Appl Intell (2012) 36:454–471
DOI 10.1007/s10489-010-0271-y

A self-organization mining based hybrid evolution learning for TSK-type fuzzy model design

Sheng-Fuu Lin · Jyun-Wei Chang · Yung-Chi Hsu

Published online: 24 December 2010

Abstract In this paper, a self-organization mining based hybrid evolution (SOME) learning algorithm for designing a TSK-type fuzzy model (TFM) is proposed. In the proposed SOME, group-based symbiotic evolution (GSE) is adopted in which each group in the GSE represents a collection of only one fuzzy rule. The proposed SOME consists of structure learning and parameter learning. In structure learning, the proposed SOME uses a two-step self-organization algorithm to decide the suitable number of rules in a TFM. In parameter learning, the proposed SOME uses the data mining based selection strategy and data mining based crossover strategy to decide groups and parental groups by the data mining algorithm that called frequent pattern growth. Illustrative examples were conducted to verify the performance and applicability of the proposed SOME method.

Keywords Genetic algorithm · Fuzzy model · Group-based symbiotic evolution · Data mining · Identification · FP-Growth

1 Introduction

Neural fuzzy networks [1–3] are capable of inferring complex nonlinear relationships between input and output variables. This property is important when the system to be modeled is nonlinear. The key advantage of the neural fuzzy approach lies in the fact that it does not require a mathematical description of the system while modeling it. The system can perform the nonlinear mapping once the system parameters are trained based on a sequence of input and desired response pairs.

The training of the parameters (parameter learning) is an issue in designing a neural fuzzy system. Backpropagation (BP) training is widely used for solving this issue. It is a powerful training technique that can be applied to networks with feed-forward structure, to transform them into adaptive systems. Since neural fuzzy systems can be represented as layered feedforward networks, the same concept of BP can be applied and be particularly useful in cases where complex interaction among independent variables necessitates training for all system parameters. Since steepest descent optimization technique is used in BP training to minimize the error function, the algorithm may reach the local minima very fast but never find the global solution. Besides, BP training performance depends on the initial values of the system parameters, and for different network topologies one has to derive new mathematical expressions for each network layer.

Considering the aforementioned disadvantages one may be faced with suboptimal performance even for a suitable neural fuzzy network topology. Hence, techniques capable of training the system parameters and finding the global solution while optimizing the overall structure are needed. In this respect, recently, several evolutionary algorithms, such as genetic algorithm (GA) [4], genetic programming [5], evolutionary programming [6], and evolution strategies [7], have been proposed. They are parallel and global search techniques. Because they simultaneously evaluate many points in the search space, they are more likely to converge toward the global solution. Therefore, an evolutionary method using for training the fuzzy model has become an important field.

S.-F. Lin (✉) · J.-W. Chang
Department of Electrical Engineering, National Chiao Tung University, Hsinchu City, Taiwan, Republic of China
e-mail: sflin@mail.nctu.edu.tw

Y.-C. Hsu
Graduate Institute of Network Learning Technology, National Central University, Jhongli City, Taiwan, Republic of China

The evolutionary fuzzy model generates a fuzzy model automatically by incorporating evolutionary learning procedures [8–18], where the most well-known procedure is the genetic algorithm (GA). Several genetic fuzzy models have been proposed in [8–16]. In [8], Karr applied GA to the design of the membership functions of a fuzzy controller, with the fuzzy rule set assigned in advance. Since the membership functions and rule sets are co-dependent, simultaneous design of these two approaches will be a more appropriate methodology. In [9], a fuzzy controller design method uses a GA to find the membership functions and the rule sets simultaneously. In [8] and [9], the input space was partitioned into a grid. The number of fuzzy rules (i.e., the length of each chromosome in the GA) increased exponentially as the dimension of the input space increased. In [10], a GA was used to tune the consequent parameters of TSK-type fuzzy rules [1] as well as the membership functions in the precondition parts. Juang [11] proposed a TSK-type recurrent fuzzy network with a GA for control problems. In [12], Juang et al. proposed genetic reinforcement learning in designing fuzzy controllers. In [13], a fuzzy-genetic algorithm used preprocessing data at the remote terminal unit in a power system. In [14, 15], Alcalá et al. proposed smartly tuned fuzzy logic controllers in heating, ventilating, and air conditioning (HVAC) systems. In [16], Kaya and Alhajj used GA to dynamically adjust and optimize membership functions.

Recently, several improved evolution algorithms have been proposed [17–21]. In [17], Bandyopadhyay et al. used the variable-length genetic algorithm (VGA) that allows the differentia of lengths of chromosomes in a population. Carse et al. [18] used the genetic algorithm to evolve fuzzy rule based controllers. Lin and Xu [19] proposed a sequential-search based dynamic evolution (SSDE) to ensure better-performing chromosomes will be initially generated and better mutation points will be determined for performing dynamic-mutation at upcoming generation. In [20] the group-based symbiotic evolution (GSE) was proposed to solve the issue of the traditional GA that all the fuzzy rules were encoded into one chromosome. In [21], authors proposed a hybrid evolution learning algorithm (HELA). The HELA combines the compact genetic algorithm (CGA) and the modified variable-length genetic algorithm, and performs the structure/parameter learning for constructing the network dynamically.

Although the above evolution learning algorithms [17–21] can improve the traditional genetic algorithms, these algorithms may encounter one or more of the following issues: 1) all the fuzzy rules are encoded into one chromosome; 2) the numbers of fuzzy rules have to be assigned in advance; 3) the population cannot evaluate each fuzzy rule locally; 4) the selection and crossover steps are vulnerable to local optima solution.

Recently, data mining becomes a popular field [22, 23]. Data mining is a method of mining information in a database formed by transactions. The data mining can be regarded as a new way of data analysis. One goal of data mining is to find association rules among frequent item sets in transactions. Several methods have been proposed to achieve goal [24–27]. In [24], the authors proposed a mining method of ascertain large item sets to find association rules in transactions. Han et al. [25] proposed the frequent pattern growth (FP-Growth) to mine frequent patterns without candidate generation. In [26], an algorithm of data mining for transaction data with quantitative values was proposed. In [26], each quantitative item is translated to a fuzzy set and the authors use these fuzzy sets to find fuzzy rules. Wu et al. [27] proposed a data mining based GA algorithm to efficiently improve the Traditional GA by using analyzing support and confidence parameters. Since data mining is able to find the information within large item sets, it should be useful to solve the problems that mentioned above.

In this paper, a self-organization mining based hybrid evolution learning (SOME) for designing a TFM is proposed for improving the evolution learning algorithms. The SOME consists of structure and parameter learning. In structure learning, the SOME determines the number of fuzzy rules automatically and processes the variable combination of chromosomes. The SOME proposes a two-step self-organization algorithm (TSSO) to decide the suitable number of rules. In the TSSO, the modified compact genetic algorithm (MCGA) [21] is adopted. Different from [21], the TSSO uses two steps for determining the suitable number of rules to avoid the number of rules may from falling into local optima solution.

In parameter learning of the SOME, this paper proposes the data mining based selection strategy (DMSS) and the data mining based crossover strategy (DMCS) to decide the selected group indexes and parental group indexes of chromosomes by FP-Growth. In the DMSS, the groups are selected according to the frequent item sets based on three different actions. These three actions are the normal, the searching, and the exploring action. After performing the DMSS, the suitable groups are obtained and the chromosomes are selected from these groups. In the DMCS, similar to the DMSS, the parental groups are selected based on the same three actions. After performing the DMCS and DMSS, the good combination of individuals can be retrieved while the exploration of other combinations continues to avoid the formerly-retrieved information from falling into the local optimum solution.

The advantages of the proposed SOME are summarized as follows: 1) the SOME uses the GSE so that each group represents only one fuzzy rule; 2) the TSSO is proposed to decide the suitable number of rules; 3) the SOME uses the GSE to evaluate the fuzzy rule locally; 4) the DMSS and DMCS are proposed to select the suitable group indexes and parental group indexes.

This paper is organized as follows. Section 2 introduces the TFM. The proposed SOME is described in Sect. 3. Section 4 presents the simulation results. The conclusions are summarized in the last section.

2 Structure of TSK-type Fuzzy Model (TFM)

A fuzzy controller is a knowledge-based system characterized by a set of rules, which model the relationship among control input and output. The reasoning process is defined by means of the employed aggregation operators, the fuzzy connectives and the inference method. The fuzzy knowledge base contains the definition of fuzzy sets stored in the fuzzy database and a collection of fuzzy rules, which constitute the fuzzy rule base. Fuzzy rules are defined by their antecedents and consequents, which relate an observed input state to a desired control action. Most fuzzy systems employ the inference method proposed by Mamdani in which the consequence parts are defined by fuzzy sets [1]. A Mamdani-type fuzzy rule has the form:

$$\begin{aligned}&\text{IF } x_1 \text{ is } A_{1j}(m_{1j},\sigma_{1j}) \text{ and } x_2 \text{ is } A_{2j}(m_{2j},\sigma_{2j})\\ &\quad\text{and } \ldots \text{ and } x_n \text{ is } A_{nj}(m_{nj},\sigma_{nj})\\ &\text{THEN } y' \text{ is } B_j(m_j,\sigma_j)\end{aligned} \tag{1}$$

where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation with ith dimension and jth rule node respectively. The consequences B_j of jth rule is aggregated into one fuzzy set for the output variable y'. The crisp control action is obtained through defuzzification, which calculates the centroid of the output fuzzy set. Besides the common fuzzy inference method proposed by Mamdani,

Takagi, Sugeno and Kang introduced a modified inference scheme [3]. The first two parts of the fuzzy inference process, fuzzifier the inputs and applying the fuzzy operator are exactly the same. A Takagi-Sugeno-Kang (TSK) type controller employs different implication and aggregation methods than the standard Mamdani controller. Instead of using fuzzy sets the conclusion part of a rule, is a linear combination of the crisp inputs.

$$\begin{aligned}&\text{IF } x_1 \text{ is } A_{1j}(m_{1j},\sigma_{1j}) \text{ and } x_2 \text{ is } A_{2j}(m_{2j},\sigma_{2j})\\ &\quad\text{and } \ldots \text{ and } x_n \text{ is } A_{nj}(m_{nj},\sigma_{nj})\\ &\text{THEN } y' = w_{0j} + w_{1j}x_1 + \ldots + w_{nj}x_n.\end{aligned} \tag{2}$$

Since the consequence of a rule is crisp, the defuzzification step becomes obsolete in the TSK inference scheme. Instead, the control output is computed as the weighted average of the crisp rule outputs, which is computationally less expensive then calculating the center of gravity.

In this paper, the TFM is adopted to solve nonlinear control problems. The structure of the TFM is shown in Fig. 1, where n and M are, respectively, the number of input dimensions and the number of rules. It is a five-layer network structure. The functions of the nodes in each layer are described as follows:

Layer 1 (Input Node) No function is performed in this layer. The node only transmits input values to layer 2. That is

$$u_i^{(1)} = x_i. \tag{3}$$

Layer 2 (Membership Function Node) Nodes in this layer correspond to one linguistic label of the input variables in layer 1; that is, the membership value specifying the degree to which an input value belongs to a fuzzy set is calculated in this layer. For an external input x_i, the following Gaussian membership function is used:

$$u_{ij}^{(2)} = \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right) \tag{4}$$

where m_{ij} and σ_{ij} are, respectively, the center and the width of the Gaussian membership function of the jth term of the ith input variable x_i.

