New Usage of SOM for Genetic Algorithms

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. Self-Organizing Map (SOM) is an unsupervised learning neural network and it is used for preserving the structural relationships in the data without prior knowledge. SOM has been applied in the study of complex problems such as vector quantizations, combinatorial optimization, and pattern recognition. This paper proposes a new usage of SOM as a tool for schema transformation hoping to achieve more efficient genetic process. Every offspring is transformed into an isomorphic neural network with more desirable shape for genetic search. This helps genes with strong epistasis to stay close together in the chromosome. Experimental results showed considerable improvement over previous results.

1 Introduction

There have been a great many genetic algorithms (GAs) for neural network (NN) optimization [9][11][18]. In order to represent solutions for NN optimization, most GAs used linear encodings following the convention of the GA community [11][15]. Mostly, every weight in an NN takes a position in a linear chromosome. However, linear encodings have limited capability in reflecting the geographic linkages of genes [1][7]. A number of 2D encodings were used to better reflect the geographic linkages of genes [1][4][7][12].

In this paper, we use a two-dimensional encoding for the genetic representation and employ 2D *geographic crossover* which demonstrated good performance for the neural network optimization problem [13] and the graph partitioning problem [12][16][17]. The weights of an NN can be represented by a 2D matrix and thus they are intrinsically suitable for 2D encoding.

A typical NN consists of input, output, and hidden layers. The neurons in the hidden layer enable the network to learn complex tasks by extracting progressively more meaningful features from the input patterns and to form the rules classifying the input patterns. Once an NN is represented by a two-dimensional encoding and the structure of the NN is fixed, the genotype of an NN depends on the order of neurons in the structure. All the hidden neurons are identical when only the connections are considered. They become different only after they start having weights in the connections. Some hidden neurons have stronger relation than between ordinary pairs of hidden neurons. However, the relationship among the hidden neurons is not revealed from neurons' placements. Since the

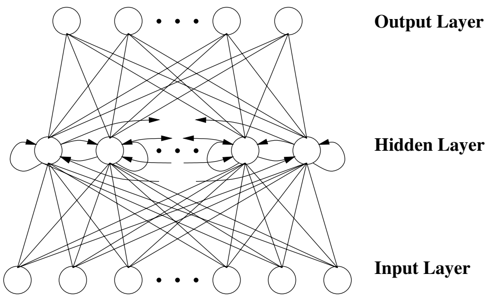


Fig. 1. The recurrent neural network architecture

indices of hidden nodes are assigned before they start having weights, there is no guarantee that strongly related neurons are assigned close indices to one another. This causes high-quality schemata not to survive well. In this paper, we transform each neural network to an isomorphic neural network with different indices of neurons and we aim to develop a better genetic algorithm by achieving a better representation of neural networks. We use self-organizing map (SOM) for the transformation.

Self-Organizing Map (SOM) is an unsupervised learning neural network and it is used for preserving the structural relationships in data without prior knowledge. SOM has been applied in the study of complex problems such as vector quantizations, combinatorial optimization, and pattern recognition. This paper proposes a new usage of SOM as a tool for schema transformation hoping to achieve more efficient genetic process.

The rest of this paper is organized as follows. In the next section, we describe the necessity of clustering strongly connected neurons and the property of SOM. In Section 3, we describe the transformation that enables useful building blocks to survive well. Section 4 shows our hybrid neuro-genetic framework for improving the performance of the neural network. We present our experimental results in Section 5 and state our conclusions in Section 6.

2 Preliminaries

2.1 Network Architecture

We use a recurrent neural network (RNN) 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 at every Δt time interval. The network is usually trained by a backpropagation-based algorithm.

2.2 Topological Ordering Property of SOM

SOM is an effective software tool for the visualization of high-dimensional data. It converts complex, nonlinear statistical relationships among high-dimensional data items into simple geometric relationships on a low-dimensional display [14]. As it thereby compresses information while preserving the most important topological and metric relationships of the primary data items on the display, it may also be thought to produce some kind of abstractions. These two aspects, visualization and abstraction, can be utilized in a number of ways in complex tasks such as process analysis, machine perception, control, and communication. The data items are organized into a meaningful two-dimensional order in which similar items are close to one another. In this sense, SOM is a similarity graph and also a clustering diagram [14].

