Applied Soft Computing 11 (2011) 3247–3259

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Reinforcement evolutionary learning using data mining algorithm with TSK-type fuzzy controllers

Chi-Yao Hsu[a], Yung-Chi Hsu[b], Sheng-Fuu Lin[a,*]

[a] *Department of Electrical Engineering, National Chiao-Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, ROC*
[b] *Graduate Institute of Network Learning Technology, National Central University, 300 Jhongda Road, Jhongli City, Taoyuan County 32001, Taiwan, ROC*

ARTICLE INFO

Article history:
Received 24 November 2008
Received in revised form
12 November 2010
Accepted 14 December 2010
Available online 23 December 2010

Keywords:
Fuzzy system
Control
Symbiotic evolution
Reinforcement learning
Association rules
R-ELDMA

ABSTRACT

Reinforcement evolutionary learning using data mining algorithm (R-ELDMA) with a TSK-type fuzzy controller (TFC) for solving reinforcement control problems is proposed in this study. R-ELDMA aims to determine suitable rules in a TFC and identify suitable and unsuitable groups for chromosome selection. To this end, the proposed R-ELDMA entails both structure and parameter learning. In structure learning, the proposed R-ELDMA adopts our previous research – the self-adaptive method (SAM) – to determine the suitability of TFC models with different fuzzy rules. In parameter learning, the data-mining based selection strategy (DSS), which proposes association rules, is used. More specifically, DSS not only determines suitable groups for chromosomes selection but also identifies unsuitable groups to be avoided selecting chromosomes to construct a TFC. Illustrative examples are presented to show the performance and applicability of the proposed R-ELDMA.

Crown Copyright © 2010 Published by Elsevier B.V. All rights reserved.

1. Introduction

Parallel and global search techniques that can simultaneously evaluate many points in the search space, such as the genetic algorithms (GAs) [1], genetic programming [2], evolutionary programming [3], and evolution strategies [4], have recently become popular. These evolutionary algorithms are more inclined to converge toward the global solution. Thus, recently, several approaches tried to use evolutionary algorithms to find the global solutions, and they have been applied in training fuzzy models (evolutionary fuzzy models) [5,6].

The evolutionary fuzzy model can generate a fuzzy system automatically using evolutionary algorithms, such as genetic algorithms (GAs). More recently, several genetic fuzzy models, which utilize GAs to generate the fuzzy models, have been proposed [7–9]. For instance, Karr applied GAs [7] to design and identify the membership functions of a fuzzy controller. Lin and Jou [8] applied GAs in reinforcement learning approaches. They proposed a GA-based fuzzy reinforcement learning approach to control magnetic bearing systems efficiently. Moreover, Juang et al. [9] applied an extended GA, named symbiotic evolution, which complements the local mapping property of a fuzzy rule, to reinforcement learning approaches. Their results indicated that the symbiotic evolution performs better than GAs in reinforcement learning.

Recently, several improved evolutionary algorithms have been proposed [6,10–15]. Most of them not only modify the structure of evolutionary algorithms to suit for generating fuzzy models but also improve the performance of each evolutionary procedure for generating fuzzy models efficiently. For instance, Bandyopadhyay et al. [10] used a variable-length genetic algorithm (VGA) that allows for different lengths of chromosomes in a population. The VGA is suitable for designing fuzzy models with different fuzzy rules. Carse et al. [11] used GA to evolve fuzzy rule-based controllers to improve the performance of traditional ones. Tang [12] proposed a hierarchical GA to enable the optimization of generated fuzzy models in particular applications. Moreover, Juang [6] combined online clustering and Q-value based GA in reinforcement fuzzy system design. His model, named CQGAF, is useful in designing the number of fuzzy rules and free parameters simultaneously in a fuzzy system. In addition, Gomez and Schmidhuber [13,14] proposed lots of work to solve these problems. Their proposed enforced sub-populations (ESP), which used sub-populations of neurons for fitness evaluation and overall control, to generated neural networks. As shown in [13,14], subpopulations are useful in evaluating solutions locally, and their proposed system has better performance than systems that use only one population. More recently, Lin and Hsu proposed a hybrid evolutionary learning algorithm (HELA) [15], which consists of the compact GA (CGA) and the

* Corresponding author.
E-mail address: sflin@mail.nctu.edu.tw (S.-F. Lin).

1568-4946/$ – see front matter. Crown Copyright © 2010 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.12.027

modified VGA to perform structure/parameter learning in dynamic construction of networks.

Although aforementioned algorithms [6,10–15] can improve traditional evolutionary algorithms, they may still have one or more of the following limitations: 1) all the fuzzy rules are encoded into one chromosome, 2) the numbers of fuzzy rules have to be assigned in advance, and 3) the population cannot evaluate each fuzzy rule locally.

To this end, this study tends to address the issues that mention above. Thus, reinforcement evolutionary learning using data mining algorithm (R-ELDMA) with a TSK-type fuzzy controller (TFC) is proposed. More specifically, a reinforcement signal is used as a fitness function for the R-ELDMA, that is, the R-ELDMA formulates the number of time steps before failure occurs and uses it as a fitness function. By the way, the R-ELDMA can evaluate the candidate solutions for the parameters of the TFC. Similar with [15], R-ELDMA entails both structure learning and parameter learning. However, in [15], all the fuzzy rules are encoded into one chromosome could cause the problem that a population cannot evaluate each fuzzy rule locally. To address this issue, R-ELDMA proposes a novel structure and parameter learning algorithms.

In structure learning, the proposed R-ELDMA not only determines the number of fuzzy rules automatically but also processes the variable combination of chromosomes. To realize this, a multi groups' symbiotic evolution (MGSE) is proposed. The MGSE is different from traditional GAs because each chromosome in the MGSE represents one rule in a fuzzy system. Moreover, similar with [13,14], MGSE is different from traditional symbiotic evolution because each group in MGSE represents a collection of only one fuzzy rule. The MGSE uses a multi groups' population to evaluate fuzzy rules locally. To compare with [13,14], the self-adaptive method (SAM) [16], which was proposed in our recent study, is used to determine the suitable number of rules in TFC. More specifically, the SAM uses two steps to determine the suitable number of rules to prevent these from falling in a local optimal solution. In other words, SAM is similar to the maturing phenomenon in society, where individuals learn to adapt to society as they acquire knowledge.

In addition, the parameter learning investigates the selection of suitable groups for TFC construct. This type of learning is useful to let groups for cooperating to generate better solutions.

To address this issue, the data mining is taken into account because it is useful in discovering hidden relations, patterns, and interdependencies. Recently, data mining has become a popular research topic [17,18]. It is the method of mining information from a database called "transactions". Data mining can be regarded as a new way of performing data analysis. One aim of data mining is to find association rules among sets of items that occur frequently in transactions. To achieve this aim, several methods have been proposed [19–21]. Agrawal and Srikant [19] proposed a mining method that ascertains large sets of items to find the association rules in transactions. In [20], association rules were used to discover meaningful associations between features that co-occur frequently in data sets. Hsu et al. [21] proposed an association-based GA approach, such that the GA is able to continue its searching tasks more efficiently.

Association rules can identify meaningful relationships between items that co-occur frequently in data sets; hence, achieving the aim of parameter learning is useful. Thus, the data-mining based selection strategy (DSS), which applies association rules, is proposed to determine which groups are suitable for TFC construction. Moreover, DSS identifies groups that should be taken into account to avoid selecting in building TFC models. To this end, the well-known association rules approach – explored by a priori algorithm [19] – is adopted. The a priori algorithm, including its variations and extensions, is widely accepted in discovery of association rules from transactions. By applying this algorithm, we can identify the suitable or unsuitable groups in transactions. In brief, the DSS contributes in identifying a suitable combination of individuals and avoiding the selection of unsuitable combinations of individuals.

The aims of R-ELDMA are summarized as follows: 1) SAM is applied to determine the suitable number of fuzzy rules in TFC models, 2) MGSE is used to evaluate the fuzzy rule locally; and 3) the DSS is used not only to select suitable groups but also to identify unsuitable groups for performing selection steps. With these aims, R-ELDMA contributes to the identification of near-optimal solutions and the reduction in the number of evolutionary generations.

This paper is organized as follows. In Section 2, a TFC is introduced. The proposed ELDMA is described in Section 3. In Section 4, reinforcement learning for ELDMA is discussed. In Section 5, simulation results are presented. The conclusions are given in the last section.

2. Structure of TSK-type fuzzy controller (TFC)

A TFC [22] uses different implication and aggregation methods from the standard Mamdani fuzzy model. Instead of using fuzzy sets, the conclusion part of a rule is a linear combination of the crisp inputs. The TSK-type fuzzy rule is shown is represented in Eq. (1), where n is the number of the input dimensions and j is the serial number of fuzzy rules.

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

The structure of a TFC is shown in Fig. 1. It is a five-layer network structure. In TFC, the firing strength of a fuzzy rule is calculated by performing the following "AND" operation on the truth values of each variable to its corresponding fuzzy sets by:

$$u_{ij}^{(3)} = \prod_{i=1}^{n} \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right) \tag{2}$$

where $u_i^{(1)} = x_i$ and $u_{ij}^{(3)}$ are the output of first and third layers respectively, and m_{ij} and σ_{ij} are the center and the width of the Gaussian membership function of the jth term of the ith input variable x_i, respectively. In this paper, we have used the Gaussian membership function because it can be a universal approximator of any nonlinear functions on a compact set [23].

The output of a fuzzy system is computed by:

$$y = u^{(5)} = \frac{\sum_{j=1}^{M} u_j^{(4)}}{\sum_{j=1}^{M} u_j^{(3)}} = \frac{\sum_{j=1}^{M} u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i)}{\sum_{j=1}^{M} u_j^{(3)}} \tag{3}$$

where $u^{(5)}$ is the output of 5th layer; w_{ij} is the weighting value with ith dimension and jth rule node; M is the number of fuzzy rule.

