Structural Learning of Neural Networks by Differential Evolution with Degeneration using Mappings

Tetsuyuki Takahama, Member, IEEE, Setsuko Sakai, Member, IEEE, Akira Hara, Member, IEEE and Noriyuki Iwane, Member, IEEE

Abstract—Structural learning, in which the structure of estimation systems are optimized, has been actively studied in researches on supervised learning of neural networks and fuzzy rules. The GA° (Genetic Algorithm with Degeneration) is one of the structural learning methods, which is modeled on genetic damage and degeneration. In GA°, a gene is defined by a pair of a normal value and a damaged rate that shows how much the gene is damaged. Simple one-point crossover and Gaussian mutation are adopted to deal with the pair. It was very difficult to incorporate newly proposed genetic operations such as blend crossover in GA or operations in differential evolution (DE). In this study, we propose a new idea to incorporate such operations by unifying the values according to a mapping, applying operations and separating the values according to the inverse mapping. This idea is applied to differential evolution, which is known to be an efficient and robust algorithm and DE° (Differential Evolution with Degeneration) is proposed. To show the advantage of DE°, it is applied to the structural learning of a simple test function and neural networks. It is shown that DE° is more robust to high degeneration pressure and can find better estimation models faster, which have less number of parameters and less estimation errors, than GA°.

I. INTRODUCTION

There are many researches on supervised learning using neural networks and they are applied in various fields. However, there are some difficulties in supervised learning using neural networks as follows: (1) It is difficult to select a proper network structure. If the network is too big, the generalization ability becomes poor. If the network is too small, the learning ability becomes insufficient. In many cases, the information about the proper network structure isn’t available. Thus, it is necessary to search the network structure in trial and error. (2) The interpretation of the hidden units is difficult. Generally, sufficient number of hidden units is prepared in order to keep the estimation error small enough. The learned knowledge is distributed to the multiple hidden units. The meanings of each unit become unclear. The interpretation of the learned knowledge becomes difficult. (3) The local minimum problem is inevitable. Because the descent method is often adopted as learning algorithm, the problem becomes serious with the increase of the network size.

In order to solve these problems, it is necessary to optimize the network structure by deleting unnecessary connections and thresholds, in addition to the optimization of the values of the connection weights and the thresholds. The researches on the structural learning of neural networks can be classified into 4 categories; selective methods [1], [2] which use information criteria [3]–[5] to select the best structure, destructive methods [6]–[8], constructive methods [9]–[11], and reducing methods [12]. In destructive/constructive methods, the process of learning parameter values such as weights and the process of deleting/adding parameter structures such as units or connections are repeated as separate processes. Changing the network structure step by step can be considered as a hill-climbing search in the space of network structures. Thus, it is difficult to search an optimal network structure. Also, usually much computation is needed for re-learning the weights, because there is rather large gap between before and after changing the structures. In reducing methods, the process of deleting connections is included in the process of learning weights. There is little gap between before and after reduction, because the values of reduced parameters are nearly zero just before they are reduced. The re-learning is not needed.

The new reducing methods of the structural learning have been proposed based on the idea of degeneration, the Genetic Algorithm with Mutant Genes (MGGA) [13], [14] that employs binary-coded GAs, the Genetic Algorithm with Damaged Genes (DGGA) [15], [16] that employs real-coded GAs and the Genetic Algorithm with Degeneration (GA°) [17]–[21] that is an extension of DGGA. The algorithms reduce or inactivate the genes that less contribute to the survival of the individuals using genetic damage, realizes degeneration and reduces unnecessary parameters in estimation systems. The problems (1) and (2) are solved by using these algorithms, because the algorithms have the ability of optimizing the parameter structure of the systems to be learned. It is expected that problem (3) can be solved, because the algorithms employ evolutionary algorithms that are comparatively difficult to fall into the local minimum problem. It was shown that the algorithms could be applied to the structural learning of polynomial models, neural networks and RBF-fuzzy rule bases.

In GA°, a gene is defined by a pair of a normal value and a damaged rate. The normal value shows the value of gene when the gene is not damaged or is in normal state. The damaged rate shows how much the gene is damaged. Simple one-point crossover and Gaussian mutation are adopted to deal with the pair. It was very difficult to incorporate newly proposed genetic operations such as blend crossover (BLX-α) in GA or operations in differential evolution (DE).
In this study, we propose a new idea to incorporate such operations by unifying the normal values and the damaged rates according to a mapping, applying operations and separating the values according to the inverse mapping. This idea is applied to differential evolution, which is known to be an efficient and robust algorithm, and DE\textsuperscript{d} (Differential Evolution with Degeneration) is proposed. To show the advantage of DE\textsuperscript{d}, it is applied to the structural learning of a simple test function and neural networks. It is shown that DE\textsuperscript{d} is more robust to high degeneration pressure and can find better estimation models faster, which have less number of parameters and less estimation errors, than GA\textsuperscript{d}.

