國 立 交 通 大 學
電機與控制工程學系

博士論文

以改良安全性增強式學習為基礎的自我適應進化演算法應用於模糊類神經控制器設計之研究

Improved Safe Reinforcement Learning Based Self Adaptive Evolutionary Algorithms for Neuro-Fuzzy Controller Design

研究生：徐永吉

指導教授：林昇甫 博士

中華民國九十八年八月

以改良安全性增強式學習為基礎的自我適應進化演算法應用於模糊類神經控制器設計之研究

Improved Safe Reinforcement Learning Based Self Adaptive Evolutionary Algorithms for Neuro-Fuzzy Controller Design

研 究 生：徐永吉　　Student：Yung-Chi Hsu

指導教授：林昇甫　博士　　Advisor：Dr. Sheng-Fuu Lin

國 立 交 通 大 學

電機與控制工程學系

博 士 論 文

A Dissertation

Submitted to Department of Electrical and Control Engineering

College of Electrical and Computer Engineering

National Chiao-Tung University

In Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy

in

Electrical and Control Engineering

August 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年八月

以改良安全性增強式學習為基礎的自我適應進化演算法應用於模糊類神經控制器設計之研究

研究生：徐永吉　　指導教授：林昇甫 博士

國立交通大學 電機與控制工程學系

摘　要

在本篇論文中，提出了改良安全性增強式學習為基礎的自我適應進化演算法應用於TSK型式模糊類神經控制器的設計上。本論文所提出的方法可以改善增強式學習的訊號設計以及傳統的進化演算法。本論文的方法可以分為兩部份來探討。首先，在第一部份中本論文提出了自我適應進化演算法來解決傳統進化演算法所遭遇到的問題，如: 1)將所有模糊規則編碼至一條染色體中；2)模糊法則需要在學習前指定；3)無法局部考量單一模糊法則。在第二部份中，本論文提出了改良安全性增強式學習。在改良安全性增強式學習中透過兩個不同的策略-判斷以及衡量策略來決定增強式訊號。此外，Lyapunov穩定性分析也被考量在本論文所提出的改良安全性增強式學習。在本文中提出了單桿以及雙桿倒單擺控制系統來驗證本論文所提出方法的效能，從實驗結果中可以發現，相較於其他進化演算法，本論文所提出的方法有較佳的效能。

關鍵字: TSK 型式模糊類神經控制器，頻率項成長演算法， 進化演算法， 安全性增強式學習， Lyapunov 穩定性。

Improved Safe Reinforcement Learning Based Self Adaptive Evolutionary Algorithms for Neuro-Fuzzy Controller Design

Student：Yung-Chi Hsu　　Advisor：Dr. Sheng-Fuu Lin

Department of Electrical and Control Engineering

National Chiao Tung University

Abstract

In this dissertation, improved safe reinforcement learning based self adaptive evolutionary algorithms (ISRL-SAEAs) are proposed for TSK-type neuro-fuzzy controller design. The ISRL-SAEAs can improve not only the reinforcement signal designed but also traditional evolutionary algorithms. There are two parts in the proposed ISRL-SAEAs. In the first part, the SAEAs are proposed to solve the following problems: 1) all the fuzzy rules are encoded into one chromosome; 2) the number of fuzzy rules has to be assigned in advance; and 3) the population cannot evaluate each fuzzy rule locally. The second part of the ISRL-SAEAs is the ISRL. In the ISRL, two different strategies (judgment and evaluation) are used to design the reinforcement signal. Moreover the Lyapunov stability is considered in ISRL. To demonstrate the performance of the proposed method, the inverted pendulum control system and tandem pendulum control system are presented. As shown in simulation, the ISRL-SAEAs perform better than other reinforcement evolution methods.

Keywords: TSK-type neuro-fuzzy controller, FP-growth algorithm, evolutionary algorithm, safe reinforcement learning, Lyapunov stability.