Layer 3 (Rule Node) The output of each node in this layer is determined by the fuzzy AND operation. Here, the product operation is utilized to determine the firing strength of each rule. The function of each rule is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)}. \tag{5}$$

Layer 4 (Consequent Node) Nodes in this layer are called consequent nodes. The input to a node in layer 4 is the output delivered from layer 3, and the other inputs are the input variables from layer 1 as depicted in Fig. 1. For this kind of node, we have

$$u_j^{(4)} = u_j^{(3)}\left(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i\right), \tag{6}$$

where the summation is over all the inputs and where w_{ij} are the corresponding parameters of the consequent part.

Layer 5 (Output Node) Each node in this layer corresponds to one output variable. The node integrates all the actions recommended by layers 3 and 4 and acts as a defuzzifier with

$$\begin{aligned} y = u^{(5)} &= \frac{\sum_{j=1}^{M_k} u_j^{(4)}}{\sum_{j=1}^{M_k} u_j^{(3)}}\\ &= \frac{\sum_{j=1}^{M_k} u_j^{(3)}(w_{0j} + \sum_{i=1}^{M_k} w_{ij}x_i)}{\sum_{j=1}^{M_k} u_j^{(3)}}\end{aligned} \tag{7}$$

where M_k is the number of fuzzy rule.

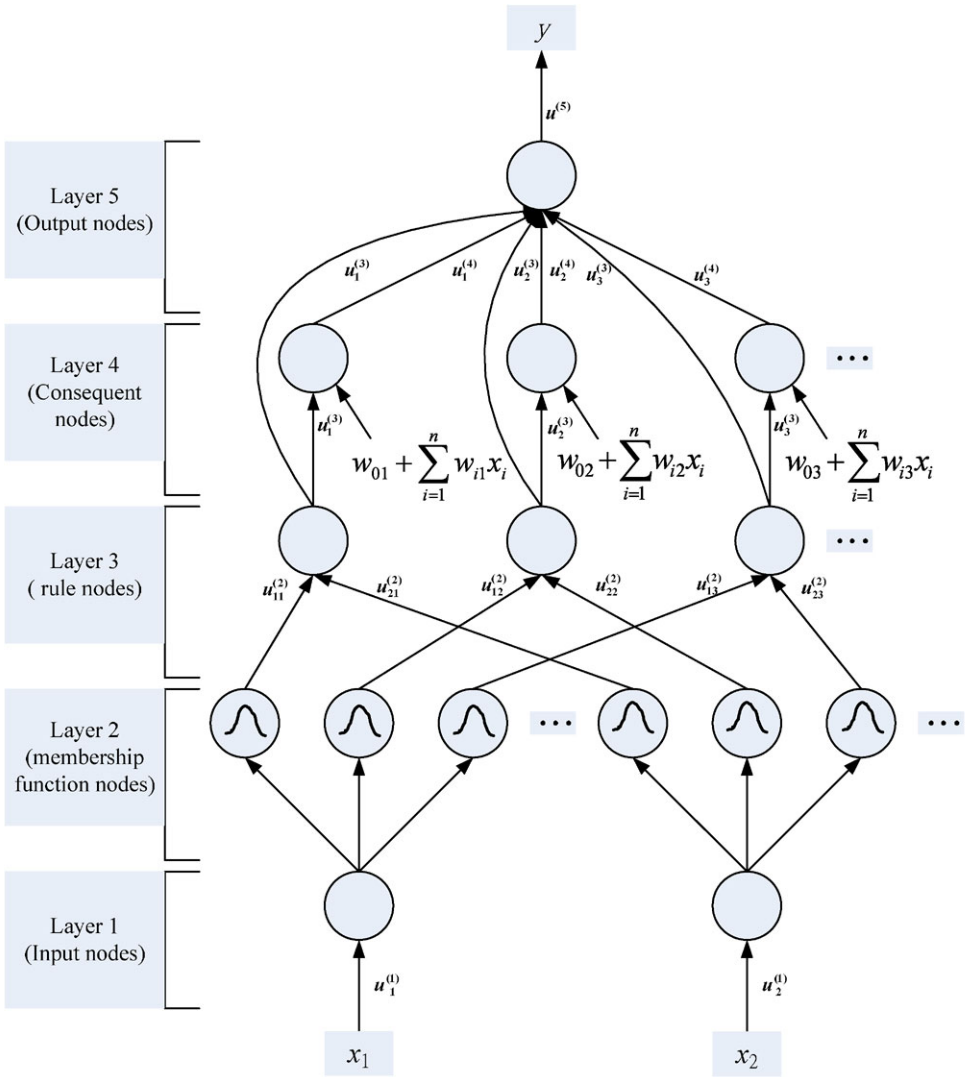

Layer 5
(Output nodes)
Layer 4
(Consequent nodes)
Layer 3
(rule nodes)
Layer 2
(membership function nodes)
Layer 1
(Input nodes)
y
$u^{(5)}$
$u_1^{(3)}$ $u_1^{(4)}$ $u_2^{(3)}$ $u_2^{(4)}$ $u_3^{(3)}$ $u_3^{(4)}$
$w_{01}+\sum_{i=1}^{n} w_{i1}x_i$
$w_{02}+\sum_{i=1}^{n} w_{i2}x_i$
$w_{03}+\sum_{i=1}^{n} w_{i3}x_i$
$u_{11}^{(2)}$ $u_{21}^{(2)}$ $u_{12}^{(2)}$ $u_{22}^{(2)}$ $u_{13}^{(2)}$ $u_{23}^{(2)}$
$u_1^{(1)}$ $u_2^{(1)}$
x_1 x_2

Fig. 1 Structure of the proposed TFM

3 A self-organization mining based hybrid evolution learning

This section will introduce the SOME. Recently, many researches try to enhance the traditional GAs have been made [28–30]. One category of them tries to modify the structure of a population. Examples in this category include the distributed GA [28], the cellular GA [29], and the symbiotic GA [30].

In structure learning of these papers, the SOME determines the number of fuzzy rules automatically and processes the variable combination of chromosomes. The length of combination of chromosomes denotes the rule sets that can form a TFM. In traditional symbiotic evolution, each individual in the population represents only a partial solution. The complete solution was consisted of several individuals. The partial solution can be characterized as specializations which avoid converging to local optimum solution [30]. Although the specialization property ensured diversity, there still has a problem which the population cannot evaluate performance of each partial solution locally. To cope with this, the SOME adopts the GSE. The GSE is different from the traditional symbiotic evolution; with each population in the GSE method is divided to several groups. Each group represents a set of the chromosomes that belong to one single fuzzy rule. Each group will perform the reproduction and crossover operations in each generation independently.

In the proposed SOME, the numbers of rules in a TFM are variable. The structure of chromosomes in the SOME is shown in Fig. 2. For determining the numbers of fuzzy rules, the TSSO is proposed. In the TSSO, a building block (BB) is used to determine the number of fuzzy rules and the number of selection times of the TFM with different number of fuzzy rules. In Fig. 3, the TSSO codes the probability vector into the BBs, M_k represents k rules that used to form a TFM; V_{M_k} is a probability vector which represents the suitability of a TFM model with k rules. In the

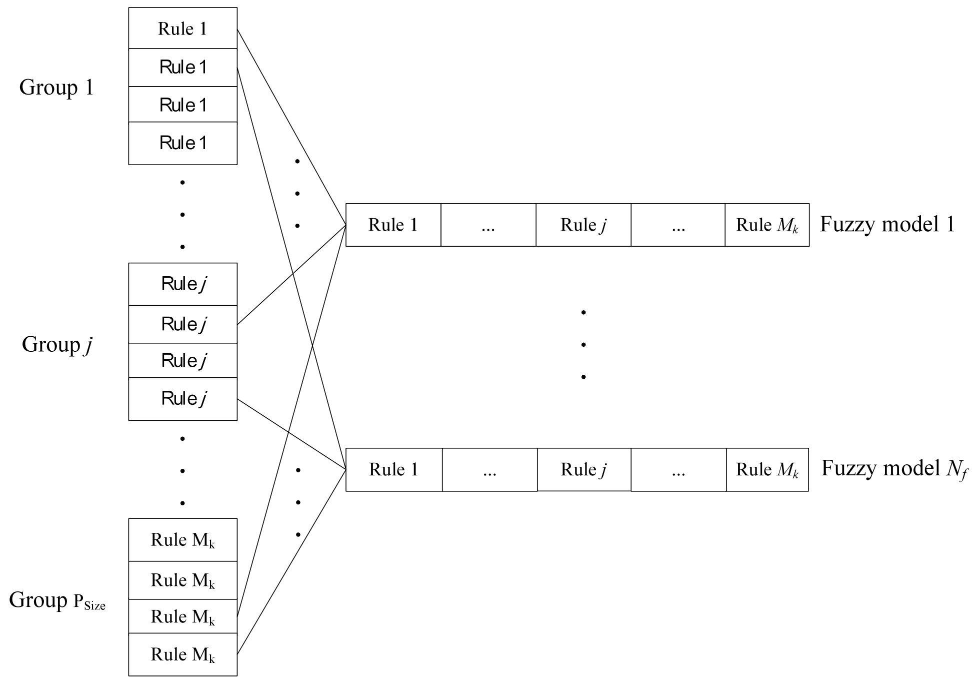


Fig. 2 The structure of the chromosome in the SOME

$V_{M_{\min}}$	$V_{M_{\min+1}}$	…	V_{M_k}	…	$V_{M_{\max\text{-}1}}$	$V_{M_{\max}}$

Fig. 3 Coding the probability vector into the building blocks (BBs) in the TSSO

TSSO, the maximum number of rules ($M_{\max}$) and the minimum number of rules ($M_{\min}$) must be predefined in advance.

In parameter learning, this study proposes the data mining based selection strategy (DMSS) and the data mining based crossover strategy (DMCS) by the FP-growth to decide M_k and to decide parental groups that are used to perform crossover step. The FP-Growth was proposed by Han et al. In the proposed DMSS, the FP-Growth is used to find the frequent group sets from transactions. In this paper, a transaction means the collection of well-performing chromosomes selected from M_k groups to form a TFM. After the frequent group sets have been found, the DMSS uses three actions to form a TFM with k rules. The three actions are defined as the normal, searching, and exploring actions. In normal action, the selection of groups that are used to perform crossover is random. In search action, the groups are chosen from the frequent patterns obtained by FP-growth. And in explore action, the groups are chosen from the non-frequent patterns set to avoid the mined frequent patterns from falling into local optimum. In the DMCS, the parental groups are selected according to same three actions (normal, searching, and exploring actions). The whole learning process of the SOME is shown in Fig. 4.

In the proposed SOME, the coding structure of the chromosomes must be fit for group based symbiotic evolution. Figure 5 describes a fuzzy rule that has the form of (2) where m_{ij}, σ_{ij}, and w_{ij} represent a Gaussian membership function with mean and deviation and weight with ith dimension and jth rule node and w_{0j} represents the coefficient with jth rule.