The rest of this paper is organized as follows: Section II describes genetic algorithms with degeneration (GA\textsuperscript{d}). Section III provides the idea of mapping, operation and inverse mapping. Section IV describes differential evolution with degeneration (DE\textsuperscript{d}). Section V presents experimental results of structural optimization of a simple test function. Section VI presents experimental results of structural learning of neural networks. Finally, Section VII concludes with a brief summary of the paper and some remarks.

II. GENETIC ALGORITHMS WITH DEGENERATION

A. Degeneration

In the nature, the phenomenon “degeneration” is well known. Degeneration is the phenomenon that unnecessary organs are lost in the process of evolution. It is assumed that degeneration occurs by the damaged genes that are different from the normal genes: When the genes related to the unnecessary organs are damaged, the individual who has the genes can survive and produce the descendants. The damaged genes are inherited by the descendants. The organ of the descendants, which is related to the damaged genes, will weaken or even disappear. From the result, degeneration occurs. If a parameter is treated as the organ, the unnecessary parameters can be reduced and the number of parameters can be optimized in the system described by many parameters.

B. damaged genes

A damaged gene is a gene of which state is changed by mutation and of which state is different from that of a normal gene. There are various types of mutation such as substitution, insertion, deletion and so on. It is difficult to assume all states that the damaged genes may have. In GA\textsuperscript{d}, a gene is represented by its normal value and damaged rate. The normal value shows the property or type of gene when the gene is in normal state. The damaged rate shows how much the gene is damaged. The damaged rate takes a value in the interval [0, 1]. The damaged rate of a normal gene is 0. The damaged rate of the gene of which character is completely lost is 1.

In GAs, an individual is usually represented by a chromosome, which holds genetic information. The chromosome is represented by an array of genes. Let the array of genes be denoted by $G = g_1, g_2, \ldots, g_L$ ($L$ is the chromosome length), the mapping function from genotype to phenotype by $h$, and the fitness function by $f$. Then the fitness value of the individual is given by $f(h(G))$.

GA\textsuperscript{d} is the algorithm in which damaged genes are introduced and the character of each individual or phenotype is affected by damaged genes. An individual in GA\textsuperscript{d} holds the following information.

- Array of genes($G^d$)
  $G^d = (g_1, d_1, g_2, d_2, \cdots, g_L, d_L)$. $g_i$ is the normal value of $i$-th gene and $d_i$ is its damaged rate. In GA\textsuperscript{d}, it is assumed that a gene is damaged, or the damaged rate increases, with some probability. Also it is assumed that a gene is repaired, or the damaged rate decreases, with some probability.

- Fitness value
  In GA\textsuperscript{d}, the mapping function from genotype to phenotype $h^d$ depends on an array of damaged rates $D = d_1, d_2, \cdots, d_L$ as well as $G$. So, the fitness value of an individual in GA\textsuperscript{d} is given by $f(h^d(G, D))$. Usually, the following mapping is used:

$$h^d(G, D) = g_1 \cdot (1 - d_1) \cdot \cdots \cdot g_L \cdot (1 - d_L) \quad (1)$$

In this mapping, when the gene is completely damaged or $d$ is 1, the phenotype becomes 0. When the gene is in normal state or $d$ is 0, the phenotype is same as the normal value.

Fig.1 shows an example of an individual in GA\textsuperscript{d}.

![Fig. 1. An individual in GA\textsuperscript{d}](image)

C. Algorithm of GA\textsuperscript{d}

The algorithm of GA\textsuperscript{d} is as follows:

0) Initialization: Initial population is created randomly. Usually each damaged rate of an initial individual is generated as a random number in the interval [0, 1]. However all damaged rates can be set all 0 or all 1.