3. Evolutionary learning using data mining algorithm (ELDMA)

The proposed ELDMA aims to improve the symbiotic evolution [24] and determine the suitable fuzzy rules of TFC automatically. Moreover, ELDMA also determine the suitable individuals used to construct a TFC. Thus, the ELDMA entails both structure learning and parameter learning.

In structure learning, the ELDMA is used to determine the number of fuzzy rules automatically. To achieve this aim, ELDMA processes the variable length of a combination of chromosomes. The length of a combination of chromosomes denotes the rule sets that are used to construct a TFC model. To deal with this, this paper

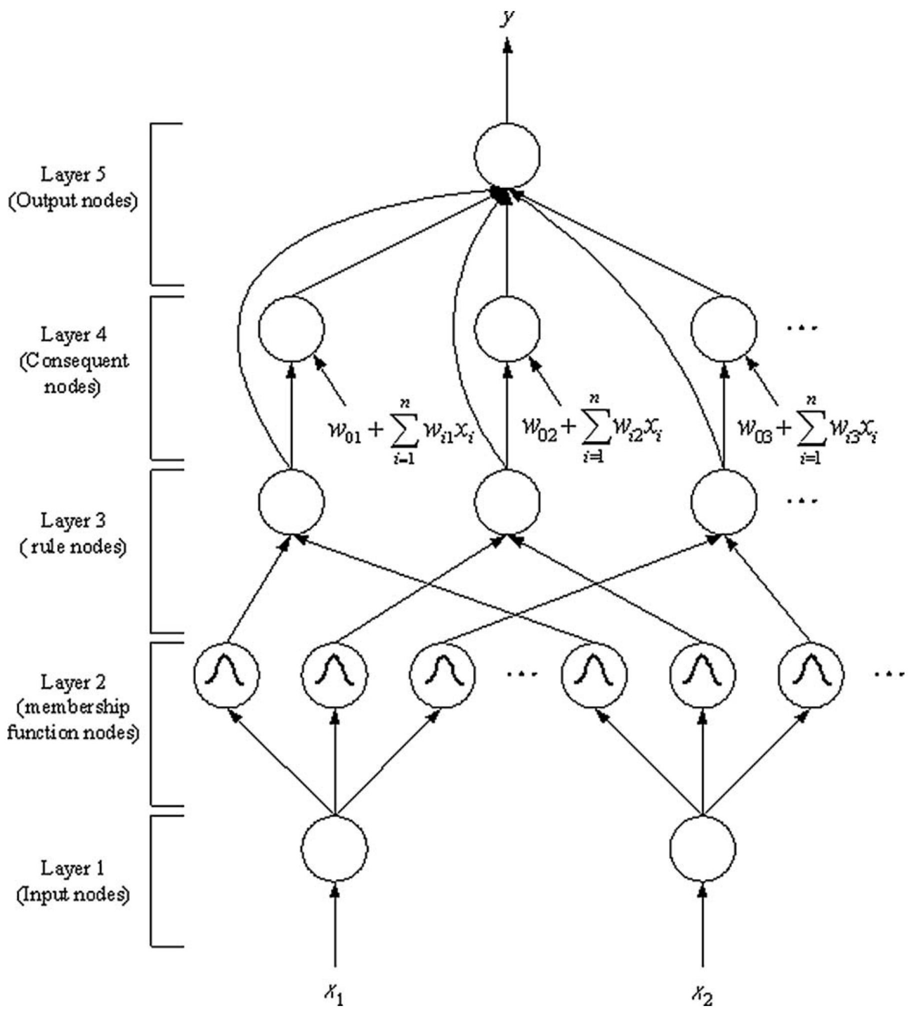

y
Layer 5 (Output nodes)
Layer 4 (Consequent nodes)
Layer 3 (rule nodes)
Layer 2 (membership function nodes)
Layer 1 (Input nodes)
$w_{01}+\sum_{i=1}^{n} w_{i1}x_i$
$w_{02}+\sum_{i=1}^{n} w_{i2}x_i$
$w_{03}+\sum_{i=1}^{n} w_{i3}x_i$
x_1
x_2

Fig. 1. The structure of the TSK-type neuro-fuzzy system.

proposes MGSE, which is developed from symbiotic evolution. The idea of symbiotic evolution was first proposed in an implicit fitness-sharing algorithm used in an immune system model [25]. More specifically, artificial antibodies were developed to identify artificial antigens. Each antibody can match only one antigen; hence, a different population of antibodies is required to effectively defense against a variety of antigens. As shown in [24,25], partial solutions can be characterized as specializations. Specialization ensures diversity, preventing a population from converging into suboptimal solutions. This implies that a single partial solution cannot "take over" a population because other specializations are present. Unlike the standard evolutionary approach, which always causes a given population to converge—hopefully at the global optimum, but often at a local one, the symbiotic evolution finds solutions in different, unconverted populations [24,25]. Moreover, unlike the traditional symbiotic evolutions, each population in the MGSE is divided into several groups, and each group represents a set of chromosomes that belongs to one fuzzy rule.

In ELDMA, the population is structured, such that each group represents a set of chromosomes that belongs to a fuzzy rule. Moreover, the number of rules in TFC models is variable. The structure of a chromosome in the ELDMA is shown in Fig. 2. As shown in the figure, each rule represents a chromosome selected from a group, P_{size} denotes that there are P_{size} groups in a population, and M_k means that there are M_k rules used in TFC construction.

In the traditional evolution algorithms, e.g., [6,10–14], the number of fuzzy rules needs to be assigned in advance. To solve this problem, our recent research model, the self-adaptive method (SAM), is applied. The SAM used the building blocks (BBs) not only to represent the suitability of TFC models with different fuzzy rules but also to determine the selection number of such models. In Fig. 3, SAM codes the probability vector V_{M_k}, representing the suitability of a TFC with M_k rules, into the building blocks (BBs). Furthermore, in SAM, the minimum and maximum number of rules must be predefined to limit the number of fuzzy rules to a certain bound, i.e., $[M_{\min}, M_{\max}]$.

In parameter learning, although SAM can determine the suitable number of rules, there is a need to identify the suitable groups used to select individuals to construct TFC models. Moreover, the unsuitable groups should avoid choosing to build the TFC models. More specifically, we should consider the well-performing groups of individuals to cooperate for producing better a generation than the current one. Conversely, groups of individuals with bad performance should avoid cooperating. To face these issues, this study proposes DSS not only to determine which groups should be used to select individuals but also to identify which groups should avoid choosing to construct a TFC.

The DSS involves two major parts, namely, finding frequent patterns and searching association rules. Regarding the former, the FP-growth algorithm [26] is used to find the frequent patterns that do not have candidate generation. Regarding latter, the a priori algorithm is used to identify association rules. In DSS, the FP-growth is used to find the sets of groups that occur frequently from transactions. In this paper, a "transaction" refers to the collection of groups

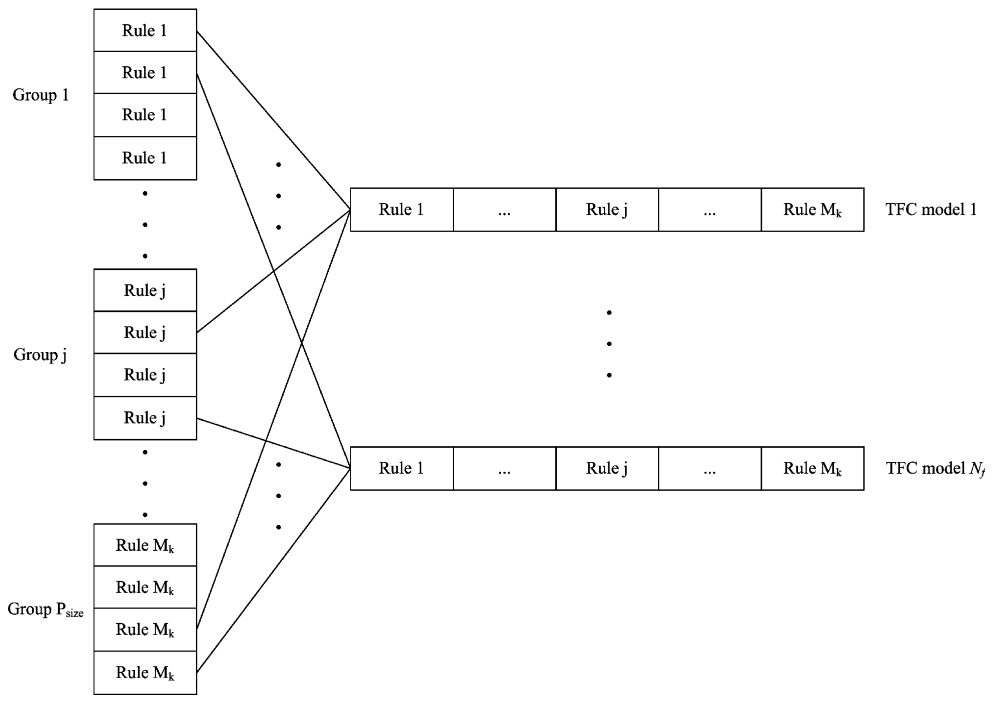


Fig. 2. The structure of the chromosome in the ELDMA.

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

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

that have good or bad performance. After the candidate sets of frequently occurring groups are found, DSS identifies the association rules using the a priori algorithm and uses the found association rules to determine M_k groups that are used to select M_k chromosomes that form TFC models with M_k rules. To this end, two actions are defined in this study: normal and search actions. In the normal action, M_k groups are chosen randomly. In the search action, M_k groups are chosen according to association rules.

The coding structure of chromosomes in our proposed ELDMA is shown in Fig. 4. The figure describes a fuzzy rule that has the form of Eq. (1), where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation, respectively, and w_{jM_k} is the weight with *i*th dimension and *j*th rule node.