The learning process of the SOME in each group involves seven major operators: initialization, the TSSO, the DMSS, fitness assignment, elite-based reproduction strategy (ERS), the DMCS, and the self-organization mutation strategy (SOMS). The whole learning process is described step-by-step as follows:

3.1 Initialization step

Before the SOME is designed, individuals forming several initial groups should be generated. The initial groups of the SOME are generated randomly within a fixed range. The following formulations show how to generate the initial chromosomes in each group:

Deviation:

$$Chr_{g,c}[p] = random[\sigma_{\min}, \sigma_{\max}]$$
$$\text{where } p = 2, 4, \ldots, 2n;$$
$$g = 1, 2, \ldots, M_k;\ c = 1, 2, \ldots, N_C; \tag{8}$$

Mean:

$$Chr_{g,c}[p] = random[m_{\min}, m_{\max}]$$
$$\text{where } p = 1, 3, \ldots, 2n-1; \tag{9}$$

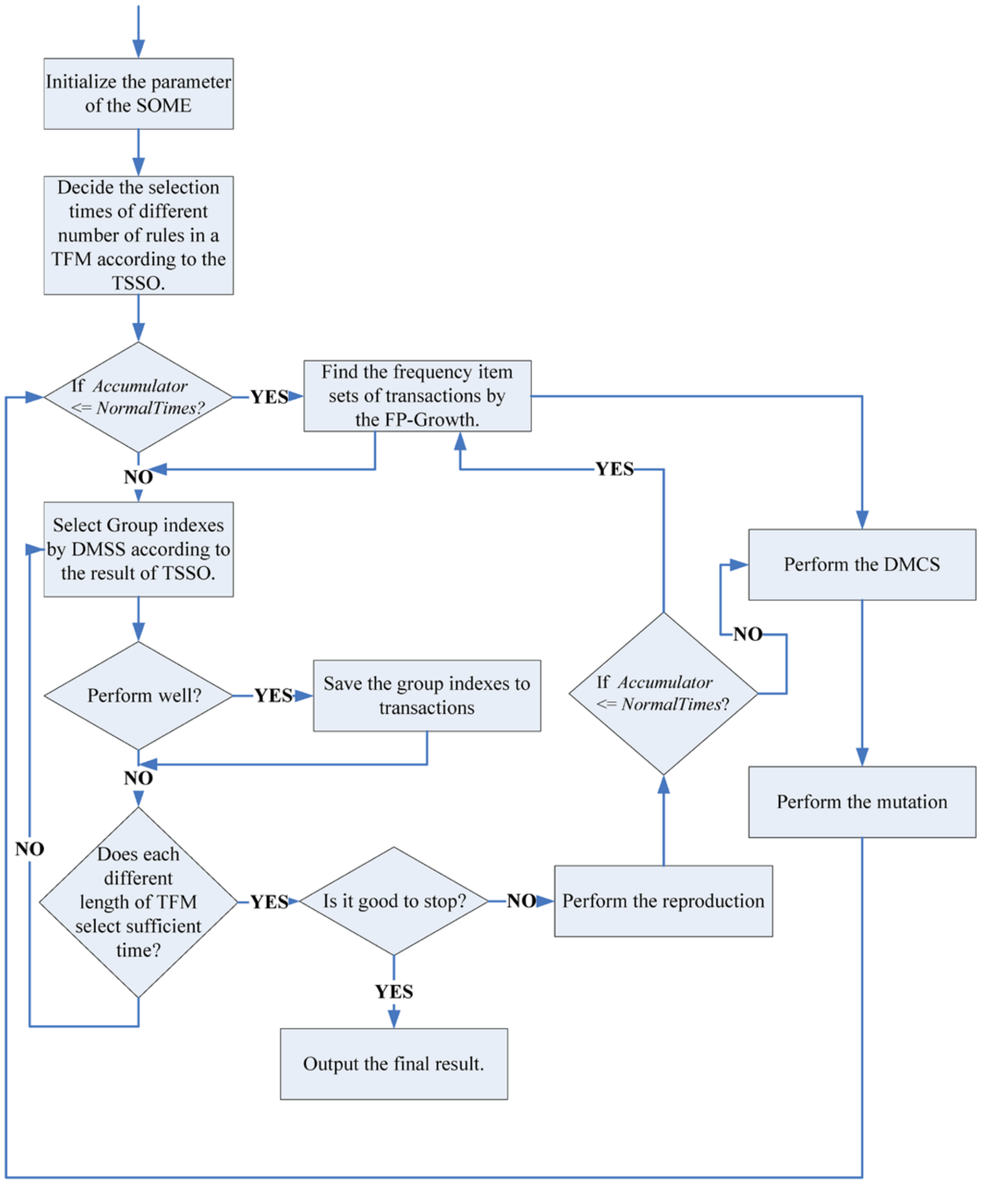

Initialize the parameter of the SOME
Decide the selection times of different number of rules in a TFM according to the TSSO.
If Accumulator <= NormalTimes?
YES
Find the frequency item sets of transactions by the FP-Growth.
NO
YES
Select Group indexes by DMSS according to the result of TSSO.
Perform the DMCS
NO
Perform well?
YES
Save the group indexes to transactions
If Accumulator <= NormalTimes?
NO
Perform the mutation
NO
Does each different length of TFM select sufficient time?
YES
Is it good to stop?
NO
Perform the reproduction
YES
Output the final result.

Fig. 4 The learning process of the SOME method

m_{1j}	σ_{1j}	m_{2j}	σ_{2j}	$\cdots$	m_{nj}	σ_{nj}	w_{0j}	w_{1j}	$\cdots$	w_{nj}

Fig. 5 Coding a rule of a TFM into a chromosome in the SOME

Weight:

$$Chr_{g,c}[p] = random[w_{\min}, w_{\max}]$$

$$\text{where } p = 2n+1, 2n+2, \ldots, 2n+(n+1), \tag{10}$$

where $Chr_{g,c}$ represents cth chromosome in gth group; M_k represents k rules that used to form a TFM and N_C is the total number of chromosomes in each group; p represents the pth gene in a $Chr_{g,c}$; and $[\sigma_{\min}, \sigma_{\max}]$, $[m_{\min}, m_{\max}]$, and $[w_{\min}, w_{\max}]$ represent the range that are predefined to generate the chromosomes.

3.2 Two-step self-organization algorithm (TSSO)

After every group is initialized, the SOME proposes the TSSO to decide the suitable selection times of each number of rules (in this paper the number of rules lie between $[M_{\max}, M_{\min}]$); that is, it determines the selection times of M_k groups which form a TFM with k rules. After the TSSO, the selection times of the suitable number of rules in a TFM will increase, and the selection times of the unsuitable number of rules will decrease. The details of the TSSO are listed as follows:

Step 0 Initialize the probability vectors of the BBs:

$$V_{M_k} = 0.5, \quad \text{where } M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max} \tag{11}$$

$$Accumulator = 0 \tag{12}$$

Step 1 Update the probability vectors of the BBs according to the following equations:

$$\begin{cases} V_{M_k} = V_{M_k} + (Upt\_value_{M_k} * \lambda), & \text{if } Avg \le fit_{M_k} \\ V_{M_k} = V_{M_k} - (Upt\_value_{M_k} * \lambda), & \text{otherwise} \end{cases}$$
$$\text{where } M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max} \tag{13}$$

$$Avg = \sum_{M_k=M_{\min}}^{M_{\max}} fit_{M_k} / (M_{\max} - M_{\min}) \tag{14}$$

$$Upt\_value_{M_k} = fit_{M_k} \bigg/ \sum_{M_k=M_{\min}}^{M_{\max}} fit_{M_k} \tag{15}$$

$$\text{if } Fitness_{M_k} \ge (Best\_Fitness_{M_k} - ThreadFitnessvalue)$$
$$\text{then } fit_{M_k} = fit_{M_k} + Fitness_{M_k}; \tag{16}$$

where V_{M_k} is the probability vector in the BBs and represents the suitable number of k rules; λ is a predefine threshold value; Avg represents the average fitness value in the whole population; $Best\_Fitness_{M_k}$ represents the best fitness value with k rules; fit_{M_k} is the sum of fitness value with k rules when the fitness value with k rules greater than $Best\_Fitness_{M_k}$ minus a predefined threshold value named $ThreadFitnessvalue$. As shown in (13), if $fit_{M_k} \ge Avg$, then the suitable k rules that from a TFM should be increased. On the other hand, if $fit_{M_k} < Avg$, then the k rules that from a TFM should be decreased.

Step 2 Determine the selection times according to the probability vectors of the BBs as follows:

$$Rp_{M_k} = (Selection\_Times) * (V_{M_k}/Total\_Velocy)$$
$$\text{where } M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max}; \tag{17}$$

$$Total\_Velocy = \sum_{M_k=V_{\min}}^{V_{\max}} V_{M_k} \tag{18}$$

where $Selection\_Times$ represents total selection times in each generation; $Rp_{M_k}$is the selection times of M_k groups that use to select k chromosomes for constructing a TFM.

Step 3 After step 2, the selection times of every numbers of rules in a TFM are obtained. Then the Rp_{M_k} times are used to select k chromosomes form M_k groups to construct a TFM.

Step 4 In the proposed TSSO, for avoiding suitable M_k groups may fall in the local optima solution, the TSSO proposed two different actions to update the V_{M_k}. Decide the deferent action according to the following equations:

$$\text{if } Accumulator \le TSSATimes$$
$$\text{then do Step 1, Step 2, and Step 3;} \tag{19}$$

$$\text{if } Best\_Fitness_g = Best\_Fitness$$
$$\text{then } Accumulator = Accumulator + 1; \tag{20}$$

$$\text{if } Accumulator > TSSATimes$$
$$\text{then do Step 0 and } Accumulator = 0, \tag{21}$$

where $TSSOTimes$ is a predefined value; $Best\_Fitness_g$ represents the best fitness value of the best combination of chromosomes in gth generation; $Best\_Fitness$ represents the best fitness value of the best combination of chromosomes in current generations. Equations (19)–(21) represents that if the fitness is not changed for a sufficient number of generations, the suitable M_k groups may fall in the local optima solution.

3.3 The data mining based selection strategy (DMSS)

After the TSSO step, the selection times of each rule number in a TFM is decided. The SOME then performs selection step. The selection step in the SOME can be divided by selection of groups and chromosomes. In the selection of groups, this paper proposes the DMSS to improve the random selection. In the DMSS, the groups are selected according to the frequent patterns found by FP-Growth. In the proposed DMSS, the FP-Growth finds the frequent groups from a transaction (in this paper a transaction means a set of the M_k group indexes that perform well). After the frequent group indexes have been found, the DMSS selects the M_k groups indexes according to the frequent group indexes. To avoid the frequently-occurring groups from falling in the local optimal solution, the DMSS uses three actions to select M_k groups. The three actions defined in this paper are normal, search, and explore. The detail of the DMSS is shown as follows:

Step 0 The transactions are building as follow equation:

$$\text{if } Fitness_{M_k} \ge Best\_Fitness_{M_k} - ThreadFitnessvalue$$
$$\text{then } Transaction_j[i] = TNFSRuleSet_{M_k}[i]$$
$$\text{where } i = 1, 2, \ldots, M_k;$$
$$M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max};$$
$$j = 1, 2, \ldots, TransactionNum, \tag{22}$$

where the $Fitness_{M_k}$ represents the fitness value of TFM with k rules; $ThreadFitnessvalue$ is the predefined value; $TransactionNum$ is the total number of transactions $Transaction_j[i]$ represents the ith item in the jth transaction; and $TNFSRuleSet_{M_k}[i]$ represents ith group index in M_k group indexes that are selected to form a TFM with k rules. The transactions have the form that shown in Table 1. As shown in Table 1, every transaction represents M_k group indexes

Table 1 Transactions in a FP-Growth

Transaction index	Group indexes
1	1, 4, 8
2	2, 4, 7, 10
⋮	⋮
TransactionNum	1, 3, 4, 6, 8, 9

that form a TFM with k rules. For example, as shown in Table 1, the first transaction of the transaction set means the 3 rules TFM that select from 1st group, 4th group, and 8th group has a well performance. The building transactions step continues in normal, searching, and exploring actions.