2.3 Gene Reordering

Each gene location has an explicit meaning in most genetic algorithms. This type encoding is called locus-based encoding. Many researchers have studied the gene reordering. Bagley [2] tried to change the gene ordering at random. Bui and Moon [3][6] tried to exploit the useful clustering information of graphs and showed that the performance of a GA can be improved by reordering genes in the graph bipartitioning problem with a linear encoding. Kim and Moon [13] proposed a reordering heuristic for neural-network representation. They believed that edges connecting units with strong relationships would have stronger patterns than a random set of edges and used the relative strengths of the connection weights to measure the relationships among genes. This approach showed considerable improvement of GA performance.

3 Transforming into Isomorphic Structures

Since the hidden neurons of the network are fully connected to one another, there are great many hidden node orderings with equivalent function. All these structures represent the same network but have different genotypes. Some of them are more advantageous than the others in genetic search. Figure 2 shows an example with matrix encoding. In the figure, the three neurons j, i, and $i+1$ are assumed to have strong relationships. The positions of corresponding weights are marked with rectangles. The set of rectangles constructs a schema of perhaps high quality. Although the geographic crossover has the new-schema creation power for 2D encoding, if useful features on the encoding are not clustered (see the left weight matrix in Figure 2), corresponding schemata are not easy to survive in most crossovers. It is important that the given structure must be transformed to the functionally equivalent structure containing the useful clustered blocks. When we exchange the positions of neurons j and $i-1$ in the left weight matrix, the schema is transformed to another one with better shape for survival (See the right weight matrix in Figure 2). We aim to transform high

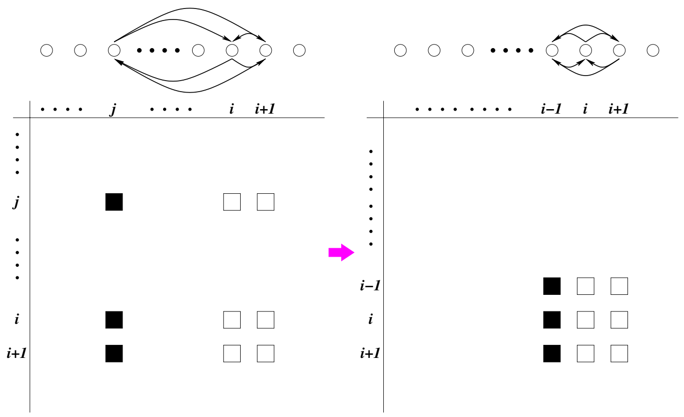


Fig. 2. Change of schema after transformation

quality schemata into better shapes with equivalent function. The more clusters of the high correlated neurons, the finer approximation of the complex system to be modeled can be eventually obtained. The idea takes advantage of the ability of SOM for clustering neurons.

3.1 SOM-Based Transformation

Each hidden neuron has a vector of weights and affects other neurons according to the weights. A large weight corresponds to a strong activation of another neuron. On the other hand, a significantly small negative weight corresponds to strong inhibition. If a neuron strongly activates or inhibits another neuron, we consider that these neurons are highly correlated. This information is utilized in the transformation. However, it is difficult to analyze the relative relationship among neurons and the spatial ordering of the corresponding neurons. In this work, we transform the topology of neural networks using SOM. This approach is motivated by a desire to better preserve the relationships of hidden neurons in the genetic process.

We describe the details of the SOM-based transformation in the following:

Input : an RNN with weights between neurons.
Output : a one-dimensional order of hidden neurons

$$\Pi = \langle \pi_1, \cdots, \pi_p \rangle, \qquad \pi_i \in \{1, 2, \cdots, p\}.$$

1. The hidden layer is mapped to the competitive layer of SOM. That is, the number of hidden neurons is the same as the number of competitive neurons in SOM. The synaptic weights of the RNN are used as the input vectors for the SOM. Let the number of input neurons, hidden neurons, and output neurons of RNN be n, p, and q, respectively. Then the SOM has the $(n+p+q)$ input neurons and the p competitive neurons.