The learning process of the ELDMA in each group involves four major operators: SAM, DSS, crossover strategy, and mutation strategy. The learning process stops as the system sustains successful status for a predefined time steps. The whole learning process is described below:

a. Review of Self-adaptive method (SAM):

The purpose of SAM is to determine the suitable selection times of each number of rules. The "selection times" indicates how many TFC models should be produce in one generation. In other words, SAM is used to determine the number of TFC models with M_k rules in every generation. After SAM is performed, the selection times of the suitable number of rules in a TFC increase; moreover, the selection times of the unsuitable number of rules in a TFC decrease.

m_{1j}	σ_{1j}	m_{2j}	σ_{2j}	$\cdots$	m_{nj}	σ_{nj}	w_{j1}	w_{j2}	$\cdots$	w_{jM_k}

Fig. 4. Coding a rule of a TNFS into a chromosome in the ELDMA.

The processing steps of the SAM are briefly reviewed below:

Step 0. Initialize the probability vectors of the BBs:

$$V_{M_k} = 0.5, \quad \text{for } M_k = M_{\min}, M_{\min+1}, \cdots, M_{\max}; \tag{4}$$

$$\textit{and Accumulator} = 0. \tag{5}$$

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

$$\begin{cases} V_{M_k} = V_{M_k} + (Upt\_value_{M_k} * \lambda), & \text{if } Avg \le fit_{M_k} \\ V_{M_k} = V_{M_k} - (Upt\_value_{M_k} * \lambda), & \text{otherwise} \end{cases} \tag{6}$$

$$Avg = \sum_{M_k=M_{\min}}^{M_{\max}} fit_{M_k} / (M_{\max} - M_{\min}); \tag{7}$$

$$Upt\_value_{M_k} = fit_{M_k} / \sum_{M_k=M_{\min}}^{M_{\max}} fit_{M_k}; \tag{8}$$

$$\begin{aligned} &\text{if} \quad Fitness_{M_k} \ge (Best\_Fitness_{M_k} - ThreadFitnessvalue) \\ &\text{then} \quad fit_{M_k} = fit_{M_k} + Fitness_{M_k}, \end{aligned} \tag{9}$$

where V_{M_k} is the probability vector in the BBs, λ is a predefined threshold value, *Avg* represents the average fitness value in the whole population, $Best\_Fitness_{M_k}$ represents the best fitness value of TFC models with M_k rules, and fit_{M_k} is the sum of the fitness values of the TFC models with M_k rules. In Eq. (6), the conditions of "$fit_{M_k} \ge$ or $<$ *Avg*" that affect the suitability of TFC models with M_k rules should be increased or decreased.

Step 2. Determine the selection times of TFC models with different rules according to the probability vectors of the BBs as follows:

$$\begin{aligned} Rp_{M_k} &= (Selection\_Times) * (V_{M_k} / Total\_Velocy), \ \text{for } M_k \\ &= M_{\min}, M_{\min+1}, \cdots, M_{\max}; \end{aligned} \tag{10}$$

$$Total\_Velocy = \sum_{M_k=M_{\min}}^{M_{\max}} V_{M_k}, \tag{11}$$

where *Selection _ Times* represents the total selection times in each generation and Rp_{M_k} represents the selection times of TFC models with M_k rules in one generation.

Step 3. To prevent suitable selection times from falling into the local optimal solution, SAM uses two different actions to update V_{M_k}. These actions are defined according to the following equations:

$$\text{if}\quad Accumulator \le SAMTimes,\ \text{then do Steps 1 to 3} \tag{12}$$

$$\begin{aligned}\text{if}\quad Best\_Fitness_g &= Best\_Fitness,\ \text{then}\ Accumulator\\ &= Accumulator + 1;\end{aligned} \tag{13}$$

$$\begin{aligned}&\text{if}\quad Accumulator > SAMTimes,\ \text{then do Step 0 and}\\ &Accumulator = 0,\end{aligned} \tag{14}$$

where *SAMTimes* is a predefined value, *Best _ Fitness*$_g$ represents the best fitness value of the best combination of chromosomes in the gth generation, and *Best _ Fitness* represents the best fitness value of the best combination of chromosomes in the current generations. Eqs. (12)–(14) imply that if the best fitness value does not improve within a predetermined number of generations, the suitable selection times may fall into the local optimal solution. At this time, the processing procedure of SAM should return to Step 0 to initialize the BBs.

b. The data-mining based selection strategy (DSS):

After operating SAM, the selection times of the TFC models with different numbers of rules are determined. Thereafter, ELDMA performs the selection step, which involves the selection of groups and the selection of chromosomes. In selection of groups, this paper proposes DSS to determine the suitable groups for chromosomes selection to form a TFC.

In DSS, suitable groups are selected according to the groups, which conduct from association rules that indicate good performance. In contrast, unsuitable groups are avoided selecting according to the groups, which conduct from association rules that demonstrate bad performance. To achieve these aims, DSS utilizes the FP-growth [26] and the a priori algorithm. Regarding former, the FP-growth is used to identify frequently pattern. It was proposed by Han et al. [26], and it aims to find the frequently occurring patterns that do not have candidate generation. In the proposed DSS, the FP-growth is used to find the frequently occurring groups from transactions. To reiterate, a transaction refers to a set of the groups that have good or bad performance. Regarding latter, after the frequently occurring groups are found, DSS adopts the a priori algorithm to construct association rules. The a priori algorithm is the most well-known association rules algorithm and is useful in several fields [19]. After performing these two steps, the found association rules are utilized to selects M_k groups that are used to choose chromosomes to form TFC models with M_k rules. To prevent the selected groups from falling into the local optimal solution, DSS uses normal and search actions to select well-performed groups. The details of the DSS are discussed below:

Step 0. The transactions are built, as in the following equations:

$$\begin{aligned}&\text{if}\quad Fitness_{M_k} \ge (Best\_Fitness_{M_k} - ThreadFitnessvalue)\\ &Transaction_j[i] = TFCRuleSet_{M_k}[i]\\ &then\\ &Transaction_j[M_k+1] = g\\ &\text{where}\quad i = 1,\ 2,\ \cdots,\ M_k;\\ &M_k = M_{\min},\ M_{\min+1},\ \cdots,\ M_{\max};\\ &j = 1,\ 2,\ \cdots,\ TransactionNum,\end{aligned} \tag{15}$$

Table 1
Transactions in the DSS.

Transaction index	Groups
1	l, 4, 8, g
2	2, 4, 7, 10, b
...	...
TransactionNum	l, 3, 4, 6, 8, 9, g

$$\begin{aligned}&\text{if}\quad Fitness_{M_k} < (Best\_Fitness_{M_k} - ThreadFitnessvalue)\\ &Transaction_j[i] = TFCRuleSet_{M_k}[i]\\ &then\\ &Transaction_j[M_k+1] = b\\ &\text{where}\quad i = 1,\ 2,\ \cdots,\ M_k;\\ &M_k = M_{\min},\ M_{\min+1},\ \cdots,\ M_{\max};\\ &j = 1,\ 2,\ \cdots,\ TransactionNum,\end{aligned} \tag{16}$$

where the $Fitness_{M_k}$ represents the fitness value of TFC with M_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, $TFCRuleSet_{M_k}[i]$ represents the ith group in the M_k groups used for chromosomes selection, and $Transaction_j[M_k+1]=g$ and $Transaction_j[M_k+1]=b$ represent the last terms (“g” or “b”) that are inserted into the jth transaction. Such two terms represent good (“g”) and bad (“b”) performance, respectively. The transactions have the form shown in Table 1. From this table, to skip the last item of all transactions, every transaction represents the M_k groups that form a TFC with M_k rules. For example, as shown in Table 1, the first transaction of the transaction set means that the three-rule TFC formed by the first, fourth, and eighth groups have “good” performance. In contrast, the second transaction indicates that the four-rule TFC formed by the second, fourth, seventh, and the tenth groups have “bad” performance. The steps in building transactions continue with the normal and search actions.

Step 1. Normal action:

After the transactions are built, the DSS selects groups according to different action types. If the action type is normal, DSS selects the groups using the following equation:

$$\begin{aligned}&\text{if}\quad Accumulator \le NormalTimes\\ &\text{then}\quad GroupIndex[i] = Random[1,\ P_{Size}];\\ &\text{where}\quad i = 1,\ 2,\ \cdots,\ M_k; M_k = M_{\min},\ M_{\min+1},\ \cdots,\ M_{\max},\end{aligned} \tag{17}$$

where *Accumulator* defined in Eq. (12) is used to determine which action should be adopted, *GroupIndex*[*i*] represents the selected *i*th group of the M_k groups, and P_{Size} indicates that there are P_{Size} groups in a population in ELDMA. In this action, the algorithm is used to accumulate the transaction set. Therefore, the groups are stored in a transaction if the groups fit Eqs. (15) and (16). If the best fitness value does not improve for a sufficient number of generations (*NormalTimes*), then DSS selects the groups by switching to another action type (which go to the next steps).

Step 2. Finding association rules:

If the current action is the search action (the *Accumulator* exceeds the *NormalTimes*), DSS, which consists of FP-growth and a priori algorithm, is used to find the suitable or unsuitable groups in transactions. The details of the two major parts are presented below.

i. FP-growth

Frequently occurring groups are found according to the predefined *Minimum_Support*, which represents the minimum fraction of transactions that contain an item set. After *Minimum_Support* is defined, data mining using FP-growth is performed. The FP-growth algorithm can be viewed with two parts: construction of the FP-

Table 2
Sample transactions.

Transaction index	Groups
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 1-groupset of sample transactions.

Group name	Count	Group name	Count
B	4	M	3
C	5	O	3
F	4		

Table 4
F-list of sample transactions.