Step 1 Normal action:

After building up the transactions, the DMSS selects group according to different action types. If the action type is normal action, the DMSS selects the group as following equation:

$$\begin{aligned}&\text{if } \mathit{Accumulator} \le \mathit{NormalTimes}\\&\text{then } \mathit{GroupIndex}[i] = \mathit{Random}[1, \mathit{PulationSize}]\\&\quad\text{where } i = 1, 2, \ldots, M_k;\ M_k = M_{\min},\\&\qquad\quad M_{\min+1}, \ldots, M_{\max},\end{aligned} \tag{23}$$

where *Accumulator* is defined in (20); $\mathit{GroupIndex}[i]$ represents selected ith group index of the M_k group indexes and *PulationSize* represents there are *PulationSize* groups in a population in the SOME.

Step 2 Find the frequent groups:

If the action is searching or exploring action, the DMSS uses the FP-Growth to find frequent group indexes in transactions. The frequent group indexes are found according to the predefined *Minimum_Support*. The *Minimum_Support* means the minimum fraction of transactions that contain an item set. The FP-Growth algorithm can be viewed as two parts: construction of the FP-tree and FP-growth. The sample transactions shown in Table 2 are taken as examples. $\mathit{Minimum\_Support} = 3$ is considered in this example. Frequent group indexes generated by FP-growth shown in Table 3 are then thrown into the pool that's named *FrequentPool*.

Step 3 Select the group indexes according to different actions:

After obtaining the frequent item sets, the DMSS selected group indexes according to different actions that describe as follows:

Table 2 Sample transactions

Transaction index	Group indexes
1	{b, c, e, f, g, h, p}
2	{a, b, c, f, i, m, o}
3	{c, f, i, m, o}
4	{b, c, e, s, p}
5	{a, b, c, d, f, m, o}

Table 3 Frequent group indexes generated by FP-growth with $\mathit{Minimum\_Support} = 3$

Suffix group	Cond. group base	Cond. FP-tree	Frequent group indexes
B	c:4	c:4	cb:4
F	cb:3, c:1	c:4, cb:3	cf:4, bf:3, cbf:3
M	cbf:2, cf:1	cf:3	cm:3, fm:3, cfm:3
O	cbfm:2, cfm:1	cfm:3	co:3, fo:3, mo:3, cfo:3, cmo:3, fmo:3, cfmo:3

(a) In the searching action, the group indexes are selected from the frequent item as follow equations:

$$\begin{aligned}&\text{if } \mathit{NormalTimes} < \mathit{Accumulator}\\&\qquad\qquad \le \mathit{SearchingTimes}\\&\text{then } \mathit{GroupIndex}[i] = w,\\&\quad\text{where}\\&\quad w = \mathit{Random}[1, \mathit{PulationSize}] \quad \text{and}\\&\qquad w \in \mathit{FrequentItemSet}[q];\\&\quad \mathit{FrequentItemSet}[q] = \mathit{Random}[\mathit{FrequentPool}];\\&\quad q = 1, 2, \ldots, \mathit{FrequentPoolNum};\\&\quad i = 1, 2, \ldots, M_k;\ M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max},\end{aligned} \tag{24}$$

where *SearchingTimes* is a predefined value that judge to perform searching action; *FrequentPool* represents the sets of frequent item set that obtain from FP-Growth; *FrequentPoolNum* presents the total number of sets in *FrequentPool* and $\mathit{FrequentItemSet}[i]$ presents a frequent item set that select from *FrequentPool* randomly. In (24), if M_k greater than the size of $\mathit{FrequentItemSet}[i]$, the remaining groups are selected by (23).

(b) In the exploring action, the group indexes are selected according to the frequent item as follow equations:

$$\begin{aligned}&\text{if } \mathit{SearchingTimes} < \mathit{Accumulator}\\&\qquad\qquad \le \mathit{ExploringTimes}\end{aligned}$$

$$\text{then } GroupIndex[i] = w,$$
$$\text{where}$$
$$w = Random[1, PulationSize] \quad \text{and}$$
$$w \notin FrequentItemSet[i];$$
$$FrequentItemSet[i] = Random[FrequentPool];$$
$$i = 1, 2, \ldots, M_k;\ M_k = M_{\min}, M_{\min+1}, \ldots, M_{\max}, \tag{25}$$

where *ExploringTimes* is a predefined value that judge to perform exploring action.

Step 4 After selecting M_k group indexes, the k chromosomes are selected from M_k group as follows:

$$ChromosomeIndex[i] = q,$$
$$\text{where } q = Random[1, N_c];$$
$$i = 1, 2, \ldots, k, \tag{26}$$

where N_c represents the number of chromosomes in each group; *ChromosomeIndex*$[i]$ represents the index of a chromosome that select from ith group.

The illustration of the DMSS is shown in Fig. 6 with a simple description as follows:

Suppose the TSSO determines that 4 rules are expected, and 3 out of 7 groups, group 2, 3 and 6, are deemed as frequent groups. If the current action type of the DMSS is normal action, then 4 random groups will be selected to form a TFS. If the search action is taken, then frequent group 2, 3 and 6 will be selected. The remaining one group will be draw randomly from group 1, 4, 5 and 7. If the explore action is taken, then the 4 non-frequent group 1, 4, 5 and 7 will be selected in case of the problem of local optimum.

3.4 Fitness assignment step

The fitness value of a rule (an individual) is calculated by concatenating this individual with elites of other groups selected by DMSS. The details for assigning the fitness value are described as follows:

Denote $G_1, G_2, \ldots, G_{Mk}$, the M_k groups selected by the DMSS; $G_j \cdot p_i$ denotes the i-th individual of the j-th group; y_j refers to the elite individual of the j-th group. Then the fitness of the individual $G_j \cdot p_i$ can be computed as follows:

$$\begin{aligned} &fitness(G_j \cdot p_i) \\ &\quad = fitness(G_1 \cdot y_1, \ldots, G_j \cdot p_i, G_{j+1} \cdot y_{j+1}, \\ &\qquad \ldots, G_{M_k} \cdot y_{M_k}). \end{aligned} \tag{27}$$

3.5 Elites-based reproduction strategy (ERS)

Reproduction is a process in which individual strings are copied according to their fitness value. A fitness value is assigned to each chromosome in each group according to a fitness assignment method in which high numbers denote a good fit. The goal of the SOME is to maximize the fitness value. For keeping the stable of the algorithm, this study proposes an elite-based reproduction strategy (ERS) to let the best combination of chromosomes in each group can be kept in the next generation. In the SOME, the chromosome that has best fitness value may not be the chromosome in the best combination. About this, in the ERS, every chromosome in the best combination of M_k groups must be kept by performing reproduction step. In the remaining chromosomes in each group, this study uses the roulette-wheel selection method for this reproduction process. The best performing chromosomes in the top half of each group [12] advance to the next generation. The other half is generated to perform crossover and mutation operations on chromosomes in the top half of the parent generation. In the reproduction step, the top half of the population for each group must be kept the same number of chromosomes.

3.6 The data mining based crossover strategy (DMCS)

Although the ERS operation can search for the best existing individuals, it does not create any new individuals. In nature, an offspring has two parents and inherits genes from both. The main operator working on the parents is the crossover operator, the operation of which occurs for a selected pair with a crossover rate. In this paper, the data mining based crossover strategy (DMCS) is proposed to perform the crossover operation. The DMCS mimics the cooperation phenomenon in society [42, 43], in which individuals become more suitable to the environment as they acquire and share more knowledge of their surroundings. In the DMCS, as same with the DMSS, the DMCS uses FP-Growth to select the parental group indexes to perform crossover operation in the next generation. Moreover, the DMCS also proposed threes actions to select parental group indexes according to the frequent item set. The best performing individuals in the top half of selected parental group indexes that are called elites are used to select the parents for performing with the DMCS. Details of the DMCS are shown below.

Step 1 The first one of the parents that is used to perform the crossover operation is selected from the original group by using the following equations:

$$Fitness\_Ratio_{g,t} = \frac{\sum_{u=1}^{t} fitness_{g,u}}{\sum_{c=1}^{Nc} fitness_{g,c}},$$
$$\text{where } t = 1, 2, \ldots, Nc; \tag{28}$$

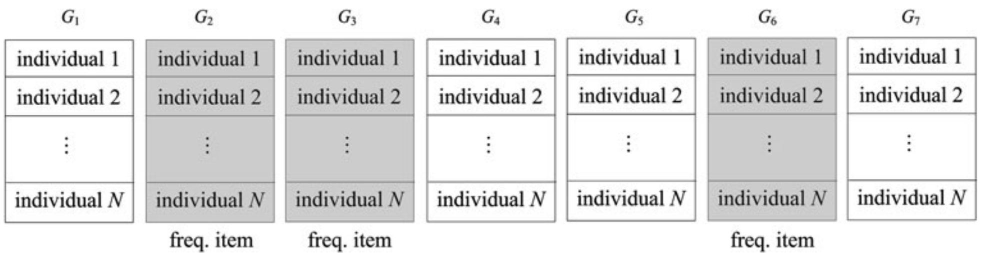


Fig. 6 The example of the DMSS

$$Rand\_Value[g] = Random[0,1],$$
$$\text{where } g = 1, 2, \ldots, PulationSize; \quad (29)$$

$$Parent\_SiteA[g] = t,$$
$$\text{if } Fitness\_Ratio_{g,t-1} < Rand\_Value[g] \le Fitness\_Ratio_{g,t}, \quad (30)$$

where $Fitness\_Ratio_{g,t}$ is a fitness ratio of the fitness value of tth chromosome in the gth group; $Rand\_Value[g] \in [0,1]$ is the random values of gth group; $Parent\_siteA[g]$ is the site where the first parent is. According (30), if the $Rand\_Value[g]$ is greater than the fitness ratio at $(t-1)$th chromosome in gth group and smaller or equal to the fitness ratio at tth chromosome in gth group, the site of the first parent of gth group is assigned to t.

Step 2 After determining the first parent, the second parental group index is decided according to different actions that describe as follows:

(a) In the normal action, the best performing elites in each group is used to determine the other parent. In this step, the total fitness ratio of every group is computed according to the following equations:

$$Total\_Fitness_g = \sum_{c=1}^{Nc} fitness_{g,c},$$
$$\text{where } g = 1, 2, \ldots, PulationSize; \quad (31)$$

$$Total\_Fitness\_Ratio_w = \frac{\sum_{u=1}^{w} Total\_Fitness_u}{\sum_{g=1}^{PulationSize} Total\_Fitness_g},$$
$$\text{where } w = 1, 2, \ldots, PulationSize; \quad (32)$$

where $Total\_Fitness_g$ represents the summation of the fitness value of every chromosomes in gth group; $Total\_Fitness\_Ratio_w$ is a total fitness ratio of wth group. After computing the total fitness ratio, the group where the chromosome is selected from to be the other parent for performing crossover with the $Parent\_SiteA[g]$-th chromosome in gth group is determined according to the following equations:

$$Group\_Rand\_Value[g] = Random[0,1],$$
$$\text{where } g = 1, 2, \ldots, M; \quad (33)$$

$$Parent\_Group\_SiteB[g] = w,$$
$$\text{if } Total\_Fitness\_Ratio_{w-1} < Group\_Rand\_Value[g] \le Total\_Fitness\_Ratio_w, \quad (34)$$

where $Group\_Rand\_Value[g] \in [0,1]$ is a random values of gth group; $Parent\_Group\_SiteB[g]$ represents the site of the group that the second parent is selected from.