誌謝

經過四年的奮戰，博士生涯終告一段落，回想起這四年，心中五味雜陳，有痛苦，有喜悅，有沮喪，當然也有歡樂；或許在得到甜美果實前，總是需要努力耕耘吧。

這四年，需要感謝的人太多，因為攻讀博士學位又豈是易事，若不是有這麼多貴人相助與支持，怎有辦法成就今日。

家人，往往是最好的避風港，感謝父親徐茂珍先生、母親陳玉枝女士如此無怨無悔的付出，沒有您們的支持，我無法辦到，身為您們兒子，實在太幸福了；相對於您們的付出，我並不是個稱職的好兒子，總在實驗室忙到沒日沒夜，為了學業的繁忙而忘了關心，照顧您們，卻從未聽您們抱怨，甚至在經濟上如此的資助我，謝謝您們，您們是我永遠的靠山，而將來我也會努力，成為您們的靠山。哥哥徐有順先生以及嫂嫂劉麗芬女士，感謝你們的體諒；如此漫長的求學，父親母親都靠你們照顧，感謝你們的體諒。也要感謝待我如子的小舅陳錦鎮先生以及小舅媽丁淑惠女士，感謝您們一路的照顧與鼓勵，支持我取得博士學位。

指導教授是博士論文產生最重要的一環，我的指導教授-林昇甫博士，感謝您指導學生，在您的教導下，學生學習到很多，您在學術研究上的身教、言教，讓學生的博士生活能無比的充實，也增加了多元的歷練。

博士論文的完善需要口試委員的監督指導，感謝我的口試委員—潘晴財教授、莊仁輝教授、陳錫明教授、林正堅教授以及蔡秀滿教授，感謝您們不辭辛勞不遠千里而來，也感謝您們指導學生口試，有了您們的指導，學生的博士論文才能更臻完備。

研究路上的戰友互相扶持至今，也增添博士生活的多彩多姿，感謝實驗室博士班學長弦澤、同學培家及學弟啟曜及俊偉，碩士班學弟昆義及子航，有你們的並肩作戰，博士生涯才不致如此漫長，我不會忘記深夜與你們在實驗室買雞排當宵夜以及把音樂開很大聲的日子，特別感謝博士班學弟逸章及碩士班學弟長安，感謝你們的協助，此博士論文才能順利完成，你們辛苦了。

感謝工研院機械所 G100、巨匠電腦竹北分校、巨匠電腦新竹認證中心、巨匠電腦中

壢分校以及明新科技大學資管系全體同仁，謝謝你們對我的照顧，讓我在博士班最後階段可以有著多元化的經驗累積。

最後要感謝的，是我未來的家人—秋晨，因為妳在我的身邊，攻讀博士學位才不致如此沉重，感謝妳包容我的脾氣，對妳的依賴，是我攻讀博士學位最大的勇氣，一路走來，因為有妳，我才能更加堅強，未來的日子裡，我會努力成為妳堅強的臂膀。

在此，僅將此論文獻給我最愛的家人、師長、學長、同學、學弟妹、以及我未來的家人秋晨，願與大家分享這難得的榮耀。

本論文完成時適逢八八水災，在此，願 天佑台灣，也希望災區居民能盡快走出傷痛，重建家園，台灣加油!

徐永吉

九十八年　　八月　　十七日

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In real world application, many control problems are so complex that designing controllers by conventional means is either impractical or results in poor performance such as double link control system, inverted pendulum control system, tandem pendulum control system, water temperature control system, and ball and beam balance system, etc. Among them, mathematical models for designing controllers are needed. Inaccurate mathematical modeling of plants usually degrades the performance of the controllers, especially for nonlinear and complex problems. Moreover, even if the neuro-fuzzy controller is adopted for avoiding the complex mathematical models, in real-world applications, precise training data are usually difficult and expensive to obtain. For solving these problems, this dissertation provides a methodology for designing such controllers automatically by evolving improved safe reinforcement learning (ISRL) via neuro-fuzzy controllers using self adaptive evolutionary algorithms (SAEAs).

The introduction of this dissertation is introduced in this chapter. In Section 1.1, a motivation of this dissertation is discussed. The research purpose of this dissertation is introduced in Section 1.2. In Section 1.3, the approach of this dissertation is described. The overview of this dissertation is introduced in the final section.