Group name	Count	Group name	Count
C	5	M	3
B	4	O	3
F	4		

Table 5
Transactions after discarding the infrequent groups and sorting the remaining groups in the same order as the F-list.

Transaction index	Groups	Ordered groups
1	{b, c. e, f, g, h, p}	{c, b, f}
2	{a. b. c, f. i, m, o}	{c, b, f, m, o}
3	{c, f, i, m, o}	{c, f, m, o}
4	{b, c, e, s, p}	(c. b)
5	{a, b, c, d. f, m, o}	{c, b, f, m, o}

tree and FP-growth. The sample transactions shown in Table 2 are given as examples. In this example, *Minimum_Support* = 3.

(1) FP-tree Construction:

The first step to construct a FP-tree is to scan the transactions and retrieve the frequent 1-groupset in transactions. The frequent 1-groupset represents the set with bigger support counts than *Minimum_Support* in transactions. The result is shown in Table 3. Then the retrieved frequently occurring groups are arranged in descending order based on their supports, as shown in Table 4. The ordered list in Table 4 is called the F-list. After the F-list is obtained, the next step is used not only to discard the infrequently occurring groups but also to sort the remaining groups in the same order as in the F-list in each transaction. The result is shown in Table 5. The ordered transactions are then used to construct the FP-tree. The steps in FP-tree construction are illustrated in Fig. 5(a). In the same figure,

Table 6
Frequently occurring groups generated by FP-growth data mining with Minimum_Support = 3.

Suffix group	Cond. group base	Cond. FP-tree	Frequent groups
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

formed by scanning the last transaction, the right-most chart is called the prefix-tree of the frequent 1-groupset. Each node of the prefix-tree is composed of one group, a count of the frequent 1-groupset, and a node frequently occurring group link. Thereafter, the completed FP-tree is created by combining the prefix-tree of the 1-groupset and the header-table. An example of an FP-tree is shown in Fig. 5(b). This FP-tree is constructed from the transactions shown in Table 2.

(2) FP-growth:

The FP-growth algorithm is done by following steps: construction of a conditional group base, construction of a corresponding conditional FP-tree, mining the frequently-occurring groups on the conditional FP-tree, and concatenation of the suffix group and the frequently-occurring groups on the conditional FP-tree.

First, we select each frequent 1-groupset as a suffix group, and find the corresponding set of paths connecting to the root of the FP-tree. The set of prefix paths is called the conditional group base. Second, we accumulate the count for each group in the base to construct the conditional FP-tree of the corresponding suffix group. Third, after mining the frequently occurring groups in the conditional FP-tree, FP-growth data mining is completed by the concatenation of the suffix group with the generated frequently occurring groups. Finally, the groups generated by the FP-growth, shown in Table 6, are then thrown into the pool called *FrequentPool*. The *FrequentPool* represents the candidate sets of the frequently occurring groups.

ii. Aproior algorithm

Once the frequently occurring groups are found, we further use these groups to generate association rules. Each association rule has a confidence and a support associated. The support of a rule refers to the fraction of transactions that contain the rule. The confidence of a rule is defined as the fraction of times in which if the antecedent is satisfied, the consequent is also true. The values of support and confidence are used to filter the set of association rules obtained from transactions. The task of the a priori algorithm can be formulated as finding all association rules with at least minimum support and minimum confidence, i.e., the thresholds of support and confidence. The association rules can be found using in three steps: 1) obtain the frequently occurring groups from FP-growth;

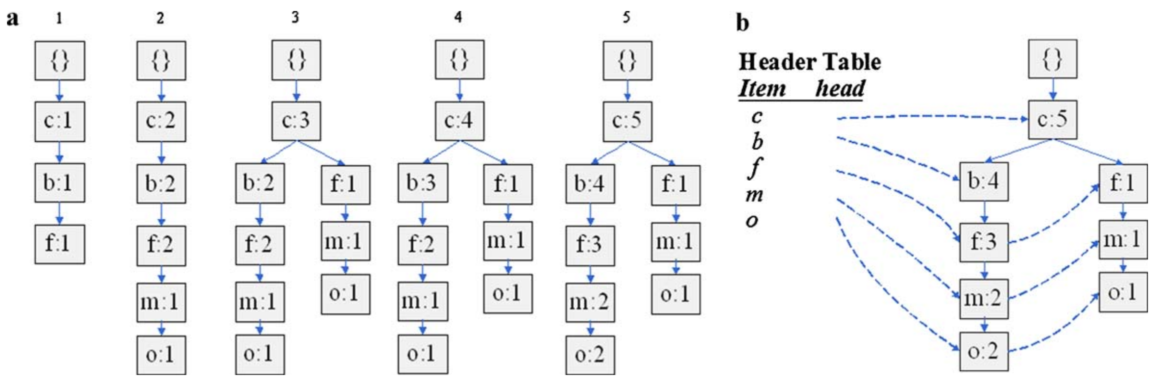


Fig. 5. Construct FP-tree. (a) Steps for constructing the FP-tree of sample transactions. (b) FP-tree of Table 5.

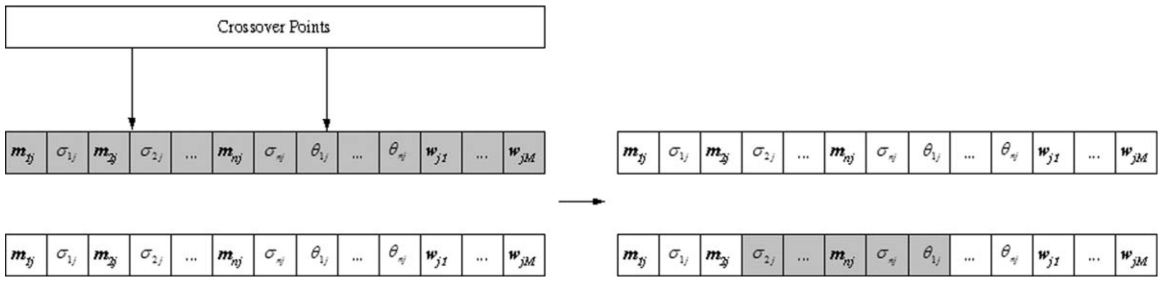

Crossover Points
m_{1j} σ_{1j} m_{2j} σ_{2j} … m_{nj} σ_{nj} θ_{1j} … θ_{nj} w_{j1} … w_{jM}

Fig. 6. Two-point crossover.

2) for each obtained group, find all nonempty subsets where each of them must contain “g” or “b” (in this study, “g” and “b” indicate target patterns); and 3) for each such subset of a group, if the confidence is bigger than the minimum confidence, producing the rule. For instance, if the confidence of $\{1,3,6\} \rightarrow \{g\}$ is bigger than the minimum confidence, then we construct this association rule. This rule indicates that the combination of the first, third, and sixth groups results in “good” performance. After doing so, the frequent patterns are conduct to the association rules.

Step 3. Select the groups according to association rules:

After the association rules are identified, DSS selects groups according to the association rules. More specifically, the selected groups, which are used to choose chromosomes, should mainly belong to the association rules with an inferred item with good performance (i.e., “g”). In contrast, the selected groups, which mainly belong to the association rules with an inferred item with bad performance (i.e., “b”), should be ignored. After doing so, DSS can not only determine which groups are suitable to construct TFC models but also identify groups that should be taken into account to avoid selecting to build the TFC.

If the best fitness value does not improve for a sufficient number of generations (*SearchingTimes*), DSS selects groups based on the normal action.

Step 4. After the M_k groups are selected, M_k chromosomes are selected from M_k groups as follows:

$$\begin{aligned}&\mathit{ChromosomeIndex}[i] = q,\\ &\text{where}\\ &q = \mathit{Random}[1,\ N_c];\\ &i = 1,\ 2,\ \cdots,\ k,\end{aligned} \tag{18}$$

where N_c represents the number of chromosomes in each group and *ChromosomeIndex*[i] represents the index of a chromosome that is selected from the ith group.

Fitness assignment notes:

As previously stated, for ELDMA, the fitness value of a rule (an individual) is calculated by summing up the fitness values of all possible combinations in the chromosomes that are selected from M_k groups that are decided by DSS. The steps in the fitness value assignment are described below:

Step 1. Choose M_k fuzzy rules to construct a TFC Rp_{M_k} times from M_k groups with size N_C. The M_k groups are obtained from the DSS.

Step 2. Evaluate every TFC that is generated from Step1 to obtain a fitness value.

Step 3. Divide the fitness value by M_k and accumulate the divided fitness value to the selected rules with their fitness value records that were set to zero initially.

Step 4. Divide the accumulated fitness value of each chromosome from M_k groups by the number of times that it has been selected. The average fitness value represents the performance of a rule.

c. Crossover strategy:

Although DSS can be used to select suitable individuals for TFC construction, it does not create any new individual. 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, a two-point crossover strategy [27] is adopted and shown in Fig. 6. In the figure, exchanging the site’s values between the selected sites of individual parents creates new individuals. The advantage of the two-point crossover is its ability of introducing a higher degree of randomness into the selection of genetic material [28]. Moreover, such crossover strategy generally yields better performance than one-point crossover due to its larger search step size [29].

d. Mutation strategy:

Although the crossover strategy produces many new strings, these strings do not provide any new information to every group at the site of an individual. Mutation can randomly alter the allele of a gene. In this paper, to emphasize the capability of the SAM and the DSS, the ELDMA attempts to simplify the mutation operation. Uniform mutation [27] is therefore adopted, and the mutated gene is drawn randomly from the domain of the corresponding variable. The benefits of uniform mutation are not only to generate new information into a population but also to keep a highly diverse array of information, which is useful to the fitness of individuals [30].

The aforementioned steps are performed repeatedly and stopped when the whole system sustains successful state for a predetermined time steps achieved.