(b) In the searching action, the second parent is decided according to the following equations:

$$FrequentItemSet[q] = Random[FrequentPool]$$
$$\text{where } q = 1, 2, \ldots, FrequentPoolNum; \quad (35)$$

$$Parent\_Group\_SiteB[g] = w,$$
$$\text{if } w \in FrequentItemSet[q]. \quad (36)$$

(c) In the exploring action, the second parent is decided according to the following equations:

$$FrequentItemSet[q] = Random[FrequentPool]$$
$$\text{where } q = 1, 2, \ldots, FrequentPoolNum; \quad (37)$$

$$Parent\_Group\_SiteB[g] = w,$$
$$\text{if } w \notin FrequentItemSet[q]. \quad (38)$$

Step 3 After the $Parent\_Group\_SiteB[g]$-th group is selected, the other parents in the selected $Parent\_Group\_SiteB[g]$-th group according to the following equations:

$$Fitness\_Ratio_{Selected\_g,t} = \frac{\sum_{u=1}^{t} fitness_{Selected\_g,u}}{\sum_{c=1}^{Nc} fitness_{Selected\_g,c}},$$

where $t = 1, 2, \ldots, Nc$;

$$Selected\_g = Parent\_Group\_SiteB[g]; \tag{39}$$

$$Rand\_Value[g] = Random[0, 1],$$

$$\text{where } g = 1, 2, \ldots, PulationSize; \tag{40}$$

$$Parent\_SiteB[g] = l,$$

$$\text{if } Fitness\_Ratio_{Selected\_g,l-1} < Rand\_Value[g] \le Fitness\_Ratio_{Selected\_g,l}, \tag{41}$$

where $Fitness\_Ratio_{Select\_g,t}$ is a fitness ratio of the fitness value of tth chromosome in the $Parent\_Group\_SiteB[g]$-th group; and $Parent\_SiteB[g]$ is the site where the second parent is.

After the DMCS selects the parents form the gth group and $Parent\_Group\_SiteB[g]$-th group, the individuals $Parent\_SiteA[g]$-th chromosome and the $Parent\_SiteB[g]$-th chromosome) are crossed and separated using a two-point crossover in the gth group. In two-point crossover, exchanging the site's values between the selected sites of parents' individual create new individuals.

Here we use Fig. 6 as the DMSS use for convenience to illustrate the DMCS. A simple example of the DMCS is described as follows:

Suppose the group 1 is scheduled to perform crossover step; group 2, 3 and 6, are deemed as frequent groups. At first, it randomly draws a chromosome from group 1 as the 1st parent chromosome of crossover. Second, the DMCS is used to determine the group that 2nd parent chromosome should be selected from based on three different actions. If the action type of the DMCS is normal action, the other group will be selected randomly from group 2 to 7. If the search action is taken, then the other group will be selected from frequent group 2, 3 and 6. If the explore action is taken, then one, beside the group 1 itself, of the non-frequent group 4, 5 and 7 will be selected. Finally, it randomly draws a chromosome from the group that DMCS determines as the 2nd parent chromosome, and applies crossover operation on it with the 1st parent chromosome.

3.7 Self-organization mutation strategy (SOMS)

Mutation is an operator that randomly alters the allele of a gene. Recently, the sequential-search based dynamic evolution (SSDE) [19, 33, 34] is widely used in mutation operator. Therefore, better mutation points will be determined to perform dynamic-mutation. To make sure the best combination of chromosomes will be mutated at the next generation, the proposed SOME adopts the SSDE method [19] to generate the new chromosomes. The search algorithm of SSDE is similar to the local search procedure in [35]. In SSDE, every gene in the previous best combination of chromosomes is selected by using a sequential search and the gene's value is updated to evaluate the performance based on the fitness value. Although the SSDE can obtain the better mutation point, the population may fall in local optimum solutions and may not easily to skip. This is because that the SSDE changes the gene according to the performance of the chromosomes, the closer performance of chromosomes leads to the fewer updating value of gene. For avoiding the issue of local optimum, the self-organization mutation strategy (SOMS) is proposed here. In the SOMS, if the SSDE cannot improve the performance of the best combination of chromosomes, the traditional mutation [32] takes over. The details of the SOMS method are listed as follows:

Step 1 Sequentially search for a gene in the best combination of previous chromosomes.

Step 2 Update the chosen gene in step 1 according to the following formula:

$$Chr_{g,best\_c}[p] = \begin{cases} Chr_{g,best\_c}[p] + \Delta(fitness\_value, m_{\max} - Chr_{g,best\_c}[p]), & \text{if } \alpha > 0.5 \\ Chr_{g,best\_c}[p] - \Delta(fitness\_value, Chr_{g,best\_c}[p] - m_{\min}), & \text{if } \alpha < 0.5 \end{cases} \tag{42}$$

where$p = 2, 4, 6, \ldots, 2n$;

$$Chr_{g,best\_c}[p] = \begin{cases} Chr_{g,best\_c}[p] + \Delta(fitness\_value, \delta_{\max} - Chr_{g,best\_c}[p]), & \text{if } \alpha > 0.5 \\ Chr_{g,best\_c}[p] - \Delta(fitness\_value, Chr_{g,best\_c}[p] - \delta_{\min}), & \text{if } \alpha < 0.5 \end{cases}$$

$$\text{where } p = 1, 3, 5, \ldots, 2n-1; \tag{43}$$

$$Chr_{g,best\_c}[p] = \begin{cases} Chr_{g,best\_c}[p] + \Delta(fitness\_value, w_{\max} - Chr_{g,best\_c}[p]), & \text{if } \alpha > 0.5 \\ Chr_{g,best\_c}[p] - \Delta(fitness\_value, Chr_{g,best\_c}[p] - w_{\min}), & \text{if } \alpha < 0.5 \end{cases}$$

$$\text{where } p = 2n+1, 2n+2, \ldots, 2n+(n+1); \tag{44}$$

$$\Delta(fitness\_value, v) = v\lambda(1/fitness\_value)^{\lambda}, \tag{45}$$

where $\alpha, \lambda \in [0, 1]$ are the random values; *fitness_value* is the fitness value; *Best_C* represents the *Best_C*th chromosome in the gth group of the best combination of chromosomes and p represents the gene in a chromosome; i and j represent the ith input dimension and jth rule.

Step 3 If the new gene that is generated from step 2 can improve the fitness value, then replace the old gene with the new gene in the chromosome. If not, recover the old gene in the chromosome. After this, go to step 1 until every gene is selected.

Step 4 If the genes check form step 1 to step 3 cannot improve the performance of the best combination of previous chromosomes, the algorithm adopts the traditional mutation operation [32] to mutate the chromosomes each group. In the following simulations, a mutation rate is set to 0.3 for performing the traditional mutation. The aforementioned steps are done repeatedly and stopped when the predetermined condition is achieved.

4 Illustrative examples

In two different simulations, the proposed SOME is used to train a TFM. The first simulation performs an approximation problem that is described in [36]. The second simulation is carried out to control a water bath temperature control system that is described in [34] and [37]. For the two computer simulations, the initial parameters are given in Table 4 before training. The initial parameters are determined by practical experimentation or trial-and-error tests.

4.1 Example 1: approximation of nonlinear functions

This example considers a nonlinear system presented by Sugeno et al. in [36],

$$z = (1 + x_1^{-2} + x_2^{-1.5})^2, \quad 1 \le x_1, x_2 \le 5 \tag{46}$$

The fuzzy network identification was based on 50 samples reported in [36]. The original samples were four inputs and one output with two dummy inputs. Figure 7 show a three-dimension input-output graph of this system. The TFM discarded the two dummy inputs using the proposed SOME for training. The parameters of this example are show in Table 5. These parameters are determined by practical experimentation or trial-and-error tests. The evolution processed for 200 generations and was repeated 50 times. Figure 8 shows the results of the probability vectors in the TSSO. In this figure, the final optima number of rules is 7.

Table 4 The initial parameters before training

Parameters	Value
Coding Type	Real number
PulationSize	16
N_C	10
Selection_Times	280
NormalTimes	10
SearchingTimes	20
ExploringTimes	30
Crossover Rate	0.5
Mutation Rate	0.3
$[M_{\max}, M_{\min}]$	[3, 10]

Table 5 The initial parameters number in the example 1

Parameters	Value
ThreadFitnessvalue	100
TSSATimes	30
$[m_{\min}, m_{\max}]$	[0, 5]
$[\sigma_{\min}, \sigma_{\max}]$	[0, 5]
$[w_{\min}, w_{\max}]$	[−10, 10]
Minimum_Support	*TransactionNum*/2

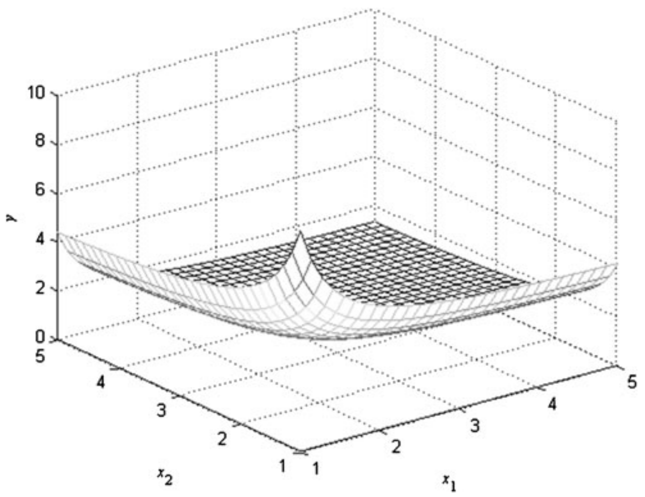


Fig. 7 Input-output relation of a nonlinear system

The obtained fuzzy rules of the TNFS using the SOME method are shown as follows:

R^1: If x_1 is $A_{1,1}(0.87, 0.68)$ and x_2 is $A_{2,1}(2.93, 1.16)$

Then $y' = 0.92 + 0.32x_1 + 6.74x_2$

R^2: If x_1 is $A_{1,2}(3.51, 2.37)$ and x_2 is $A_{2,2}(4.13, 1.04)$

Then $y' = 4.96 + 1.14x_1 + 4.12x_2$

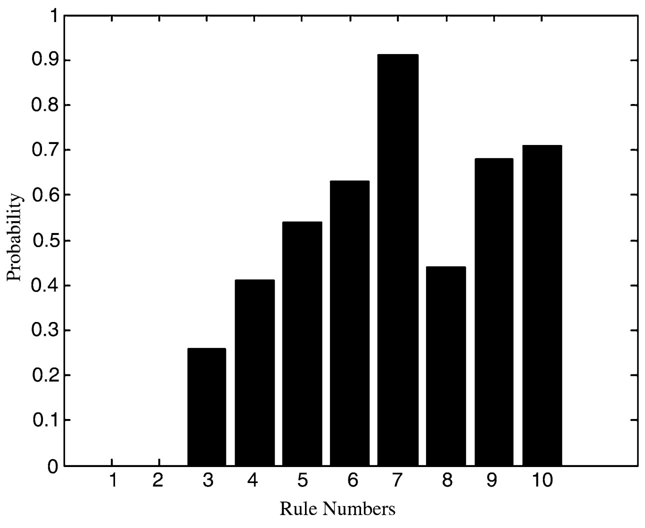


Fig. 8 The results of the probability vectors in the TSSO

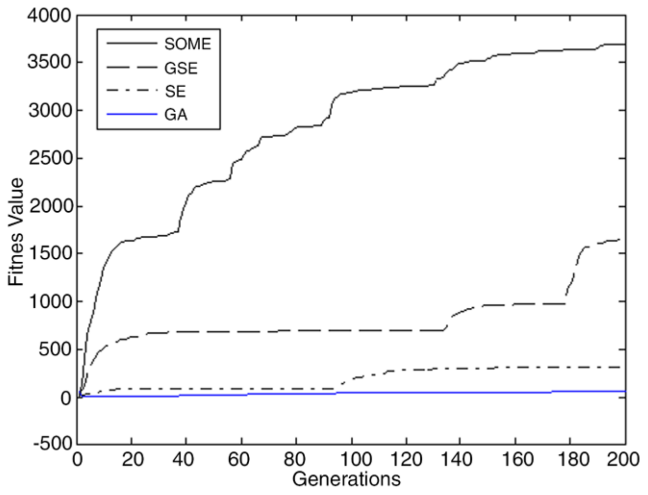


Fig. 9 The learning curves of the proposed SOME, GSE [20], SE [34] and GA [8]

R^3: If x_1 is $A_{1,3}(0.79, 0.62)$ and x_2 is $A_{2,3}(4.91, 2.63)$

Then $y' = 8.22 - 3.73x_1 + 1.97x_2$

R^4: If x_1 is $A_{1,4}(1.72, 0.68)$ and x_2 is $A_{2,4}(2.16, 0.94)$

Then $y' = -0.57 + 2.96x_1 - 0.43x_2$

R^5: If x_1 is $A_{1,5}(2.43, 4.81)$ and x_2 is $A_{2,5}(0.91, 3.46)$

Then $y' = 8.67 - 0.12x_1 - 4.97x_2$

R^6: If x_1 is $A_{1,6}(3.82, 4.21)$ and x_2 is $A_{2,6}(0.87, 1.78)$

Then $y' = -9.12 + 2.098x_1 + 8.43x_2$

R^7: If x_1 is $A_{1,7}(2.56, 4.67)$ and x_2 is $A_{2,7}(0.95, 2.21)$

Then $y' = 9.07 - 2.73x_1 + 0.78x_2$

The final mean square error (MSE) of the output approximates 0.00026. In this example, in order to show the effectiveness and efficiency of the proposed SOME method, the group-based symbiotic evolution (GSE) [20], the symbiotic evolution (SE) [34], and the genetic algorithm (GA) [8] were applied to design TFM by solving the same problem. In the GSE, the SE, and the GA, the population size and group size were set to 200 and the crossover and mutation probabilities were set to 0.5 and 0.3, respectively. There are seven rules that set to the GSE, SE, and the GA. The evolution processed for 200 generations and repeated for 50 times. Figure 9 shows the learning curves of the four models. As shown in Fig. 9, the proposed SOME method also yields better results.

Table 6 shows the results given in the relevant literature plus the error found using the SOME method. The performance of the very compact fuzzy network obtained by the proposed SOME method is better than all the previous works.

Table 6 Comparison results for Sugeno's nonlinear function approximation example

Author and reference	MSE
SOME	**0.00026**
GSE [20]	0.00063
GEFREX [41]	0.00078
SE [34]	0.0026
Lin [40]	0.005
Emami [38]	0.004
Sugeno [36]	0.01
Delgado [39]	0.231
GA [8]	0.019

To demonstrate the efficiency of the proposed TSSO, DMSS, DMCS, and SOMS methods, in this example the four different methods are used such as: the proposed SOME without TSSO, DMSS, and DMCS (Type I), the GSE method (Type II), the proposed SOME (Type III), the proposed SOME without SOMS (Type IV), and the GSE with the SSDE (Type V). In Type I method, the SOME method uses only the proposed SOMS method. In Type II method, the traditional GSE [20] is adopted. In Type III method, the SOME uses the proposed TSSO, DMSS, DMCS, and SOMS to perform structure and parameter learning. In Type IV, the SOME uses the proposed TSSO, DMSS, and DMCS methods. In Type V method, the GSE combines the SSDE to perform parameter learning. Table 7 shows the performance comparison of four methods. As shown in Table 7, the proposed SOME (Type III) obtains a better performance than other methods.

Table 7 Performance comparison of three different methods in example 1 (Time steps)

Method	MSE
Type I method (The proposed SOME without TSSO, DMSS and DMCS)	0.0005
Type II method (The GSE method)	0.00063
Type III method (The proposed SOME)	**0.00026**
Type IV method (The proposed SOME without SOMS)	0.00041
Type V method (The GSE with SSDE)	0.00054

Table 8 The initial parameters number in the example 2

Parameters	**Value**
ThreadFitnessvalue	50
TSSATimes	30
$[m_{\min}, m_{\max}]$	[0, 2]
$[\sigma_{\min}, \sigma_{\max}]$	[0, 2]
$[w_{\min}, w_{\max}]$	[−30, 30]
Minimum_Suppor	*TransactionNum*/2

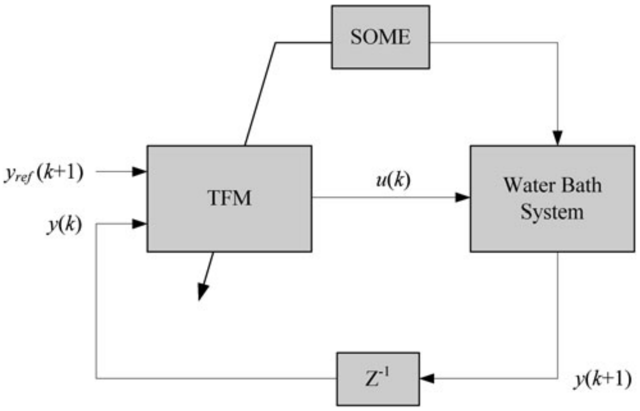


Fig. 10 Flow diagram of the TFM using the SOME for solving the temperature control problem

4.2 Example 2: water bath temperature control system

The goal of this simulation is to control the temperature of a water bath system given by

$$\frac{dy(t)}{dt} = \frac{u(t)}{C} + \frac{Y_0 - y(t)}{R_1 C} \tag{47}$$

where $y(t)$ is system output temperature in °C; $u(t)$ is heating flowing inward the system; Y_0 is room temperature; C is the equivalent system thermal capacity; and R_1 is the equivalent thermal resistance between the system borders and surroundings.

Assuming that R_1 and C are essentially constant, the system in (47) can be rewritten into discrete-time form with

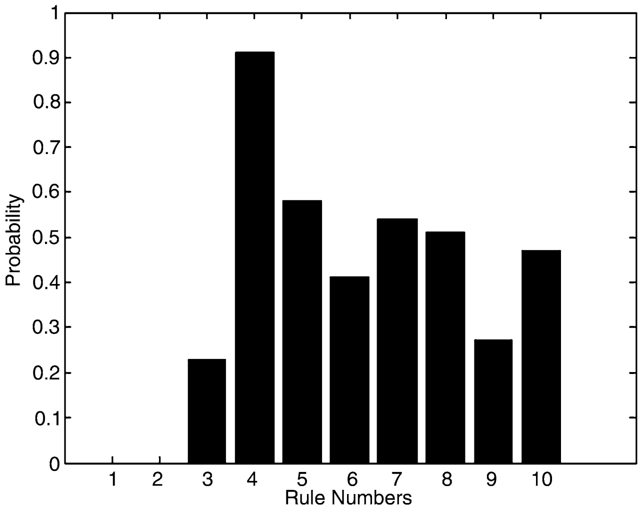


Fig. 11 The results of the probability vectors in the TSSO

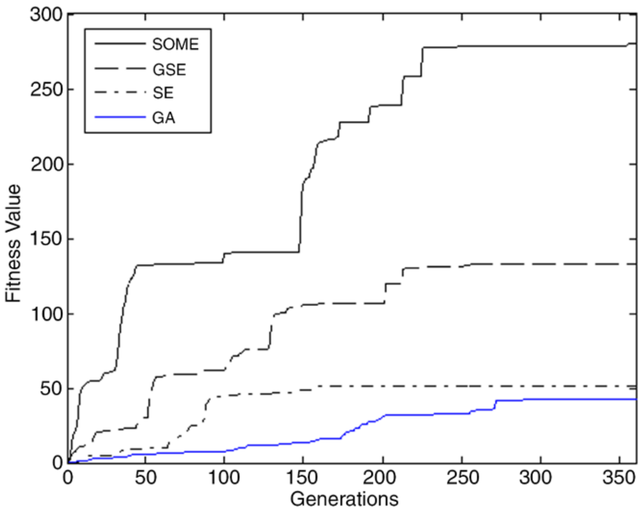
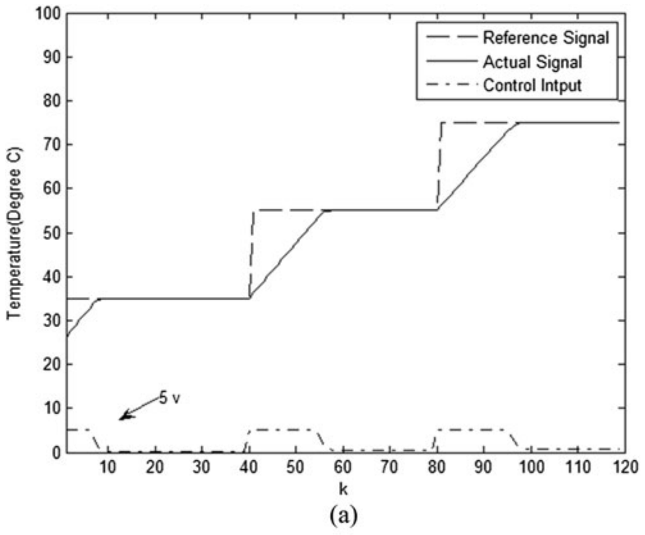


Fig. 12 The learning curves of the proposed SOME, GSE [20], SE [34] and GA [8]

some reasonable approximation. The system

$$y(k+1) = e^{-\alpha Ts} y(k) + \frac{\frac{\beta}{\alpha}(1 - e^{-\alpha Ts})}{1 + e^{0.5y(k)-40}} u(k) + \left[1 - e^{-\alpha Ts}\right] y_0 \tag{48}$$