1.1 Motivation

Neuro-fuzzy controllers ([1]-[14]) 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 neuro-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 a problem in designing a neuro-fuzzy controller. Hence, techniques capable of training the system parameters and finding the global solution while optimizing the overall structure are needed. In this respect, genetic algorithms (GAs) appear to be better candidates. Among GAs, there are two major learning structures using for tuning the parameters of neuro-fuzzy controller: supervised learning ([2], [3], and [6]) and reinforcement learning ([15]-[21]). Among them, for some real-world applications, precise training data are usually difficult and expensive to obtain. For this reason, there has been a growing interest in reinforcement learning algorithms for neural controller ([15]-[18]) or fuzzy design ([19]-[21]). For the reinforcement learning problems, training data are very rough and coarse and there are only "evaluative" when compared with the "instructive" feedback in the supervised learning problem. In reinforcement learning, there is an agent which can choose which action gets the maximum reward in every state. The only feed back is the reward signal of success or failure.

There are several evolutionary algorithms ([22]-[31]) which have been proposed to tune the parameters of the fuzzy controller. These algorithms may require one or more of the following problems: 1) it is difficult to know if it works with different conditions of the system or it can still controlled or nor in the letter time steps; 2) all the fuzzy rules are encoded into one chromosome; 3) the number of fuzzy rules has to be assigned in advance; 4) the population cannot evaluate each fuzzy rule locally.

As mentioned above, improved safe reinforcement learning (ISRL) based self adaptive evolutionary algorithms (SAEAs) for neuro-fuzzy controller are proposed. The proposed

ISRL-SAEAs focus on not only the reinforcement learning but also the structure of the chromosomes in evolutionary algorithms. Therefore, in reinforcement learning, the architecture should consider not only how well and how soon the controller controls the system but also the stability analysis of the reinforcement learning. Moreover, in evolutionary algorithms, the number of fuzzy rules should be decided automatically and the population should evaluate each fuzzy rule locally.

1.2 Review of previous works

In recent years, a fuzzy system used for control problems has become a popular research topic because of classical control theory usually requires a mathematical model for designing controllers ([1]-[10]). Inaccurate mathematical modeling of plants usually degrades the performance of the controllers, especially for nonlinear and complex problems ([11]-[14]). A fuzzy system consists of a set of fuzzy if-then rules. Conventionally, the selection of fuzzy if-then rules often relies on a substantial amount of heuristic observations to express the knowledge of proper strategies. Obviously, it is difficult for human experts to examine all the input-output data from a complex system to find proper rules for a fuzzy system. To cope with this difficulty, several approaches try to generate if-then rules from numerical data have been proposed ([2], [3], and [6]). These methods were developed for supervised learning; that is, the correct “target” output values are given for each input pattern to guide the network's learning. It is a powerful training technique that can be applied to networks. However, if the precise training data can be obtained easily, the supervised learning algorithm may be efficient in many applications. For some real-world applications, precise training data are usually difficult and expensive to obtain. For this reason, there has been a growing interest in reinforcement learning problems ([15]-[21]). For the reinforcement learning problems, training data are very rough and coarse and there are only “evaluative” when compared with

the “instructive” feedback in the supervised learning problem.

In the reinforcement learning, the well known algorithm is Barto and his colleagues’ actor-critic architecture ([17]), which consists of a control network and a critic network. However, the Barto’s architecture is complicated and is not easy to implement. About this, several researches proposed time-step reinforcement architecture to improve the Barto’s architecture ([18]-[20]). In time-step reinforcement architecture, the only available feedback is a reinforcement signal that notifies the model only when a failure occurs. An accumulator accumulates the number of time steps before a failure occurs. Even though time-step reinforcement architecture is easier to implement when compared with Barto’s architecture, it can only measure the number of time steps before a failure occurs; in other words, it only evaluates how long the controller works well instead of how soon the system can enter the desired state, which is also very important. Recently, Perkins and Barto proposed a safe reinforcement learning based on Lyapunov function design ([32]). Once the system’s Lyapunov function is identified, under Lyapunov-based manipulations on control laws, the architecture can drive the plant to reach and remain in a predefined desired set of states with probability 1. Then, the time step for the plant entering the desired set of states can indicate the concept of how soon the system becomes stable. Therefore, one major part of this dissertation is identified.

