Paterakis, and A. Kehagias. A hybrid neural-genetic multimodel parameter estimation algorithm. *IEEE Transactions on Neural Networks*, 9(5):862–876, 1998.

[24] P. Simard, Y. LeCun, J. Denker, and B. Victorri. Transformation invariance in pattern recognition, tangent distance and tangent propagation. In G. Orr and K. Muller, editors, *Neural Networks: Tricks of the Trade*. Springer, 1998.

[25] B. Zhang, M. Fu, H. Yan, and M. Jabri. Handwritten digit recognition by adaptive-subspace self-organizing map (ASSOM). *IEEE Transactions on Neural Networks*, 10(4):939–945, 1999.