1) Selection: The parents are selected from the population. In structural learning, the evaluation function for a structure will often be minimized. In GA\textsuperscript{d}, a (linear) ranking strategy [22], which is known as a robust strategy, is used to select the individuals. All individuals are ranked according to their fitness values. Let the rank of the $i$-th individual be denoted by $r_i (r_i = 1, 2, \cdots, N)$. The $i$-th individual’s selection probability $s_i$ is defined as follows:

$$s_i = \frac{1}{N} \left( \eta_{max} - (\eta_{max} - \eta_{min}) \frac{r_i - 1}{N - 1} \right) \quad (2)$$

2007 IEEE Congress on Evolutionary Computation (CEC 2007) 3435
In this study, to incorporate such operations we propose a new genetic operations such as operations in DE. In GA, the normal value and the damaged rate are inherited as a pair by the children. If parents are not mated, they will remain in the next generation. Fig. 2 shows an example of the crossover operation, where the normal values and damaged rates are 0 or 1 for simplicity.

<table>
<thead>
<tr>
<th>parent1</th>
<th>child1</th>
</tr>
</thead>
<tbody>
<tr>
<td>D 0110 10001</td>
<td>D 0110 00110</td>
</tr>
<tr>
<td>G 0101 01010</td>
<td>G 0101 01101</td>
</tr>
<tr>
<td>D 1001 000110</td>
<td>D 1001 10001</td>
</tr>
<tr>
<td>G 1100 01101</td>
<td>G 1100 00110</td>
</tr>
</tbody>
</table>

Fig. 2. Crossover operation in GA

3) Reversible mutation: A normal value is mutated with the reversible mutation rate \( P_{rm} \). In this research, a normal value \( g_i \) is changed into \( g_{i}^{new} \) according to the following equation:

\[
g_{i}^{new} = g_{i} + \Delta g
\]

where \( \Delta g \) is a random variable.

4) Biased mutation: A damaged rate is mutated with the biased mutation rate \( P_{bm} \). The probability of increasing the damaged rate is given by a mapping from a current damaged rate \( d \in [0,1] \) to a probability \( p \in [0,1] \), or a damaging probability function \( P_{dam} : d \in [0,1] \rightarrow p \in [0,1] \)

\[
P_{dam} : d \in [0,1] \rightarrow p \in [0,1]
\]

When a biased mutation occurs, the damaged rate \( d_i \) is changed into \( d_{i}^{new} \) according to the following equation:

\[
d_{i}^{new} = \begin{cases} d_{i} + \Delta d & \text{w.p. } P_{dam}(d_i) \\ d_{i} - \Delta d & \text{w.p. } 1 - P_{dam}(d_i) \end{cases}
\]

where \( \Delta d \) is a random variable.

5) Change of generation: Current population is replaced by the children. Go back to 1.

III. MAPPING, OPERATIONS AND INVERSE MAPPING

In GA, the normal value and the damaged rate should be treated as a pair. Thus, it is difficult to incorporate newly proposed genetic operations such as operations in DE. In this study, to incorporate such operations we propose a new and simple idea that pairs are mapped to phenotype values by unifying each normal value and damaged rate, genetic operations are applied to the unified values or phenotype values, and the values are inversely mapped to the normal values using the corresponding damaged rates.

Fig. 3 shows the outline of this idea. Let assume two parents are defined as follows:

\[
G_{1}^{d} = (g_{11}, d_{11}) \cdots (g_{1L}, d_{1L}) \\
G_{2}^{d} = (g_{21}, d_{21}) \cdots (g_{2L}, d_{2L})
\]

1) Mapping: Before operations are applied, two parents are mapped to phenotype by unifying the normal value and the damaged rate. The mapping function for each gene is

\[
h_{i}^{g}(g_{i}, d_{i}) = g_{i}(1 - d_{i})
\]

\[
H_{i} = g_{i1}(1 - d_{i1}) \cdots g_{iL}(1 - d_{iL}), i = 1, 2.
\]

2) Operation: Let assume that some genetic operation such as arithmetic crossover is applied to \( H_{1} \) and \( H_{2} \), and \( H_{1}' \) and \( H_{2}' \) are generated.

\[
H_{i}' = h_{i1}' \cdots h_{iL}', i = 1, 2.
\]

Also, let assume that damaged rates \( D_{1} \) and \( D_{2} \) are recomposed by some crossover operation such as one-point crossover or uniform crossover, and \( D_{1}'' \) and \( D_{2}'' \) are generated.

\[
D_{i}' = d_{i1}' \cdots d_{iL}', i = 1, 2.
\]

3) Inverse mapping: The normal value is obtained from \( H_{1}' \) by inverse mapping using \( D_{1}' \), and new individual \( G_{1}''^{d} \) is generated. The inverse mapping function for each gene is

\[
h_{i}^{-1}(h_{ij}, d_{ij}) = h_{ij}/(1 - d_{ij})
\]

\[
G_{1}''^{d} = (g_{11}', d_{11}') \cdots (g_{1L}', d_{1L}') \cdots (g_{21}', d_{21}') \cdots (g_{2L}', d_{2L}')
\]

\[
g_{ij}' = \begin{cases} h_{ij}/(1 - d_{ij}) & d_{ij} \neq 1 \\ 0 & d_{ij} = 1 \end{cases}
\]

When a gene is completely damaged or the damaged rate is 1, the normal value becomes zero to avoid division by zero.