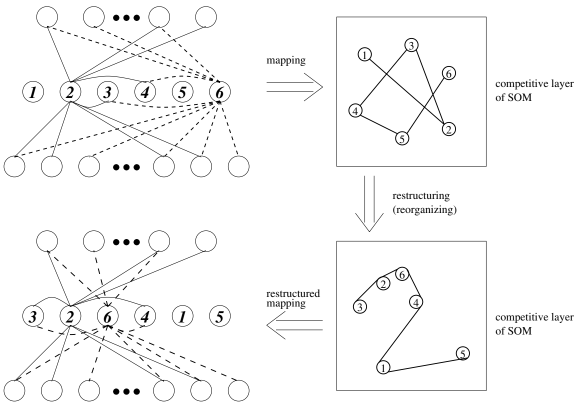


Fig. 3. Mapping between RNN and SOM

2. SOM is trained with the synaptic weights related to the hidden neurons of the RNN. SOM produces a similarity graph reflecting the relationship among these synaptic weights. The number of input data pairs for SOM is p. That is, SOM trains the data pairs $\langle \mathbf{x}_j, d_j \rangle$ where $\mathbf{x}_j$ is an input vector for the SOM and d_j is the label corresponding to $\mathbf{x}_j$ as follows:

$$\mathbf{x}_j = \mathbf{w}_j, \qquad d_j = i \qquad (1 \le i \le p)$$
$$\mathbf{w}_j = \langle w_{j1}^{(0)}, \cdots, w_{jn}^{(0)}, w_{j1}^{(1)}, \cdots, w_{jp}^{(1)}, w_{j1}^{(2)}, \cdots, w_{jq}^{(2)} \rangle$$

   where

   $w_{jk}^{(0)}$: weight from input unit k to the hidden unit j of the RNN
   $w_{jk}^{(1)}$: weight from hidden unit k to the hidden unit j of the RNN
   $w_{jk}^{(2)}$: weight from hidden unit j to the output unit k of the RNN

3. The SOM produces a linear order of the competitive neurons. The hidden neurons are rearranged according to the order produced by SOM. Sometimes several competitive neurons may have the same index. It occurs when the synaptic weights for different hidden neurons without being transformed represent a similar distribution. In this case, we assign the same index to multiple competitive neurons. This attempts to make the neural network obtain the effect of the weight-sharing [19].

Figure 3 shows an example SOM-based transformation. In the figure, the solid and dashed lines indicate the weights of high magnitudes. SOM trains the synaptic weights of the given network (the left-top neural network in Figure 3) and produces an ordering $\Pi = \langle 3, 2, 6, 4, 1, 5 \rangle$. The location of each node on the competitive layer of SOM reflects the relative relationship among hidden neurons. The high correlated neurons 2 and 6 locate distant to each other before the transformation; they are clustered after the transformation.

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

Fig. 4. The template of the hybrid GA

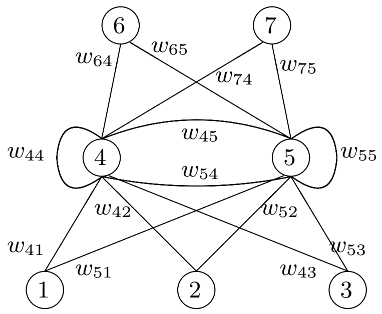
4 GA for Optimizing the Neural Network

4.1 The GA Structure

The template of the proposed genetic algorithm is shown in Figure 4. It is a typical steady-state hybrid genetic algorithm except for the SOM-based transformation. In the following, we describe the genetic framework.