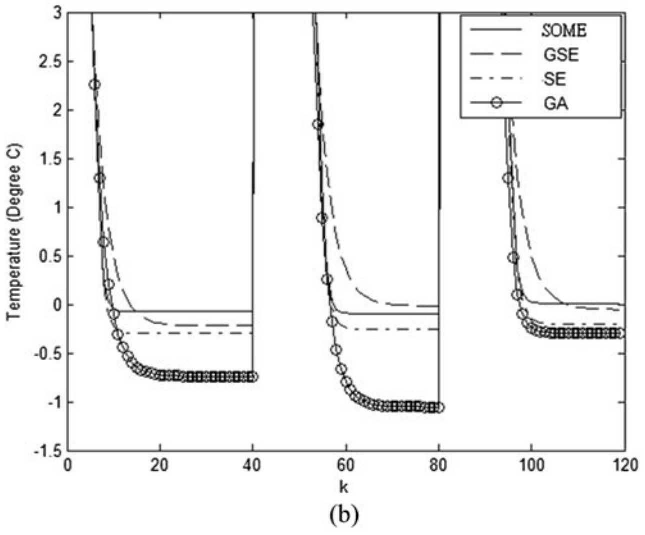
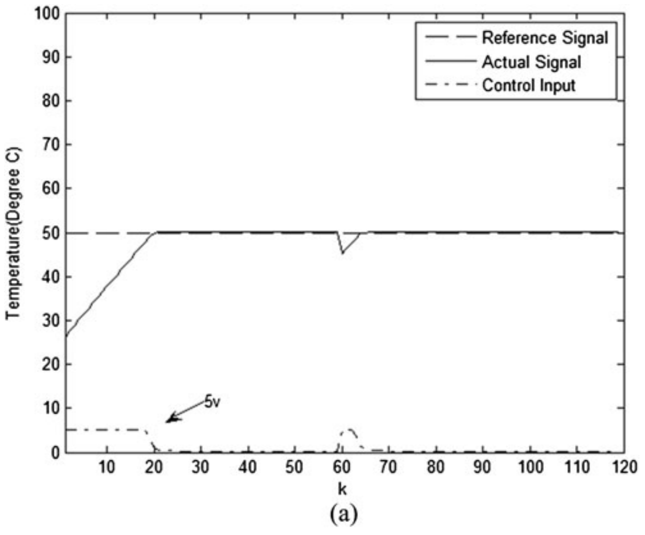


Fig. 13 **(a)** Final regulation performance of the TFM training by the proposed SOME for water bath system. **(b)** The error curves of the SOME, GSE [20], SE [34] and GA [8]



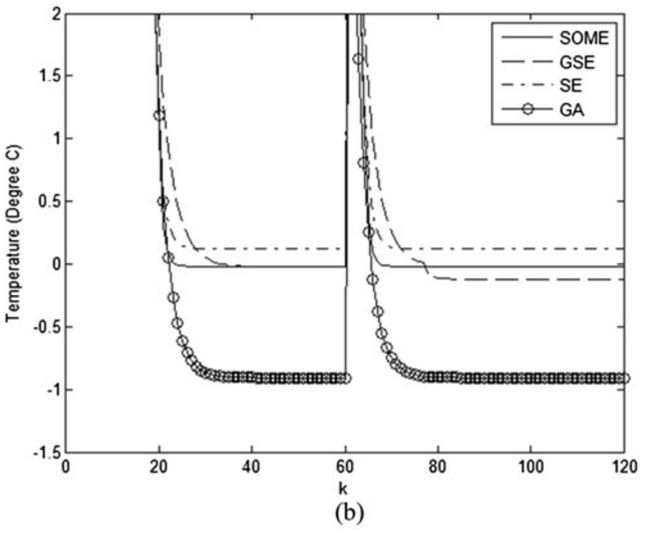


Fig. 14 **(a)** Behavior of the TFM training by the proposed SOME under the impulse noise for water bath system. **(b)** The error curves of the SOME, GSE [20], SE [34] and GA [8]

is obtained, where α and β are some constant values describing R and C. The system parameters used in this example are $\alpha = 1.0015e^{-4}$, $\beta = 8.67973e^{-3}$, and $Y_0 = 25.0$ (°C), which were obtained from a real water bath plant in [34] and [37]. The input $u(k)$ is limited to 0 and 5 represent voltage unit. The sampling period is $Ts = 30$. The system configuration is shown in Fig. 10, where y_{ref} is the desired temperature of the controlled plant.

In the TFM, a sequence of random input signals $u(k)$ limited to 0 and 5 V is injected directly into the simulated system described in (48). The 120 training patterns are chosen from the input-outputs characteristic in order to cover the entire reference output. The initial temperature of the water is 25°C, and the temperature rises progressively when random input signals are injected. The two input variables y_{ref} and $y(k)$ and the output $u(k)$ are normalized between 0 and 1 over the following ranges, y_{ref} : [25, 85], $y(k)$: [25, 85], $u(k)$: [0, 5]. The parameters of this example are show in Table 8. These parameters are determined by practical experimentation or trial-and-error tests. The evolution processed for 360 generations and was repeated for 50 times. Figure 11 shows the results of the probability vectors in the TSSO. In this figure, the final optima number of rules is 4.

In this example, the SOME is also compared the performance with other methods (the GSE [20], the SE [34], and the GA [8]). In the [20, 34], and [8], the parameters are the same with example 1. There are four rules that set to the GSE, the SE, and the GA. The evolution processed for 360 generations and was repeated for 50 times. Figure shows the learning curves of the four models. In Fig. 12, the proposed SOME also obtains a better fitness value than

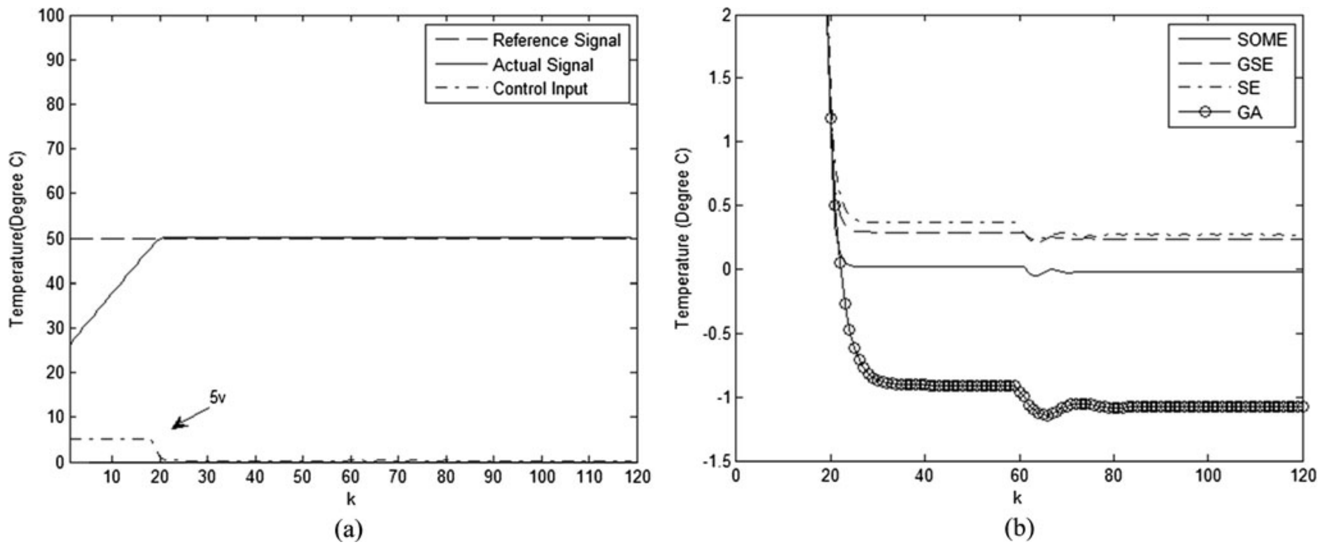


Fig. 15 **(a)** Behavior of the TFM training by the proposed SOME when a change occurs in the water bath system. **(b)** The error curves of the SOME, GSE [20], SE [34] and GA [8]

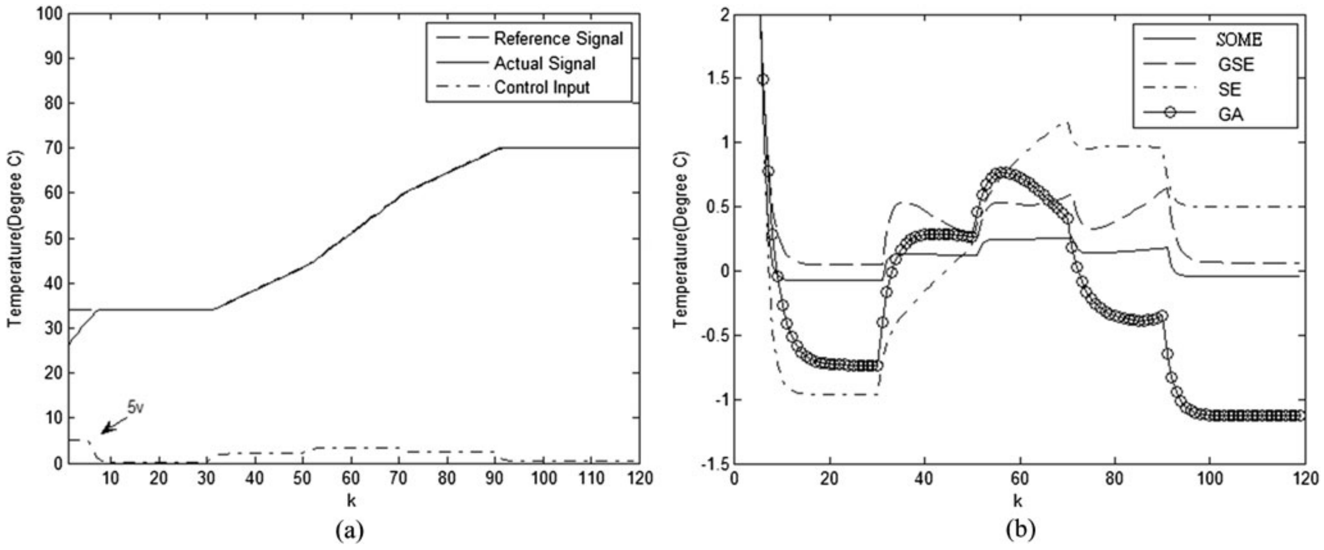


Fig. 16 **(a)** The tracking performance of the TFM training by the proposed SOME for the water bath system. **(b)** The error curves of the SOME, GSE [20], SE [34] and GA [8]

other models. To test the performance of the four models, the comparison performance measures include set-points regulation, the influence of impulse noise, and a large parameter variation in the system, and tracking capability of the controllers.

The first task is to control the simulated system to follow three set-points.

$$y_{ref}(k) = \begin{cases} 35°\text{C}, & \text{for } k \le 40 \\ 55°\text{C}, & \text{for } 40 < k \le 80 \\ 75°\text{C}, & \text{for } 80 < k \le 120. \end{cases} \tag{49}$$

The regulation performance of the SOME is shown in Fig. 13(a). In this paper, the regulation performance is also test by using the GSE [20], the SE [34], and the GA [8]. The error curves of the SOME, the GSE, the SE, and the GA are shown in Fig. 13(b). In this figure, the SOME obtains smaller errors than other controllers.

The second set of simulations is carried out for the purpose of studying the noise-rejection ability of the four models when some unknown impulse noise is imposed on the process. One impulse noise value −5°C is added to the plant output at the sixtieth sampling instant. A set-point of 50°C is performed in this set of simulations. The behaviors of the SOME under the influence of impulse noise and the corresponding errors of the SOME, the GSE, the SE, and the GA are shown in Fig. 14(a)–(b).