The consecutive operations of mapping, operations, and inverse mapping correspond to a usual crossover operation in GA. After the operations are applied, mutation operation will be applied. Reversible mutation will be applied to the normal values if needed. Biased mutation will be applied to the damaged rates.

IV. DIFFERENTIAL EVOLUTION WITH DEGENERATION

A. Differential evolution

Differential evolution is a variant of ES proposed by Storn and Price [24], [25]. DE is a stochastic direct search method using population or multiple search points. DE has been successfully applied to the optimization problems including non-linear, non-differentiable, non-convex and multi-modal functions. It has been shown that DE is fast and robust to these functions.
There are some variants of DE that have been proposed, such as DE/best/1/bin and DE/rand/1/exp. The variants are classified using the notation DE/base/num/cross. “base” indicates the method of selecting a parent that will form the base vector. For example, DE/rand/num/cross selects the parent for the base vector at random from the population. DE/best/num/cross selects the best individual in the population. “num” indicates the number of difference vectors used to perturb the base vector. “cross” indicates the crossover mechanism used to create a child. For example, DE/base/num/bin shows that crossover is controlled by binomial crossover using constant crossover rate. DE/base/num/exp shows that crossover is controlled by a binomial crossover using exponentially decreasing the crossover rate.

In DE, initial individuals are randomly generated within the search space and form an initial population. Each individual contains \(n\) genes as decision variables or a decision vector. At each generation or iteration, all individuals are selected as parents. Each parent is processed as follows: The mutation process begins by choosing \(1 + 2\) num individuals from the parents except for the parent in the processing. The first individual is a base vector. All subsequent individuals are paired to create \(num\) difference vectors. The difference vectors are scaled by the scaling factor \(F\) and added to the base vector. The resulting vector is then recombined or crossed-over with the parent. The probability of recombination at an element is controlled by the crossover rate \(CR\). This crossover process produces a trial vector. Finally, for survivor selection, the trial vector is accepted for the next generation if the trial vector is better than the parent.

In this study, DE/rand/1/exp variant, where the number of difference vector is 1 or \(num = 1\), is used.

The algorithm of DE/rand/1/exp variant, which is used in this study, is as follows:

0) Initialization: Initial \(N\) individuals \(x^i\) are generated as the initial search points, where there is an initial population \(P(0) = \{x^i, i = 1, 2, \ldots, N\}\).
1) Termination condition: If the number of generations (iterations) exceeds the maximum generation \(T_{\text{max}}\), the algorithm is terminated.
2) Mutation: For each individual \(x^i\), three individuals \(x^{i1}\), \(x^{i2}\) and \(x^{i3}\) are chosen from the population without overlapping \(x^i\) and each other. A new vector \(x'\) is generated by the base vector \(x^{i1}\) and the difference vector \(x^{i2} - x^{i3}\) as follows:
\[
x' = x^{i1} + F(x^{i2} - x^{i3})
\]
where \(F\) is a scaling factor.
3) Crossover: The vector \(x'\) is crossed-over with the parent \(x^i\). A crossover point \(j\) is chosen randomly from all dimensions \([1, n]\). The element at the \(j\)-th dimension of the trial vector \(x^{\text{new}}\) is inherited from the \(j\)-th element of the vector \(x'\). The elements of subsequent dimensions are inherited from \(x'\) with exponentially decreasing probability defined by a crossover factor \(CR\). Otherwise, the elements are inherited from the parent \(x^i\). In real processing, mutation and crossover are integrated as one operation.
4) Survivor selection: The trial vector \(x^{\text{new}}\) is accepted for the next generation if the trial vector is better than the parent \(x^i\).
5) Go back to 1.