4. A reinforcement learning for the ELDMA

Unlike the supervised learning problem, in which the correct “target” output values are given for each input pattern, the reinforcement learning problem has only very simple “evaluative” or “critical” information, rather than “instructive” information. In the extreme cases, there is only a single bit of information to indicate whether the output is right or wrong. The training environment of reinforcement ELDMA (R-ELDMA), which interacts with a reinforcement learning problems, is shown in Fig. 7. In this paper, the reinforcement signal indicates whether a success or a failure occurs.

As shown in Fig. 7, R-ELDMA consists of a TFC, which determines the proper action according to the current input vector (environment state). The structure of R-ELDMA is different from Barto and his colleagues’ actor-critic architecture [31], which consists of a control network and a critic network. The input of the TFC is the state of the plant, and the output is a control action of the state denoted by *f*. The only available feedback is a reinforcement signal that notifies the TFC only when failure occurs. An accumulator plays the role of a relative performance measure, and it is shown in Fig. 7. It accumulates the number of time steps before a failure occurs. In this paper, the feedback is decided by using an accumulator, which determines how long the experiment remains a “success”. In R-ELDMA, the accumulator is used as a relative measure of the fitness. In other words, the accumulator indicates the

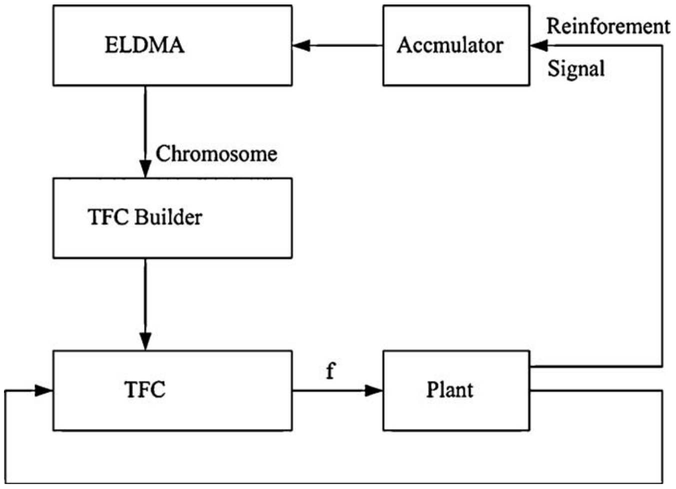


Fig. 7. Block diagram of the R-ELDMA for the TFC model.

"fitness" of the current TFC. The key of the R-ELDMA is formulating a number of time steps before a failure occurs and using this formulation as the fitness function of R-ELDMA. Its advantage is its ability to meet global optimization.

The flowchart of R-ELDMA is shown in Fig. 8. The R-ELDMA runs in a feed-forward fashion to control the environment (plant) until failure occurs. In this paper, fitness function is defined as the number of time steps before a failure occurs. The goal of R-ELDMA is to maximize the fitness value. The fitness function is defined by:

$$Fitness\,Value = TIME - STEP \tag{19}$$

where *TIME-STEP* represents how long the experiment remains a "success". Eq. (19) reflects the fact that the long-time steps before a failure occurs could implies higher fitness of the R-ELDMA. In other words, Eq. (19) is mainly used to evaluate the performance of a controller, which can keep the desired control goal.

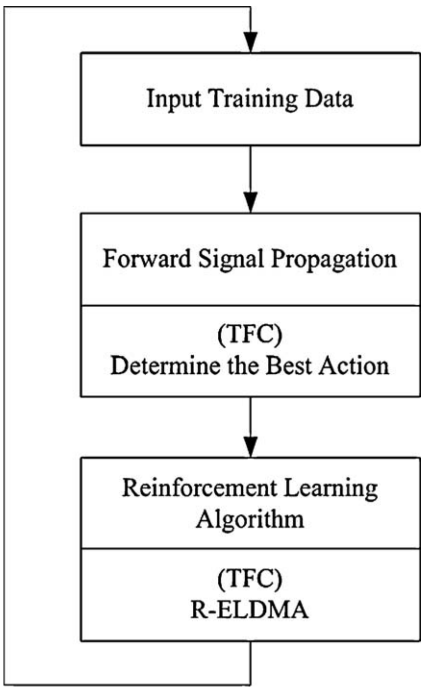

Input Training Data
Forward Signal Propagation
(TFC)
Determine the Best Action
Reinforcement Learning Algorithm
(TFC)
R-ELDMA

Fig. 8. Flowchart of the R-ELDMA.

Table 7
The initial parameters before training.

Parameters	Value	Parameters	Value
P_{size}	16	$[M_{min}, M_{max}]$	[3, 10]
N_c	20	$[m_{min}, m_{max}]$	[0, 1]
Selection_Times	250	$[\sigma_{min}, \sigma_{max}]$	[0, 1]
NormalTimes	20	$[w_{min}, w_{max}]$	[−15, 15]
SearchingTimes	30	*Mininmum_Support*	*TransactionNum*/2
Crossover Rate	0.4	*Mininmum_Confidence*	70%
Mutation Rate	0.3		

5. Illustrative examples

Two simulations are discussed in this section. The first is a cart pole balance system described in [15] and [32]. The second is a simulation to control a chaotic system [6,33]. For the two simulations, the initial parameters are given in Table 7. The initial parameters are determined by practical experimentation or trial-and-error tests.

Example 1: Control of a Cart-Pole Balancing System

In this simulation, R-ELDMA is applied to the classic control problem of the cart-pole balancing system. This system is often used as an example of inherently unstable and dynamic systems to demonstrate both of modern and classical control techniques [32], or the reinforcement learning schemes [15,31]. Here, it is used as a control benchmark. As shown in Fig. 9, the cart-pole balancing problem is the problem of learning how to balance an upright pole. The bottom of the pole is hinged to a cart that travels along a finite-length track. Both the cart and the pole can move only on the vertical plane; that is, each has only one degree of freedom.

There are four state variables in the system: θ, the angle of the pole in an upright position (in degrees); $\dot{\theta}$, the angular velocity of the pole (in degrees per second); x, the horizontal position of the cart's center (in meters); and $\dot{x}$, the velocity of the cart (in meters per second). The only control action is f, which is the amount of force (in Newtons) applied to the cart to move it to the left or right. The system fails when the pole falls past a certain angle (±12° is used) or when the cart runs into the bounds of its track (the distance is 2.4 m from the center to each bound of the track). The goal of this control problem is determining a sequence of forces applied to the cart to balance the pole upright. The equations of motion used are as follows:

$$\theta(t+1) = \theta(t) + \Delta\dot{\theta}(t), \tag{20}$$

$$\begin{aligned}\dot{\theta}(t+1) \quad &= \dot{\theta}(t) + \Delta\frac{(m+m_p)g\sin\theta(t)}{(4/3)(m+m_p)l - m_pl\cos^2\theta(t)} \\ &\quad - \frac{\cos\theta(t)\left[f(t) + m_pl\dot{\theta}(t)^2\sin\theta(t) - \mu_c sgn(\dot{x}(t))\right]}{(4/3)(m+m_p)l - m_pl\cos^2\theta(t)} \\ &\quad - \frac{(\mu_p(m+m_p)\dot{\theta}(t)/m_pl)}{(4/3)(m+m_p)l - m_pl\cos^2\theta(t)},\end{aligned} \tag{21}$$

$$x(t+1) = x(t) + \Delta\dot{x}(t), \tag{22}$$

$$\begin{aligned}x(t+1) \quad &= \dot{x}(t) + \Delta\frac{f(t) + m_pl[\dot{\theta}(t)^2\sin\theta(t) - \ddot{\theta}(t)\cos\theta(t)]}{(m+m_p)} \\ &\quad - \frac{\mu_c sgn(\dot{x}(t))}{(m+m_p)},\end{aligned} \tag{23}$$

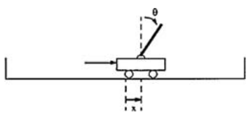


Fig. 9. The cart-pole balancing system.

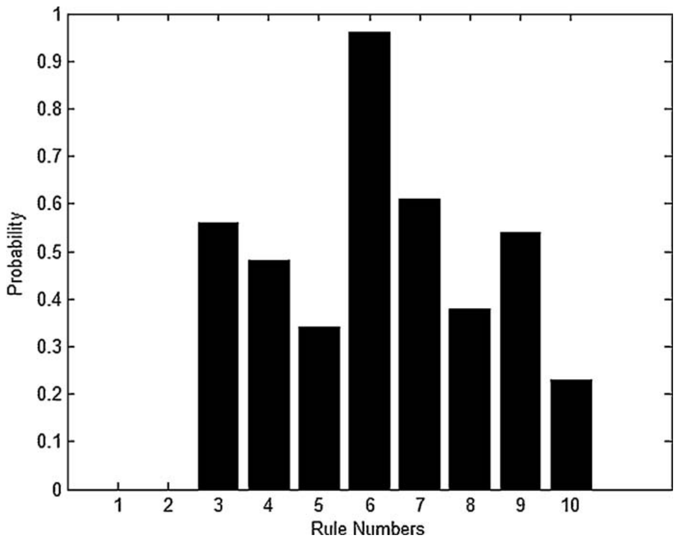


Fig. 10. The results of the probability vectors in the SAM.

where

$$
\begin{aligned}
&l && = 0.5\,\text{m},\ \text{the length of the pole;}\\
&m && = 1.1\,\text{kg},\ \text{combined mass of the pole and the cart;}\\
&m_p && = 0.1\,\text{kg},\ \text{mass of the pole;}\\
&g && = 9.8\,\text{m/s},\ \text{acceleration due to the gravity;}\\
&\mu_c && = 0.0005,\ \text{coefficient of friction of the cart on the track;}\\
&\mu_p && = 0.000002,\ \text{coefficient of friction of the pole on the cart;}\\
&\Delta && = 0.02\,(\text{s}),\ \text{sampling interval;}
\end{aligned}
\tag{24}
$$