In learning algorithm, the most well known learning algorithm is back-propagation (BP) ([3], [6]-[8]). Since the steepest descent technique used in BP can minimize the error function, the algorithm may reach the local minima very fast and never find the global solution. In addition, the performance of BP training 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. Recently, several evolutionary algorithms, such as the genetic algorithm (GA) ([22]), genetic programming ([23]), evolutionary programming ([24]), and evolution strategies ([25]), have grown into a popular researching area. 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. In recent years, there are several approaches try to use evolutionary algorithms to converge toward the global solutions. The one important field of these approaches is to use evolutionary algorithms for training fuzzy models ([26]-[28]).

The evolutionary fuzzy model generates a fuzzy system automatically by incorporating evolutionary learning procedures. The well-known evolutionary algorithms are the genetic algorithms (GAs). Several genetic fuzzy models, that is, fuzzy models that are augmented by a learning process based on GAs, have been proposed ([26]-[28]). In [26], Karr applied GAs to design the membership functions of a fuzzy controller, with the fuzzy rule set assigned in advance. Carse *et al.* ([27]) used the genetic algorithm to evolve fuzzy rule-based controllers. Lin and Jou ([28]) proposed GA-based fuzzy reinforcement learning to control magnetic bearing systems.

Recently, several improved evolutionary algorithms have been proposed. One catalogs is focus on modified the structure of the chromosomes ([29]-[44]). In such researches, the chromosomes in population represent partial solution or are with different length. In [29], Juang *et al.* proposed genetic reinforcement learning in the design of fuzzy controllers. The GA adopted in [29] was based upon traditional symbiotic evolution which, when applied to fuzzy controller design, complemented the local mapping property of a fuzzy rule. In [30], Bandyopadhyay *et al.* used the variable-length genetic algorithm (VGA) that allows for different lengths of chromosomes in a population. In [33], Ting *et al.* used multiobjective (MO) variable length genetic algorithm to solve the problem of placing wireless transmitters to meet particular objectives. As shown in [33], the authors used multiobjective (MO) variable length genetic algorithm for searching the optimal number, types, and positions of heterogeneous transmitters by considering coverage, cost, capacity, and overlap simultaneously. In [34], Saeidpour *et al.* used variable length genetic algorithm for fuzzy

controller design and used it for promotion voltage profile. In [35], Lin and Hsu proposed a reinforcement self-adaptive evolutionary algorithm with fuzzy system for solving control problems. As shown in [35], both the number of rules and the adjustment of parameters in the fuzzy system are designed concurrently by the proposed algorithm. The illustrative example was conducted to show the performance and applicability of the proposed algorithm.

In [36], the Saha *et al.* proposed a differential evolution based fuzzy clustering for automatic clustering data set. The proposed algorithm has been used as a stochastic optimization tool. As shown in [36], the proposed algorithm performs better than others. In [37], Tang proposed a hierarchical genetic algorithm. The hierarchical genetic algorithm ([38]-[39]) enables the optimization of the fuzzy system design for a particular application. In [39], authors used a hierarchical genetic algorithm for solving the multilevel redundancy allocation problems. As shown in [39], the authors applied the HGA and a conventional GA separately for solving two multilevel series redundancy allocation optimization problems. The simulation results showed that the performance of the HGA is superior to the conventional GA, because it does not depend on the use of vector coding and preserve the original design space.