**B. Mapping, operations and inverse mapping in differential evolution**

As described above, four individuals are used in DE/rand/1/exp. In DE, these individuals are mapped to phenotype values, the values are recombed by mutation and crossover operation, and the generated values are inversely
mapped to the normal values. The algorithm of DE\(^d\) can be described as follows:

1) Termination condition: If the number of generations (iterations) exceeds the maximum generation \(T_{max}\), the algorithm is terminated.

2) Mapping: For each individual \(G_i\), a base individual \(G_i^p\), and two individuals \(G_{p_2}\) and \(G_{p_3}\) for a difference vector are chosen from the population without overlapping \(G_i\) and each other. All individuals are converted to phenotype values by the mapping for each gene \(h^d(g_j, d_j) = g_j(1 - d_j)\), and \(H_i\) and \(H_{p_k}\), \(k = 1, 2, 3\) are obtained.

3) Operation: A new vector \(H'\) is generated by the base vector \(H_{p_1}\) and the difference vector \(H_{p_2} - H_{p_3}\) as follows:

\[
H' = H_{p_1} + F(H_{p_2} - H_{p_3})
\]

The vector \(H'\) is recomposed with the parent vector \(H_i\). Also, the damaged rate \(D_i\) is recomposed with the damaged rate of base individual \(D_{p_1}\) and \(D'_{i}\) is obtained.

4) Inverse mapping: The normal values of the parent \(G_i'\) is obtained by the inverse mapping for each gene \(h^{d^{-1}}(h_j, d_j) = h_j/(1 - d_j)\) and a new individual \(G_i'^{d} = (G_i', D_{i}')\) is generated.

5) Survivor selection. The trial vector \(H' = h^d(G_i', D_{i}')\) is accepted for the next generation if the trial vector is better than the parent vector \(H_i\).

6) Go back to 1.

Fig. 4 shows sample pseudo-code of DE\(^d/rand/1/exp\). Generally in structural learning, the objective value is estimation error for training data and the value is minimized.

V. PROPERTIES OF DE\(^d\)

In this section, the properties of DE\(^d\) are examined by optimizing a simple test function.

A. Test Function

The following sphere function \(F_1\) in Eq.(15), which includes the variable \(x_3\) that is independent of the value of \(F_1\), is used for the test function. \(F_1\) is minimized.

\[
F_1(x_1, x_2, x_3) = (x_1 - 1)^2 + (x_2 - 4)^2
\]

The function \(F_1\) has the minimum value at \((x_1, x_2) = (1, 4)\). Since \(x_1\) and \(x_2\) contribute to the value of \(F_1\), it is preferable that they are represented by effective genes. On the contrary, since \(x_3\) is independent of the value of \(F_1\), it is preferable that \(x_3\) is represented by an ineffective or highly damaged gene and is reduced finally.

To show the advantage of DE\(^d\) and mappings, three algorithms DE\(^d\), DE\(^d\) without mappings and GA\(^d\) are compared. The condition of the experiment is as follows:

- The representation of genes: Every variable \(x_i\) is represented by a gene of which the normal value and the damaged rate are represented by real numbers. The length of a chromosome \(L = 3\).
- The creation of the initial population: The normal values and the damaged rates in initial individuals are generated as uniform random numbers from the interval \([-5, 12, 5, 11]\) and \([0, 1]\), respectively. These ranges are the same as that of MGGA and GA\(^d\) [14], [18].
- The other common conditions: The population size \(N = 50\). The damaged rates are changed by biased mutation and the increment/decrement \(\Delta d\) is given by a normal random variable obeying \(N(0, 0.2^2)\). To examine the effect of degeneration pressure, cases where biased mutation rate \(P_{m} = 0.5L, 1L, 2L\) and constant damaging probability function \(P_{dam}(d) = 1, 0.95, 0.9\) are tested.
- Conditions for DE\(^d\) and DE\(^d\) without mappings: DE/rand/1/exp is used with scaling factor \(F = 0.8\) and crossover rate \(CR = 0.95\). In DE\(^d\), the mapping to phenotype values and the inverse mapping to normal values are used and damaged rates are recomputed by uniform crossover. In DE\(^d\) without mappings, normal values are optimized using DE operations independently with damaged rates which are recomputed by uniform crossover. The reversible mutation is not used. As for survivor selection, the trial vector \(H'\) is accepted for the next generation if the objective value of trial vector is better than that of parent vector \(H_i\), or \(f(H') < f(H_i)\).
• Conditions for GA$^d$. The maximum expected value in the linear ranking selection $T_{max} = 2.0$. Pairs of a normal value and a damaged rate are recombined by one-point crossover with the crossover rate $P_c = 0.8$.