The constraints on the variables are $-12^\circ \le \theta \le 12^\circ$, $-2.4\,\text{m} \le x \le 2.4\,\text{m}$, and $-10\,\text{N} \le f \le 10\,\text{N}$. A control strategy is deemed successful if it can balance the pole for 100,000 time steps. The four input variables (θ, $\dot{\theta}$, x, $\dot{x}$) and the output (f_t) are normalized between 0 and 1 over the following ranges, θ: [−12, 12], $\dot{\theta}$: [−60, 60], x: [−2.4, 2.4], $\dot{x}$: [−3, 3], f_t: [−10, 10]. The four normalized state variables are used as inputs to the proposed TFC model. The coding of a rule in a chromosome is given in Fig. 4. The values are floating-point numbers initially assigned using R-ELDMA. The fitness function used in this simulation to train the TFC model is defined in Eq. (19). This equation indicates how long before the cart-pole balancing system fails, after which it receives a penalty signal of −1. Failure in this simulation is defined as when the pole falls past a certain angle (±12° is used) and when the cart runs into the bounds of its track (the distance is 2.4 m from the center to each bound of the track). These conditions are defined based on the basic requirements of the cart-pole balancing problem having an angle θ in the range [−12, 12] and a position x in the range [−2.4, 2.4]. The *ThreadFitnessvalue* in this simulation is set to 350, and it is determined by practical experimentation or trial-and-error simulation tests.

A total of 15 runs are performed in this simulation. Each run starts at different initial states. Fig. 10 shows the results of one run of the probability vectors in the SAM. In this figure, the final optimal number of the rules is 6. Table 8 shows the mean, best, and worst of the optimal number of rules from 15 runs.

Table 8
The optimal number of runs from fifteen rims of the SAM.

Method	Mean	Best	Worst
R-ELDMA	6	3	10

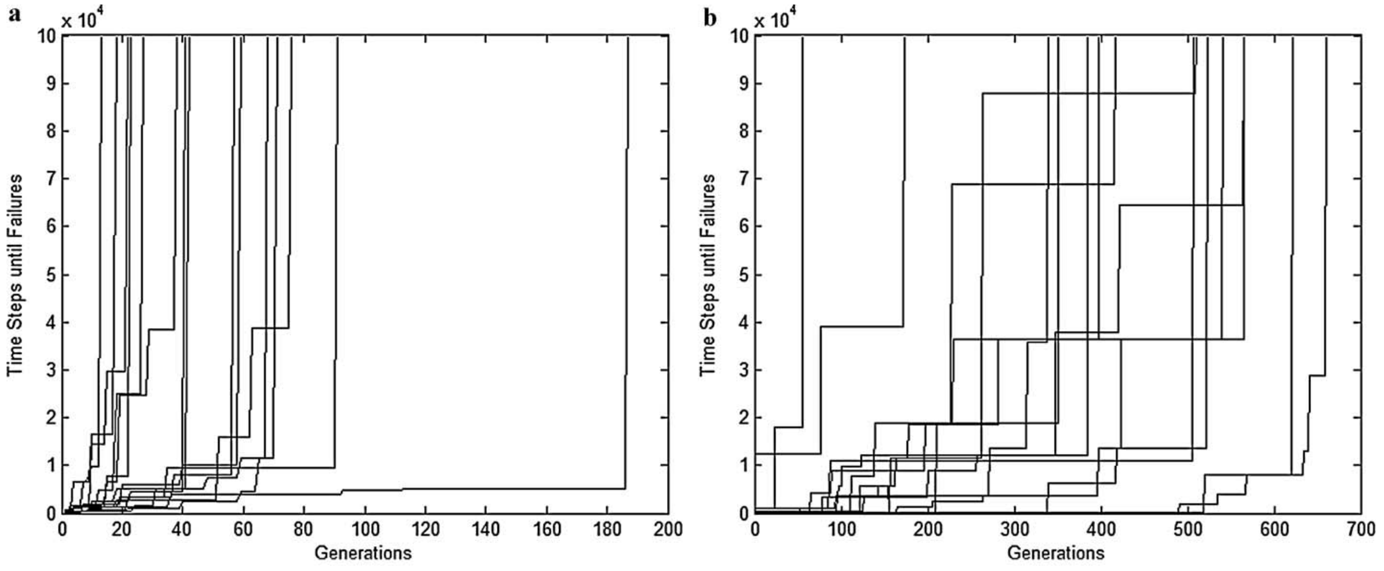
The learning curve of the proposed R-ELDMA after 15 runs is shown in Fig. 11(a). The learning curve represents how long before the cart-pole balancing system fails, after which it receives a penalty signal of −1. Therefore, each line in Fig. 11(a) represents each run in R-ELDMA. Each run indicates that the largest fitness value in the current generation is selected before the cart-pole balancing system fails. Thus, there are 15 lines in Fig. 11(a). As shown in this figure, the TFC model learns, on average, to balance the pole in the 67th generation. When R-ELDMA is stopped, the best combination of strings from the best groups in the final generation is selected and tested on the cart-pole balancing system.

The simulation is carried out for 15 runs. The successful results, including the pole angles, cart positions and controller outputs, are shown in Fig. 12. Each line in Fig. 12 represents each run with a different initial state. The results shown in this figure are the first 1000 time steps of 100,000 control time steps. As shown in Fig. 12, R-ELDMA successfully controls the cart-pole balancing system in 15 runs.

To show the efficiency of the proposed R-ELDMA method, the reinforcement symbiotic evolution (R-SE) [9] and the reinforcement genetic algorithm (R-GA) [7] are applied to the same problem. To compare these methods, a parameter exploration is used. The parameter exploration was firstly proposed by De Jong [34]. As shown in [34], a small population size is good for initial performance, and a large population size is good for long-term performance. Moreover, a low mutation rate is good for online performance, and a high mutation rate is good for offline performance. In [35], the best population size and mutation rate are 30 and 0.01, respectively. Parameters affecting the methods in this study are as follows: 1) the population size affects the final performance and the efficiency of GAs, 2) the crossover rate deals with the frequency at which the crossover step is applied, and 3) the mutation rate deals with the second search step, which increases the variability of the population.

In this experiment, five rules are set for R-SE and R-GA, and the parameters are found using the method given in [35]. Therefore, the population size has the range of 10–250 in increments of 10, the crossover rate has the range of 0.25–1 in increments of 0.05, and the mutation rate has the range of 0–0.3 in exponential increments. The results using the parameters set for R-SE and R-GA are as follows: 1) population sizes are 170, and 70, respectively; 2) the crossover rates are 0.55 and 0.6, respectively; 3) the mutation rates are 0.08 and 0.02, respectively. Fig. 11(b) and (c) shows that, on average, R-SE and R-GA methods learned to balance the pole in the 330th and 489th generations, respectively. The proposed R-ELDMA only compares the performance of the fitness value with the R-SE and R-GA. It is due to the fact that the reinforcement learning signal adopted in this study indicates that a good-performing controller is defined as a controller that does not exceed the predefined boundaries. As shown in Fig. 11, the control capabilities of R-ELDMA are better than those of [7] and [9] in the cart-pole balancing system.

Symbiotic adaptive neuro-evolution (SANE) [24], GENITOR [36], CQGAF [6], and R-HELA [15] have been applied to the same control problem. Their simulation results and a comparison on the number of pole-balance trials, reflecting the number of training episodes required, are shown in Table 9. The initial parameters of these methods [6,15,24,36] are determined according to [35]. After trial-and-error tests, in [36], the network consists of 20 hidden nodes. In [24], the network consists of 23 hidden nodes. The final average number of rules of 15 runs with [6] and [15] is 7 and 14, respectively. This paper also compares the CPU times with those of other existing methods [7,9,6,15,24,36]. The results are shown in Table 10. Clearly, the proposed R-ELDMA uses a shorter CPU time than other existing models. This experiment uses a Pen-



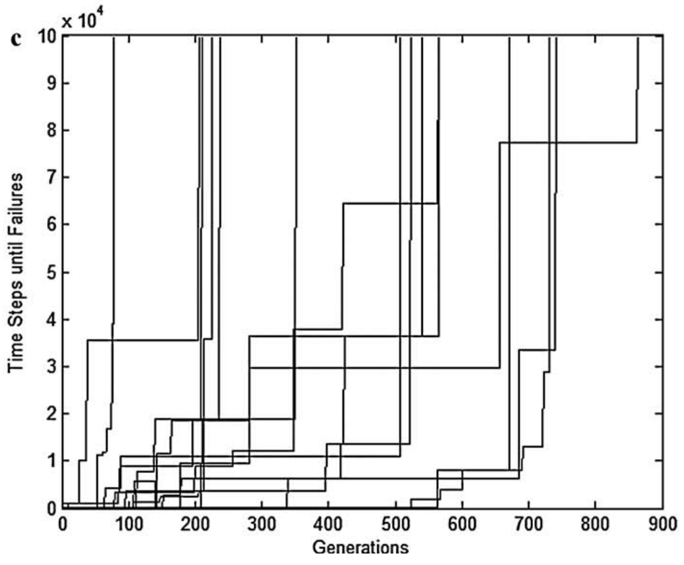


Fig. 11. The performance of (a) the R-ELDMA method, (b) the R-SE method [9], (c) the R-GA [7] method on the cart-pole balancing system.

tium 4 chip with a 1.5 GHz CPU, a 512 MB memory, and the visual C++ 6.0 simulation software. In short, the comparisons in Table 9 and Table 10 show that the proposed R-ELDMA method is feasible and effective.