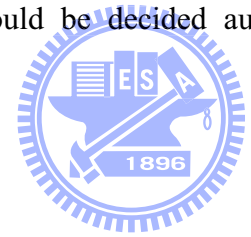
Gomez and Schmidhuber proposed lots of work to evaluate the solution locally ([40] and [41]). The proposed enforced sub-populations (ESP) used sub-populations of neurons for the fitness evaluation and overall control. As shown in [40] and [41], the sub-populations that use to evaluate the solution locally can obtain better performance compared to systems of only one population be used to evaluate the solution. In [42], Li and Miao proposed using ESP backpropagation (BP) neural network to the agent controllers in intelligent virtual environment (IVE). As shown in [42], the ESP was used to solve the task assignment problem of collaboration in an entertainment IVE platform.

Juang [43] proposed the combination of online clustering and Q-value based GA for reinforcement fuzzy system (CQGAF) to simultaneously design the number of fuzzy rules

and free parameters in a fuzzy system. Lin and Xu ([44]) proposed a sequential search-based dynamic evolution (SSDE) to enable better chromosomes to be initially generated while better mutation points are determined for performing dynamic-mutation.

Although the above evolutionary learning algorithms ([29]-[44]) improve the evolutionary learning algorithms through modifying the structure of chromosomes, these algorithms may have one or more of the following problems: 1) all the fuzzy rules are encoded into one chromosome; 2) the number of fuzzy rules has to be assigned in advance; and 3) the population cannot evaluate each fuzzy rule locally.

About above problems, this dissertation focuses on not only the reinforcement learning but also the evolutionary algorithm. Therefore, in reinforcement learning, the architecture should consider how soon the system becomes stable. Moreover, in evolutionary algorithm, the numbers of fuzzy rules should be decided automatically and the population should evaluate each fuzzy rule locally.



1.3 Research Purpose

In this dissertation, improved safe reinforcement learning (ISRL) based self adaptive evolutionary algorithms (SAEAs) for neuro-fuzzy controller is proposed for improving not only the reinforcement signal designed but also evolutionary algorithms mentioned in Section 1.1. There are two parts in the proposed ISRL-SAEAs.

In the first part, self adaptive evolutionary algorithms (SAEAs) are proposed to solve the following problems: 1) all the fuzzy rules are encoded into one chromosome; 2) the number of fuzzy rules has to be assigned in advance; and 3) the population cannot evaluate each fuzzy rule locally. In this dissertation, the proposed self adaptive evolutionary algorithms (SAEAs) consist of three different evolution methods to provide different ways to solve the above problems.

First of all, the hybrid evolutionary algorithm (HEA) with a TSK-type neuro-fuzzy controller is proposed, the proposed HEA determines the number of fuzzy rules automatically and processes the variable-length chromosomes. The length of each individual denotes the total number of genes in that individual. The initial length of each individual may be different from each other, depending on the total number of rules encoded in it. Individuals with an equal number of rules constitute the same group. Thus, initially there are several groups in a population. For keeping the best group in every generation, the elite-based reproduction strategy (ERS) is proposed. In the ERS, the best group can be reproduced many times for each generation. The advantages of the proposed HEA are summarized as follows: 1) it determines the number of fuzzy rules and tunes the free parameters of the neuro-fuzzy controller in a highly autonomous way. Thus, users need not give it any *a priori* knowledge or even any initial information on these. 2) It is applicable to chromosomes of different lengths. 3) It does not require precise training data for setting the parameters of the neuro-fuzzy controller.

Although the proposed HEA can determine the number of fuzzy rules automatically, all the fuzzy rules are encoded into one chromosome. Therefore, partial solution cannot be evaluated independently in the population. The partial solutions can be characterized as *specializations*. The specialization property ensures diversity and prevents a population from converging to suboptimal solutions. A single partial solution cannot "take over" a population since it must correspond with other specializations. For solving this problem, the secondary algorithm of the SAEAs is proposed. In the secondary algorithm of the SAEAs, a self adaptive group cooperation based symbiotic evolution (SAGC-SE) is proposed not only for solving the problem that all the fuzzy rules are encoded into one chromosome but also for letting the population evaluate each fuzzy rule locally. Therefore, in the proposed SAGC-SE, each chromosome represents only one fuzzy rule and an n-rules TSK-type neuro-fuzzy controller is constructed by selecting and combining n chromosomes from several groups. The SAGC-SE, which promotes both cooperation and specialization, ensures diversity and