Each normal value is changed by reversible mutation and the increment/decrement $\Delta g$ is given by a normal random variable obeying $N(0, 0.2d^2)$.

Every trial in an experiment is continued for 200 generations and the average result of 20 trials is used for evaluation.

**B. Experimental Results**

Table I shows the results of the experiments for $F_1$. The columns labeled “Algorithm”, “$P_{bm}$” and “$P_{dam}$” show the name of algorithm, the biased mutation rate and the value of damaging probability function, respectively. The columns “$avg$” and “$std$” under the label “$F_1$” show the average values of $F_1$ and its standard deviations. The columns “$min$”, “$avg$”, “$max$” and “$std$” under the label “#parameters” shows the minimum, average and maximum numbers of effective variables and its standard deviations, respectively.

DE$^d$ found very good objective values for all parameter settings. Also, DE$^d$ succeeded to reduce the unnecessary variable $x_3$ in almost all cases except for one trial in $P_{bm} = 0.5/L$ and $P_{dam}(d) = 0.95$. As $P_{bm}$ and $P_{dam}$ becomes higher, degeneration pressure becomes higher and unnecessary variables are reduced easier, but it becomes more difficult to find good objective values in general. DE$^d$ found very good objective value 7.28×$10^{-11}$ on average in highest degeneration pressure of $P_{bm} = 2/L$ and $P_{dam}(d) = 1$.

Thus, DE$^d$ is the algorithm that is robust to high degeneration pressure and can find near optimal solution with reducing unnecessary parameters.

As for DE$^d$ without mappings, it found objective values ranged from about $10^{-4}$ to $10^{-5}$ that is very worse than those found by DE$^d$. Also, DE$^d$ without mappings could not reduce the variable $x_3$ stably. Thus, it is thought that the mapping and the inverse mapping are essential for DE$^d$. As for GA$^d$, it could reduce the variable $x_3$ stably if degeneration pressure is properly given such as $P_{bm} = 0.5/L$ and $P_{dam}(d) = 1$.

However, objective value found on average in this case is $4.44 \times 10^{-6}$ which is very worse than that found by DE$^d$. Also, when degeneration pressure becomes high, GA$^d$ often failed to find satisfactory solutions. Therefore, DE$^d$ is more effective for structural learning than GA$^d$.

**VI. STRUCTURAL LEARNING OF NEURAL NETWORKS**

In this section, the representation of neural networks and structural learning of them is described.

**A. Coding**

A chromosome $G^d$ which represents a neural network which consists of layers. Each layer consists of neurons and each neuron consists of connection weights and a threshold as follows:

$$G^d = L^2L^3\cdots L^m$$

$$L^k = N_1^kN_2^k\cdots N_k^k$$

$$N_i^k = (w_{i1}^{k}\phi_{1}^{k})\cdots (w_{i(n-1)}^{k}\phi_{(n-1)}^{k})(\theta_i^k\phi_{0}^{k})$$

where $L^k$ is the $k$-th layer, and $m$ is the number of layers. $N_i^k$ is the $i$-th neuron in the $k$-th layer, and $n^k$ is the number of neurons in the $k$-th layer. $(w_{ij}^k\phi_{j}^{k})$ represents the connection weight between the $i$-th neuron in the $k$-th layer and the $j$-th neuron in the $(k-1)$-th layer. $(\theta_i^k\phi_{0}^{k})$ represents the threshold of the $i$-th neuron in the $k$-th layer.

From the chromosome, the following weights and thresholds are obtained.

$$w_{ij}^k = w_{ij}^k(1 - d_{ij}^k)$$

$$\theta_i^k = \theta_i^k(1 - d_{0i}^k)$$

**B. Fitness value**

The output of neurons to an input pattern $I^p$ is obtained by the following equations:

$$O_i^k(I^p) = f(\sum_j w_{ij}^kO_j^{k-1}(I^p) - \theta_i^k)$$

$$f(x) = \frac{1}{1 + \exp(-x)}$$

where $O_i^k$ is the output of the $i$-th neuron in the $k$-th layer. $f(\cdot)$ is an output function.