In addition, because the proposed R-ELDMA method mainly involves four stages, i.e., SAM, DSS, crossover, and mutation, the CPU running time for each stage is also a concern in this example. Table 11 shows the CPU running time for each stage of the proposed method. From this table, DSS spends the most time compared with other stages. Hence, although the proposed DSS can enhance the learning effects, its processing time is excessively long. Thus,

Table 9
Comparison of time steps for various existing models.

Method	Mean	Best	Worst
GENITOR [36]	3645	423	6584
SANE [24]	1349	119	4847
R-GA [7]	489	78	863
R-SE [9]	330	56	660
R-HELA [15]	214	32	373
CQGAF [6]	159	26	294
R-ELDMA	67	13	187

Table 10
Comparison of CPU time for various existing models in Example 1.

Method	Mean (s)	Best (s)	Worst (s)
GENITOR [36]	143.74	83.61	387.03
SANE [24]	91.91	54.58	204.34
R-GA [7]	84.29	47.90	194.65
R-SE [9]	66.53	31.04	127.97
R-HELA [15]	52.37	24.19	106.87
CQGAF [6]	38.24	19.33	84.63
R-ELDMA	27.83	8.24	51.08

Table 11
The CPU running time for each stage of the proposed R-ELDMA in Example 1.

Stages	Mean (s)	Best (s)	Worst (s)
SAM	9.26	2.85	17.21
DSS	13.18	4.05	24.25
Crossover	3.41	0.95	5.36
Mutation	2.32	0.62	4.57

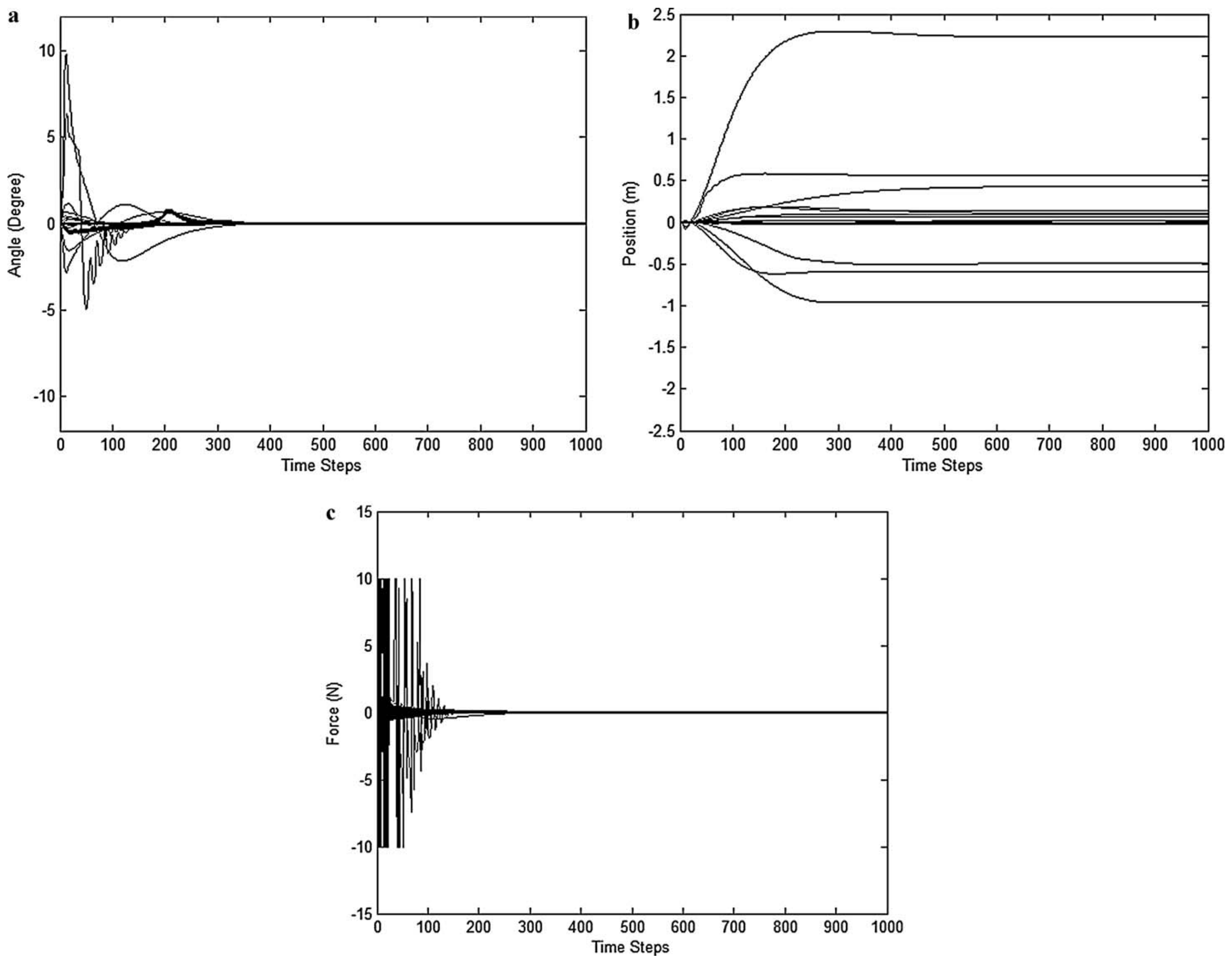


Fig. 12. Control results of the car-pole balancing system using the R-ELDMA in Example 1, (a) angle of the pole; (b) position of the cart; (c) control force.

reducing DSS computational time will be considered in our future work.

Example 2: Control of a Chaotic System

To demonstrate R-ELDMA, this example uses a multidimensional continuous state space. In this example, the proposed R-ELDMA is used to control the Mackey–Glass chaotic system. The Mackey–Glass chaotic system $x(t)$ is generated from the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) + u(t) \tag{25}$$

where $\tau > 17$. In this example, $x(0) = 0.7$, $\tau = 30$ and $\Delta t = 1$ are chosen. This means that the current state is influenced by the state of 30 time steps ahead. In [6], a feed-forward neural fuzzy network was used to predict the chaotic system when there is no control input, and nine succeeding system states are used as input data. In this example, as in [6], the nine succeeding system states were used as input to the TFC, and the reference state is set to $x_{\text{ref}} = 0.9$. The control of the Mackey–Glass chaotic system using the proposed R-ELDMA starts at 400 time steps. The control objective is to ensure that after 425 time steps, the state of the TFC is within $x_{\text{ref}} \pm 0.04$, and after 700 time steps, the state of the TFC model is within $x_{\text{ref}} \pm 0.02$, otherwise, failure occurs. The *ThreadFitnessvalue* of this example is set to 25. The *ThreadFitnessvalue* is determined by practical experimentation or trial-and-error simulation tests.

In this example, the mean, best, and worst of the optimal number of rules from 15 runs of the R-ELDMA are 7, 3, and 10 respectively. The learning curve of the proposed R-ELDMA after 15 runs is shown in Fig. 13(a). As shown in this figure, the TFC learns, on average, to balance the pole in the 35th generation. The results show the good control of the trained TFC in the chaotic system.

In this example, as with the first example, R-ELDMA is also compared with other methods, i.e., R-SE [9] and the R-GA [7]. In [7] and [9], the parameter design follows the method described in the first example. The number of rules used in the R-SE and R-GA is four. The results of the parameters used in these methods (R-SE and R-GA) are as follows: 1) population sizes are 170 and 60, respectively; 2) the crossover rates are 0.55 and 0.45, respectively; and 3) the mutation rates are 0.09 and 0.04, respectively. Fig. 13(b) and (c) shows that [7] and [9] learn, on average, to control the chaotic system in the 345th and 263rd generation, respectively. As shown in Fig. 13, the control capabilities of the trained TFC using R-ELDMA are also better than those of [7] and [9] in the chaotic system. Table 12 shows the performance of R-ELDMA (time steps and CPU time) compared with those of various existing models [7,9,6,15,24,36]. The initial parameters for these methods [6,15,24,36] are determined accord-