- *Initialization*: The initial population is generated at random. We set the population size to be 50.
- *Selection*: In each iteration, two parents are selected according to probabilities that are proportional to their fitness values. The probability that the best solution is chosen was given four times that of the worst solution. This is a typical normalized roulette-wheel proportional selection.
- *Crossover*: The offspring is produced through geographic 2D crossover. The "g2d_xover" of Figure 4 indicates the geographic 2D crossover. The geographic 2D crossover is described in Section 4.3.
- *Mutation*: With a low probability, this offspring is then modified by a *mutation* operator as follows: $\omega_i = \omega_i + \lambda_{0.5}$ where ω_i is the i^{th} gene value and λ_γ returns a random value between -γ and γ .
- *Local Optimization*: After an *offspring* is modified by a mutation operator, it is 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. After the backpropagation, the neural network has a mean-square error and a discrimination ratio on the training data. The fitness of the offspring is determined with these two values[1].
- *Transformation*: After local optimization, the genes in the offspring are reordered. This causes a structural change of the corresponding neural network

[1] $f = \varepsilon + \eta(100 - \mu)$,
where ε is the mean-square error, μ is the discrimination ratio, and η is a constant.



	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}

Fig. 5. Example of 2D encoding

without affecting the quality. The reorganizing approach was described in Section 3.

– *Replacement*: Then the offspring replaces a solution in the population by the following rule: 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 [5].
– *Stopping Condition*: For stopping, we use a fixed number of generations.

4.2 Problem Encoding

Most GAs for neural network optimization encode a solution (a set of weights) with a linear string following the convention [9][11][15]. 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 5 shows an example of such encoding. In this figure, the number within a circle (neuron) indicates the index of the neuron and w_{ij} represents the weight of the edge from neuron j to neuron i. The relationships among the edges in the network can be better reflected in this 2D representation, which was experimentally supported in [13]. In addition, we suspect that some neurons have stronger relationships with one another than with the others. We further modify the encoding by transforming the given network into an isomorphic network (See Section 3). In summary, a chromosome is represented by a $p \times (p + n + q)$ matrix with columns and rows rearranged, where p, n, and q are the numbers of hidden units, input units, and output units, respectively.

4.3 Geographic 2D Crossover

Two-dimensional encoding can preserve more geographical relationships among the genes [4][12][13]. However, when traditional straight-line-based cutting strategies are used, the power of new-schema creation is far below that of

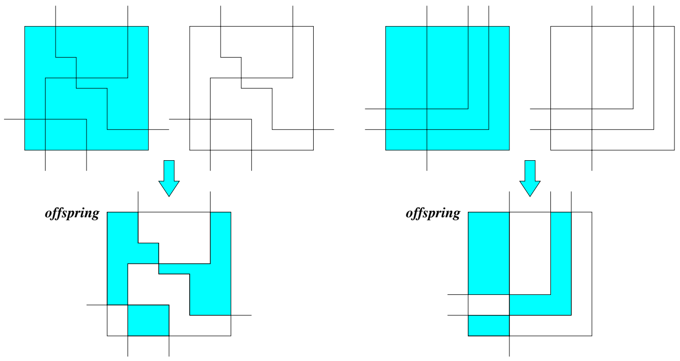


Fig. 6. Example of 2D geographic crossover

crossovers on linear encodings [12]. Geographic crossover was suggested to resolve this problem [12]. In the case of a 2D encoding, it chooses a number of monotonic lines, divides the chromosomal domain into two equivalence classes, and alternately copies the genes from the two parent chromosomes. Figure 6 shows two example geographic crossover operators. We used geographic crossover in this work. By combining two-dimensional representation, isomorphic transformation, and 2D geographic crossover, we are pursuing both reduced information loss in the stage of encoding and the power of new-schema creation.

5 Experimental Results

5.1 Database

We selected three well known datasets to measure the performance of the proposed approach. First, we used the MNIST handwritten character recognition database[2] for input patterns of 28×28 grids. The samples are evenly classified to ten digits. The digit images were written by 500 different writers. Recognition of handwritten character is an important task in the automated document analysis. This application is used to read postal addresses and bank checks, etc. We used 800 samples for training, 200 samples for the validation set, and 2000 samples for the test set. We used 45 input units and 20 hidden units by using a vertical density distribution and a horizontal density distribution as in [13].