The fitness value of learning individuals is given by the mean square error (MSE) to training data as follows:

$$MSE = \frac{1}{P} \sum_{i} (O_i^p - O_{i}^{m}(I^p))^2$$

TABLE I

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$P_{bm}$</th>
<th>$P_{dam}$</th>
<th>$F_1$</th>
<th>#parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE$^d$</td>
<td>0.5 0.9</td>
<td>2.71e-32 6.70e-32</td>
<td>2 0.00 2.00 0.00</td>
<td></td>
</tr>
<tr>
<td>0.5 0.95</td>
<td>5.96e-31 9.75e-31</td>
<td>2 0.05 3 0.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5 1</td>
<td>7.52e-30 2.07e-29</td>
<td>2 0.00 2.00 0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0.9</td>
<td>1.05e-26 3.07e-26</td>
<td>2 2.00 2.00 0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0.95</td>
<td>1.81e-24 6.73e-24</td>
<td>2 2.00 2.00 0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td>1.41e-22 4.17e-22</td>
<td>2 2.00 2.00 0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 0.9</td>
<td>9.42e-19 1.40e-18</td>
<td>2 2.00 2.00 0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 0.95</td>
<td>8.02e-16 1.70e-15</td>
<td>2 2.00 2.00 0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 1</td>
<td>7.28e-11 2.14e-10</td>
<td>2 2.00 2.00 0.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2007 IEEE Congress on Evolutionary Computation (CEC 2007)
where \( P \) is the number of patterns, \( O_P^i \) is the teaching signal to the \( i \)-th element of the \( p \)-th input pattern \( I^p \).

### C. Problem and experimental conditions

Logical function \( L \) in (24) is learned. \( L \) includes the variable \( d \) which is independent of the function value.

\[
L(a, b, c, d, e) = (a \cup b) \cap (c \cup e)
\]  

(24)

The function \( L \) is estimated by the neural network which consists of the input layer with 5 neurons, the second layer with 4 neurons, the third layer with 4 neurons and the output layer with 1 neuron. So, the maximum number of parameters \( L \) is 49. The 24 training data are selected randomly from total \( 2^5 = 32 \) data, which are obtained by assigning 0 or 1 to \( a, b, c, d \) and \( e \). The fitness value, which is defined by the mean square error MSE to the training data, is minimized.

The condition of the experiment is as follows:

- **Common conditions:** The population size \( N = 50 \). The normal values and the damaged rates in initial individuals are generated as uniform random numbers from the interval \([-5, 5]\) and \([0, 1]\), respectively. The damaged rates are changed by biased mutation and the increment/decrement \( \triangle d \) is given by a normal random variable obeying \( N(0, 0.2^2) \). To examine the effect of degeneration pressure, cases where biased mutation rate \( P_{\text{dam}} = 1/L, 2/L \) and constant damaging probability function \( P_{\text{dam}}(d) = 1, 0.95, 0.9 \) are tested.

- **Conditions for \( \text{DE}^d \):** \( \text{DE/rand/1/exp} \) is used with scaling factor \( F = 0.8 \) and crossover rate \( CR = 0.95 \). The mapping to phenotype values and the inverse mapping to normal values are used and damaged rates are recombined by uniform crossover. The reversible mutation is not used. As for survivor selection, the trial vector is accepted for the next generation if the objective value of trial vector is better than that of parent vector or the objective value of trial vector is better than allowable error \( \epsilon \) and the number of effective parameters in trial vector is less than that of parent vector. By using this rule, \( \text{DE}^d \) can find solutions with less number of effective parameters. The allowable error \( \epsilon = 0.001 \).

- **Conditions for \( \text{GA}^d \):** The maximum expected value in the linear ranking selection \( \eta_{\max} = 2.0 \). Pairs of a normal value and a damaged rate are recombined by one-point crossover with the crossover rate \( P_c = 0.8 \). Each normal value is changed by reversible mutation and the increment/decrement \( \triangle g \) is given by a normal random variable obeying \( N(0, 0.2^2) \).

Every trial in an experiment is continued for 2,000 generations and the average result of 10 trials is used for evaluation.