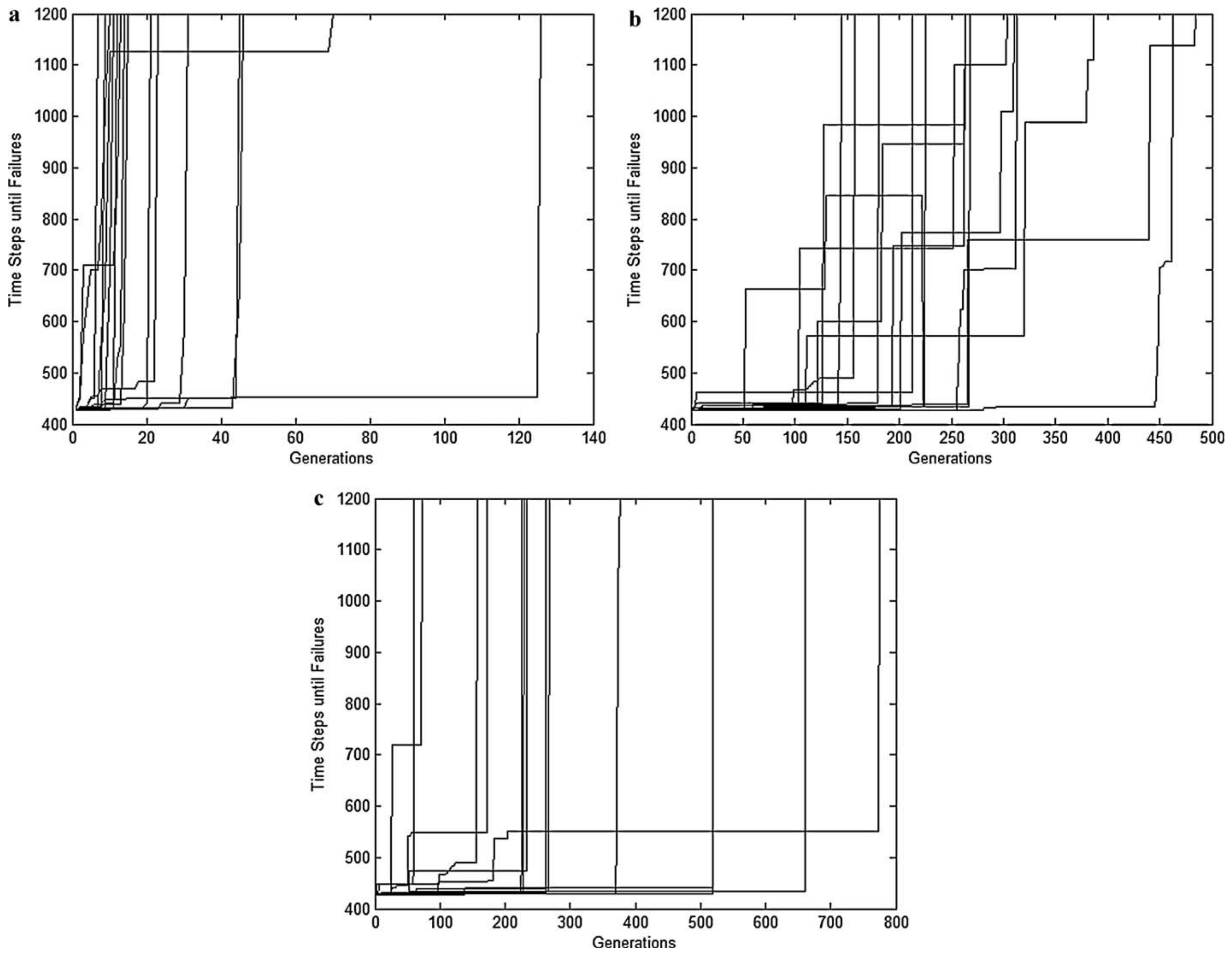


Fig. 13. The performance of (a) the R-ELDMA method, (b) the R-SE method [9], (c) the R-GA [7] method on chaotic system.

Table 12
Comparison of time steps and CPU time for various existing models.

Method	Mean	Mean (s)	Best	Best (s)	Worst	Worst (s)
GENITOR [36]	2701	84.77	343	43.38	5122	176.89
SANE [24]	1198	61.83	79	32.54	3821	123.47
R-GA [7]	345	54.34	64	28.72	775	108.31
R-SE [9]	263	47.63	53	21.31	484	98.39
R-HELA [15]	192	43.57	41	18.93	371	86.23
CQGAF [6]	113	27.36	24	12.22	253	81.59
R-ELDMA	35	13.15	7	3.84	126	40.34

ing to [35]. From Table 12, the proposed R-ELDMA method performs better than other existing models.

6. Conclusion

In this paper, reinforcement evolutionary learning using data mining algorithm (R-ELDMA) with a TSK-type fuzzy controller (TFC) is proposed. R-ELDMA entails both structure and parameter learning. More specifically, it can determine the suitable number of fuzzy rules and tune the parameters of the TFC model efficiently. Moreover, R-ELDMA can also determine the suitable groups for the selection steps. The advantages of the proposed R-ELDMA are summarized as follows: 1) R-ELDMA uses the MGSE so that each group represents only one fuzzy rule; 2) our previous research method (SAM) is applied to determine the suitable number of rules; 3) the proposed DSS is used not only to select the suitable groups but also to identify unsuitable groups for the selection steps; 4) R-ELDMA performs better and converges more quickly than some existing genetic methods. Computer simulations have shown that the R-ELDMA performs better than the other methods.

Although the proposed model demonstrates high performance, it still has some limitations. More specifically, the initial parameters are determined heuristically, which is not a systematic method to determine these parameters. Therefore, future work should identify a well-defined method to determine initial parameters. Moreover, to enhance efficiency, there is a need to reduce the processing time of the DSS.

References

[1] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[2] J.K. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992.

[3] L.J. Fogel, Evolutionary programming in perspective: the top-down view, in: J.M. Zurada, R.J. MarksII II, C. Goldberg (Eds.), Computational Intelligence: Imitating Life, IEEE Press, Piscataway, NJ, 1994.

[4] I. Rechenberg, Evolution strategy, in: J.M. Zurada, R.J. Marks II, C. Goldberg (Eds.), Computational Intelligence: Imitating Life, IEEE Press, Piscataway, NJ, 1994.

[5] W. Pedrycz, M. Reformat, Evolutionary fuzzy modeling, IEEE Trans. Fuzzy Syst. 11 (5) (2003) 652–665.

[6] C.F. Juang, Combination of online clustering and *Q*-value based GA for reinforcement fuzzy system design, IEEE Trans. Fuzzy Syst. 13 (3) (2005) 289–302.

[7] C.L. Karr, Design of an adaptive fuzzy logic controller using a genetic algorithm, in: Proc. the Fourth Int. Conf. Genetic Algorithms, 1991, pp. 450–457.

[8] C.T. Lin, C.P. Jou, GA-based fuzzy reinforcement learning for control of a magnetic bearing system, IEEE Trans. Syst. Man Cybern., Part B 30 (2) (2000) 276–289.

[9] C.F. Juang, J.Y. Lin, C.T. Lin, Genetic reinforcement learning through symbiotic evolution for fuzzy controller design, IEEE Trans. Syst. Man Cybern., Part B 30 (2) (2000) 290–302.

[10] S. Bandyopadhyay, C.A. Murthy, S.K. Pal, VGA-classifier: design and applications, IEEE Trans. Syst. Man Cybern., Part B 30 (6) (2000) 890–895.

[11] B. Carse, T.C. Fogarty, A. Munro, Evolving fuzzy rule based controllers using genetic algorithms, Fuzzy Sets Syst. 80 (3) (1996) 273–293.

[12] K.S. Tang, Genetic algorithms in modeling and optimization, Ph.D. dissertation, Dep. Electron. Eng., City Univ. Hong Kong, Hong Kong, 1996.

[13] F.J. Gomez, Robust non-linear control through neuroevolution, Ph.D. Disseration, The University of Texas at Austin, 2003.

[14] F. Gomez, J. Schmidhuber, Co-evolving recurrent neurons learn deep memory POMDPs, in: Proc. of Conf. on Genetic and Evolutionary Computation, 2005, pp. 491–498.

[15] C.J. Lin, Y.C. Hsu, Reinforcement hybrid evolutionary learning for recurrent wavelet-based neuro-fuzzy systems, IEEE Trans. Fuzzy Syst. 15 (4) (2007) 729–745.

[16] S.F. Lin, Y.C. Cheng, Two-strategy reinforcement evolutionary algorithm using data-mining based crossover strategy with TSK-type fuzzy controllers, Int. J. Innovative Comput. Control 6 (9) (2010).

[17] D.T. Larose, Discovering Knowledge in Data: An Introduction to Data Mining, Wiley Publishers, 2004.

[18] U. Fayyad, Data mining and knowledge discovery in databases: implications for scientific database, in: Proc. Int. Conf. Scientific and Statistical Database Management, 1997, pp. 2–11.

[19] R. Agrawal, R. Srikant, Fast algorithm for mining association rules, Proc. Int. Conf. VLDB (1994) 487–499.

[20] X. Wu, C. Zhang, S. Zhang, Mining both positive and negative association rules, in: Proceedings of the 19th International Conference on Machine Learning, 2002, pp. 658–665.

[21] P.L. Hsu, R. Lai, C.C. Chiu, C.I. Hsu, The hybrid of association rule algorithm and genetic algorithms for tree induction: an example of predicting the student course performance, Expert Syst. Appl. 25 (2003) 51–62.

[22] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man Cybern. 15 (1998) 116–132.

[23] C.F. Juang, C.T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, IEEE Trans. Fuzzy Syst. 6 (1) (1998) 12–32.

[24] D.E. Moriarty, R. Miikkulainen, Efficient reinforcement learning through symbiotic evolution, Mach. Learn. 22 (1996) 11–32.

[25] R.E. Smith, S. Forrest, A.S. Perelson, Searching for diverse, cooperative populations with genetic algorithms, Evol. Comput. 1 (2) (1993) 127–149.

[26] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, in: Proc. ACM-SIGMOD, Dallas, TX, 2000, pp. 1–12.

[27] O. Cordon, F. Herrera, F. Hoffmann, L. Magdalena, Genetic fuzzy systems evolutionary tuning and learning of fuzzy knowledge bases Advances in Fuzzy Systems-Applications and Theory, vol. 19, World Scientific Publishing, NJ, 2001.

[28] E. Cox, Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration, Elsevier Inc., 2005.

[29] G. Lin, X. Yao, Analysing crossover operators by search step size, in: IEEE International Conference on Evolutionary Computation, 1997, pp. 107–110.

[30] I. Dempsey, Constant generation for the financial domain using grammatical evolution, in: Genetic and Evolutionary Computation Conference Workshop Program, 2005, pp. 350–353.

[31] A.G. Barto, R.S. Sutton, C.W. Anderson, Neuron like adaptive elements that can solve difficult learning control problem, IEEE Trans. Syst. Man Cybern. 13 (5) (1983) 834–847.

[32] J.S.R. Jang, C.T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing, Prentice-Hall, 1997.

[33] C.J. Lin, Y.J. Xu, A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications, Fuzzy Sets Syst. 157 (8) (2006) 1036–1056.

[34] K.A. De Jong, Analysis of the behavior of a class of genetic adaptive systems, Ph.D. Disseration, The University of Michigan, Ann Arbor, MI, 1975.

[35] J.J. Grefenstette, Optimization of control parameters for genetic algorithms, IEEE Trans. Syst. Man Cybern. 6 (1) (1986) 122–128.

[36] D. Whitley, S. Dominic, R. Das, C.W. Anderson, Genetic reinforcement learning for neuro control problems, Mach. Learn. 13 (1993) 259–284.