Next, we experiment with Wisconsin breast cancer data (WBCD) and Cleveland heart disease data (CHDD) obtained from the UCI repository[3]. The WBCD contains 699 instances with 458 benign and 241 malignant cases. Each instance is described by a case number, nine attributes with integer values in the range [1, 10] and a binary class label. In the experiment using WBCD database, we used nine input units, nine hidden units, and two output units in the neural

[2] http://yann.lecun.com/exdb/mnist

[3] http://www.ics.uci.edu/˜mlearn/MLRepository.html

Table 1. Result for three testbeds

	BP	GA		
		Original	Heuristic	SOM
MNIST	90.28 (93.45)	94.08 (95.20)	96.52 (97.10)	96.85 (99.60)
WBCD	95.88 (96.37)	96.87 (97.21)	96.91 (97.43)	97.19 (97.79)
CHDD	80.56 (83.25)	82.22 (84.26)	82.75 (86.84)	82.82 (87.02)

network. We divided the data into 80 training samples, 20 validation samples, and 599 test samples. The CHDD contains 303 instances with 164 healthy instances. The rest are heart disease instances of various degrees of severity (4 classes). Each instance is described with 13 attributes. We used 80 samples for the training set, 20 samples for the validation set, and 203 samples for the test set. We used 13-21-4 network structure for CHDD data.

In these experiment, for robust check for the effect of the proposed strategy, we experimented with the data following the 5-fold *hold out method* cross-validation approach [10]. In such a situation, we divide the available set of data into 5 subsets. The model is trained on all the subsets except for one, and the validation error is measured by testing it on the subset left out. This procedure is repeated for a total of 5 trials, each time using a different subset for validation.

5.2 Experimental Results

Now the general effect of the proposed SOM-based transformation is illustrated with the above databases. The weights of a neural network were initialized at random between −0.5 and 0.5. The learning rate was set to 0.8 and the momentum factor was used. The weights were tuned locally by backpropagation.

Table 1 shows the classification result. In the table, “BP” indicates the neural networks that are trained by backpropagation only. “GA” represents the hybrid neuro-genetic algorithm and is classified into three types. “Original” represents the version without any transformation in the genetic process. The other cases contain transformation in the genetic framework. “Heuristic” indicates the version with a heuristic transformation by [13]. “SOM” represents the version with SOM-based transformation. The two values in each experiment show the mean and the best recognition result, respectively, from 500 trials. The hybrid approach showed improvement over the non-hybrid approach (BP) and the transformation approaches turned out to be useful. The SOM-based transformation showed considerable improvement over the heuristic-based transformation.

6 Concluding Remarks

We devised a structural transformation algorithm for neuro-genetic hybrids to effectively reflect geographic correlations or relative contributions among neurons in the genetic search. The topological ordering property of SOM was used to cluster neurons with high correlations. The proposed SOM-based transformation dynamically alters the geographical shapes of fitness landscapes; consequently, the shapes of schemata are altered as well.

To the best of our knowledge, this is the first usage of SOM in the context of genetic search. We believe that there is room for further improvement in this direction. Candidates of future studies include the enhancement of SOM-based transformation and the design of other types of transformation.

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. J. Bagley. *The Behavior of Adaptive Systems Which Employ Genetic and Correlation Algorithms*. PhD thesis, University of Michigan, Ann Arbor, MI, 1967.
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. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.
11. 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.
12. A. B. Kahng and B. R. Moon. Toward more powerful recombinations. In *International Conference on Genetic Algorithms*, pages 96–103, 1995.
13. J. H. Kim and B. R. Moon. Neuron reordering for better neuro-genetic hybrids. In *Genetic and Evolutionary Computation Conference*, pages 407–414, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
14. T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574–585, 2000.

15. 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.
16. 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.
17. 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.
18. 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.
19. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McCleland, editors, *Parallel Distributed Processing*, volume 1, chapter 8. MIT Press, Cambridge, MA, 1986.