### D. Experimental results

Table II shows the result of experiments by \( \text{DE}^d \) and \( \text{GA}^d \) with controlling the degeneration pressure. Apparently, \( \text{DE}^d \) found much better objective values and the smaller structure much more stably than \( \text{GA}^d \). Especially, in highest degeneration pressure \( P_{\text{dam}} = 2/L \) and \( P_{\text{dam}}(d) = 1 \), \( \text{DE}^d \) found solutions with only 8 effective parameters in 7 trials out of 10 trials.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( P_{\text{hm}} )</th>
<th>( P_{\text{dam}} )</th>
<th>( F_1 ) avg</th>
<th>( F_1 ) std</th>
<th>#parameters avg</th>
<th>#parameters std</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{DE}^d )</td>
<td>1 0.9</td>
<td>0.00e+00 0.00e+00</td>
<td>8 9.70 11 0.90</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 0.95</td>
<td>0.00e+00 0.00e+00</td>
<td>8 9.70 17 2.53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>0.00e+00 0.00e+00</td>
<td>8 9.40 11 0.92</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 0.9</td>
<td>1.41e-25 4.23e-25</td>
<td>8 8.80 10 0.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 0.95</td>
<td>3.55e-36 1.07e-35</td>
<td>8 8.60 10 0.66</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 1</td>
<td>0.00e+00 0.00e+00</td>
<td>8 8.30 9 0.46</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{GA}^d )</td>
<td>1 0.9</td>
<td>3.29e-07 9.87e-07</td>
<td>21 25.20 29 2.86</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 0.95</td>
<td>3.10e-13 9.06e-13</td>
<td>16 21.20 24 2.68</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>8.36e-03 2.45e-02</td>
<td>8 16.60 22 4.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 0.9</td>
<td>7.15e-07 2.10e-06</td>
<td>16 20.90 25 2.62</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 0.95</td>
<td>2.82e-03 7.24e-03</td>
<td>15 18.50 21 1.75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 1</td>
<td>4.27e-02 4.27e-02</td>
<td>9 11.60 14 1.62</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5 shows one of the simplest neural networks obtained by \( \text{DE}^d \). Only 7 connections and 1 threshold are used to estimate the logical function. The structure of the network directly reflects the structure of the logical function. This shows \( \text{DE}^d \) clarify the interpretation of hidden units and discover the structure of the function.

**Fig. 5.** One of simplest neural networks obtained by \( \text{DE}^d \)

### VII. Conclusions

We proposed a new idea to realize degeneration by using a mapping to phenotype values and the inverse mapping to normal values. We applied this idea to differential evolution which is known as an efficient and robust evolutionary algorithm, and proposed differential evolution with degeneration, \( \text{DE}^d \). Through optimization of simple test function, it is shown that the mappings are essential to \( \text{DE}^d \). The results of the structural learning of neural networks showed that \( \text{DE}^d \) could find simpler structure with less estimation error than \( \text{GA}^d \). It is thought that \( \text{DE}^d \) has the better ability of structural learning than \( \text{GA}^d \) because \( \text{DE}^d \) is higher optimization ability.
that can find better objective values, is more robust to high
degeneration pressure, and can find smaller structures.

In future, we plan to introduce a dynamic control of
degeneration pressure to find better structure more stably and
to apply DE to the other models such as fuzzy rule bases.
Also, we plan to apply DE to various real world problems
such as prediction of time series in stock markets and foreign
exchange markets.

Some researches [26], [27] performed structural learn-
ing by optimizing parameter values and then selecting the
optimal structure based on information criteria. But it is
difficult to optimize information criteria in general. Also,
many methods can be applied only to a particular type of
model. On the contrary, DE can optimize information
criteria and doesn't depend on a particular model type.
DE is a flexible and general-purpose method of structural learning.

REFERENCES

[1] T. Kurita, “A method to determine the number of hidden units of three
layered neural networks by information criteria,” Transactions of the
Institute of Electronics, Information and Communication Engineers,

criterion-determining the number of hiddenunits for an artificial neural
network model,” IEEE Transactions on Neural Networks, vol. 5, no. 6,


generalization and regularization in nonlinear learning systems,” in
Advances in Neural Information Processing Systems, J. E. Moody, S. J.
Hanson, and R. P. Lippmann, Eds. San Mateo: Morgan Kaufmann,

Advances in Neural Information Processing Systems 4, D. S.
605.

Pruning: Optimal Brain Surgeon,” in Advances in Neural Information
Processing Systems, J. D. C. S. J. Hanson and C. L. Giles, Eds. San

[8] M. C. Mozer and O. Smolensky, “Using relevance to reduce network

network systems,” in Neural Network Systems Technologies and Ap-

architecture,” in Advances in Neural Information Processing Systems
2, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1992,
pp. 642–649.

learning algorithms for pattern classification,” Neural Networks, IEEE


genes,” IEICE Trans. on Information and Systems, vol. J84-D-I, no. 9,

[14] ——, “MGGA: Genetic algorithm with mutant genes.” Systems and

with damaged genes,” in Proceedings of the 9th International Confer-
1211–1215.