Table 9 Performance comparison of various controllers

	SOME	GSE [17]	SE [31]	GA [8]
Regulation Performance	**344.80**	357.43	365.20	378.02
Influence of Impulse Noise	**241.13**	252.67	258.80	262.77
Effect of Change in Plant Dynamics	**235.87**	240.84	244.14	280.38
Tracking Performance	**40.56**	58.17	87.18	100.22
Training Time (s)	**31.43**	52.2	153.59	122.17

One common characteristic of many industrial-control processes is that their parameters tend to change in an unpredictable way. To test the robustness of the four controllers, a value of $0.7 * u(k-2)$ is added to the plant input after the sixtieth sample in the fourth set of simulations. A set-point of 50°C is used in this set of simulations. The behaviors of the SOME when there is a change in the plant dynamics are shown in Fig. 15(a). The corresponding errors of the SOME, the GSE, the SE, and the GA are shown in Fig. 15(b).

In the final set of simulations, the tracking capability of the SOME with respect to ramp-reference signals is studied. The equation of this simulations are defined

$$y_{ref}(k) = \begin{cases} 34°\mathrm{C}, & \text{for } k \le 30 \\ (34 + 0.5 * (k-30))°\mathrm{C}, & \text{for } 30 < k \le 50 \\ (44 + 0.8 * (k-50))°\mathrm{C}, & \text{for } 50 < k \le 70 \\ (60 + 0.5 * (k-70))°\mathrm{C}, & \text{for } 70 < k \le 90 \\ 70°\mathrm{C}, & \text{for } 90 < k \le 120 \end{cases} \tag{50}$$

The tracking performance of the SOME is shown in Fig. 16(a). The corresponding errors the SOME, the GSE [20], the SE [34], and the GA [8] are shown in Fig. 16(b).

To test performances, a performance index, sum of absolute error (SAE), is defined by

$$\mathrm{SAE} = \sum_k \left| y_{ref}(k) - y(k) \right| \tag{51}$$

where $y_{ref}(k)$ and $y(k)$ are the reference output and the actual output of the simulated system, respectively. For the aforementioned simulation results, Table 9 has shown that the proposed SOME method has better performance than those of other methods.

5 Conclusion

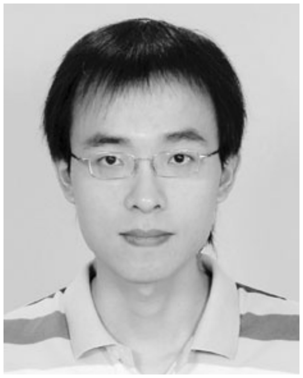
In this paper, a self-organization mining based hybrid evolution learning algorithm (SOME) for designing a TSK-type fuzzy model (TFM) is proposed. The proposed SOME has structure-and-parameter learning ability. That is, it can determine the suitable number of fuzzy rules and efficiently tune the free parameters in the TFM model. The proposed learning method also processes variable lengths of the combination of chromosomes in several groups. The advantages of the proposed SOME are summarized as follows: 1) the SOME used the GSE that each group represents only one fuzzy rule; 2) the TSSO is proposed to decide the suitable number of rules; 3) the SOME uses group-based population to evaluate the fuzzy rule locally; 4) the DMSS and DMCS are proposed to select the suitable combinations of individuals and parents; 5) it indeed can obtain better performs and converge more quickly than some genetic methods. Computer simulations have shown that the proposed SOME has a better performance than the other methods.

References

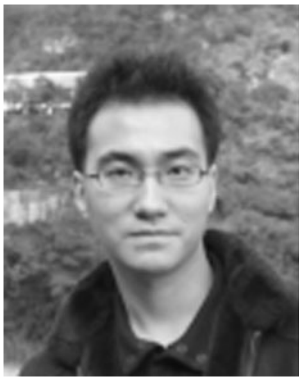
1. Lin CT, Lee CSG (1996) Neural fuzzy systems: A neural-fuzzy synergism to intelligent systems. Prentice Hall, Upper Saddle River
2. Jang J-SR, Sun C-T, Mizutani E (1997) Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Prentice Hall, Upper Saddle River
3. Takagi T, Sugeno M (1985) Fuzzy identification of systems and its applications to modeling and control. IEEE Trans Syst Man Cybern 15(1):116–132
4. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading
5. Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge
6. Fogel LJ (1994) Evolutionary programming in perspective: The top-down view. In: Zurada JM, Marks JM, Goldberg C (eds) Computational intelligence: imitating life. IEEE Press, New York
7. Rechenberg I (1994) Evolution strategy. In: Zurada JM, Marks JM, Goldberg C (eds) Computational intelligence: imitating life. IEEE Press, New York
8. Karr CL (1991) Design of an adaptive fuzzy logic controller using a genetic algorithm. In: Proceedings of the 4th international conference on genetic algorithms, pp 450–457
9. Homaifar A, Mccormick E (1995) Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. IEEE Trans Fuzzy Syst 3(2):129–139
10. Lee MA, Takagi H (1993) Integrating design stages of fuzzy systems using genetic algorithms. In: Proceedings of the IEEE international conference on fuzzy systems, pp 612–617
11. Juang CF (2002) A tsk-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. IEEE Trans Fuzzy Syst 10(2):155–170
12. Juang CF, Lin JY, Lin CT (2000) Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. IEEE Trans Syst Man Cybern, Part B, Cybern 30(2):290–302
13. Kumar P, Chandna VK, Thomas MS (2004) Fuzzy-genetic algorithm for pre-processing data at the rtu. IEEE Trans Power Syst 19(2):718–723
14. Alcalá R, Benítez JM, Casillas J, Cordón O, Pérez R (2003) Fuzzy control of HVAC systems optimized by genetic algorithms. Appl Intell 18(2):155–177
15. Alcalá R, Alcalá-Fdez J, Gacto MJ, Herrera F (2009) Improving fuzzy logic controllers obtained by experts: a case study in HVAC systems. Appl Intell 31(1):15–30
16. Kaya M, Alhajj R (2006) Utilizing genetic algorithms to optimize membership functions for fuzzy weighted association rules mining. Appl Intell 24(1):7–15
17. Bandyopadhyay S, Murthy CA, Pal SK (2000) VGA-classifier: design and applications. IEEE Trans Syst Man Cybern, Part B, Cybern 30(6):890–895

18. Carse B, Fogarty TC, Munro A (1996) Evolving fuzzy rule based controllers using genetic algorithms. Fuzzy Sets Syst 80(3):273–293
19. Lin CJ, Xu YJ (2005) Efficient reinforcement learning through dynamic symbiotic evolution for tsk-type fuzzy controller design. Int J Gen Syst 34(5):559–578
20. Lin CH, Xu YJ (2006) A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications. Fuzzy Sets Syst 157(8):1036–1056
21. Lin CJ, Hsu YC (2007) Reinforcement hybrid evolutionary learning for recurrent wavelet-based neurofuzzy systems. IEEE Trans Fuzzy Syst 15(4):729–745
22. Larose DT (2005) Discovering knowledge in data: an introduction to data mining. Wiley-Interscience, Hoboken
23. Fayyad U (1997) Data mining and knowledge discovery in databases: implications for scientific database. In: Proceedings of international conference on scientific and statistical database management, pp 2–11
24. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th international conference on very large data bases, pp 487–499
25. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. SIGMOD Rec 29(2):1–12
26. Hong TP, Kuo CS, Chi SC (1999) A data mining algorithm for transaction data with quantitative values. Intell Data Anal 3(5):363–376
27. Wu Y-T, An YJ, Geller J, Wu Y-T (2006) A data mining based genetic algorithm. In: Proceedings of the fourth IEEE workshop on software technologies for future embedded and ubiquitous systems, and the second international workshop on collaborative computing, integration, and assurance, pp 52–62
28. Tanese R (1989) Distributed genetic algorithms. In: Proceedings of the 3rd international conference on genetic algorithms, pp 434–439
29. Arabas J, Michalewicz Z, Mulawka J (1994) GAVAPS-a genetic algorithm with varying population size. In: Proceedings of the IEEE world congress on computational intelligence, pp 73–78
30. Moriarty DE, Miikkulainen R (1996) Efficient reinforcement learning through symbiotic evolution. Mach Learn 22(1–3):11–32
31. Smith RE, Forrest S, Perelson AS (1993) Searching for diverse, cooperative populations with genetic algorithms. Evol Comput 1(2):127–149
32. Cordon O, Herrera F, Hoffmann F, Magdlena L (2001) Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases. Advances in fuzzy systems—applications and theory, vol 19. World Scientific, Singapore
33. Lin CJ, Xu YJ (2006) The design of tsk-type fuzzy controllers using a new hybrid learning approach. Int J Adapt Control Signal Process 20(1):1–25
34. Lin CH, Xu YJ (2006) A hybrid evolutionary learning algorithm for tsk-type fuzzy model design. Math Comput Model 43(5–6):563–581
35. Jang J-SR, Sun C-T, Mizutani E (1997) Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Prentice Hall, Upper Saddle River, Chap 17
36. Sugeno M, Yasukawa T (1993) A fuzzy-logic-based approach to qualitative modeling. IEEE Trans Fuzzy Syst 1(1):7–31
37. Tanomaru J, Omatu S (1992) Process control by on-line trained neural controllers. IEEE Trans Ind Electron 39(6):511–521
38. Emami MR, Turksen IB, Goldenberg AA (1998) Development of a systematic methodology of fuzzy logic modeling. IEEE Trans Fuzzy Syst 6(3):346–361
39. Delgado M, GomezSkarmeta AF, Martin F (1997) A fuzzy clustering-based rapid prototyping for fuzzy rule-based modeling. IEEE Trans Fuzzy Syst 5(2):223–233
40. Lin YH, Cunningham GA, Coggeshall SV (1997) Using fuzzy partitions to create fuzzy systems from input-output data and set the initial weights in a fuzzy neural network. IEEE Trans Fuzzy Syst 5(4):614–621
41. Russo M (2000) Genetic fuzzy learning. IEEE Trans Evol Comput 4(3):259–273
42. Heppner F, Grenander U (1990) A stochastic nonlinear model for coordinated bird flocks. In: Krasner S (ed) The ubiquity of chaos. AAAS, Washington
43. Reynolds CW (1987) Flocks, herds and schools: a distributed behavioral model. Comput Graph 21(4):25–34

Sheng-Fuu Lin was born in Tainan, the Republic of China, in 1954. He received the B.S and M.S. degree in mathematics from National Normal University in 1976 and 1979, respectively, the M.S. degree in computer science from the University of Maryland in 1985, and the Ph.D. degree in electrical engineering from the University of Illinois, Champaign, in 1988. Since 1988, he has been on the faculty of the Department of Electrical Engineering at National Chiao Tung University, Hsinchu, Taiwan, where he is currently a professor. His research interests include fuzzy systems, genetic algorithms, neural networks automatic target recognition, scheduling, image processing, and image recognition.



Jyun-Wei Chang received the B.S. and M.S. degree in electronic engineering from National Kaohsiung University of Applied Sciences, Taiwan, R.O.C. in 2005 and 2007, respectively. He is currently pursuing the Ph.D. degree at the Department of electrical engineering from the National Chiao Tung University, Taiwan, R.O.C. His research interests include neural networks, fuzzy systems, and evolutional algorithms.



Yung-Chi Hsu received the B.S. degree in information management from Ming-Hsin University of Science and Technology, Taiwan, R.O.C., in 2002, the M.S. degree in computer science and information engineering from Chaoyang University of Technology, Taiwan, R.O.C., and the Ph.D. degree in electrical engineering from the National Chiao Tung University, Taiwan, R.O.C. He is currently a Post Doctoral research fellow at Graduate Institute of Network Learning Technology, National Central University, Taiwan, R.O.C. His research interests include artificial intelligence, data mining, personalization systems, and web-based learning.