prevents a population from converging to suboptimal solutions. In SAGC-SE, there are several groups in the population. Each group formed by a set of chromosomes represents a fuzzy rule. The proposed SAGC-SE consists of structure learning and parameter learning. In structure learning, as well as HEA, the SAGC-SE determines the number of fuzzy rules automatically and processes the variable length of a combination of chromosomes. In parameter learning, to let the well-performing groups of individuals for cooperating to generate better generation, an elite-based compensatory of crossover strategy (ECCS) is proposed. In the ECCS, each group will cooperate to perform the crossover steps. Therefore, the better chromosomes of each group will be selected to perform crossover in the next generation.

The advantages of the proposed SAGC-SE are summarized as follows: 1) the proposed SAGC-SE determines the number of fuzzy rules automatically. 2) The SAGC-SE uses group-based population to evaluate the fuzzy rule locally. 3) The SAGC-SE uses the ECCS to allow the better solutions from different groups to cooperate for generating better solutions in the next generation.

The SAGC-SE can solve the problem of the HEA that all the fuzzy rules are encoded into one chromosome. Moreover the SAGC-SE evaluates each fuzzy rule locally for improving the local consideration of the population. However, in the SAGC-SE, how to select groups for constructing the complete solution is a major problem. Therefore, for determining the number of fuzzy rules automatically, the SAGC-SE selects different number of groups to construct complete solution. In this way, the SAGC-SE selects groups randomly. It's obvious that the performance of the SAGC-SE dependents on the method of selecting groups. For solving this problem, in the third algorithm of the SAEAs, a self adaptive groups based symbiotic evolution using FP-growth algorithm (SAG-SEFA) is proposed.

As well as SAGC-SE, the SAG-SEFA consists of structure learning and parameter learning. In structure learning, as well as the SAGC-SE, the proposed SAG-SEFA determines

the number of fuzzy rules automatically and processes the variable combination of chromosomes. In parameter learning, although the proposed SAG-SEFA can determine the suitable number of rules, there still has a problem in which how to select the suitable groups from many groups (named candidate groups in this paper) in SAG-SEFA to construct TSK-type neuro-fuzzy controllers with different numbers of rules. Moreover, in consideration of making the well-performing groups of individuals cooperate for generating better generation, there is also a problem in which how to select suitable groups used to select individuals for cooperating to generate better generation. Regarding this, the goals of parameter learning in SAG-SEFA are used to determine which groups of chromosomes should be selected to construct TSK-type neuro-fuzzy networks with different numbers of rules and which groups should be selected for cooperating to generate better generation.

Recently, data mining has become a popular research topic ([45]-[48]). Data mining is a method of mining information from a database. The database called “transactions”. Data mining can be regarded as a new way of performing data analysis. One goal of data mining is to find association rules among sets of items that occur frequently in transactions. To achieve this goal, several methods have been proposed ([49]-[54]). In [49], the authors proposed a mining method which ascertains large sets of items to find the association rules in transactions. Hang et al. ([50]) proposed frequent pattern growth (FP-growth) to mine frequent patterns without candidate generations. In Hang’s work, items that occur more frequently will have better chances of sharing information than items that occur less frequently. In [51], an algorithm of data mining for transaction data with quantitative values was proposed. In [51], each quantitative item was translated to a fuzzy set and the authors used these fuzzy sets to find fuzzy rules. Wu et al. ([52]) proposed a data mining method based on GA algorithm that efficiently improves the traditional GA by using analysis and confidence parameters. In [53], authors proposed a hybrid model using rough sets and genetic algorithms for fast and efficient improving answering data mining query which involves a random search over large databases.

As shown in [53], authors proposed select, aggregate and classification based data mining queries to implement a hybrid model. The performance of the proposed algorithm is analyzed for both execution time and classification accuracy and the results obtained are good. In [54], Dai and Zhang proposed an association rules mining in novel genetic algorithm. The genetic algorithm in [54] are using for discovering association rules. As shown in [54], the proposed algorithm avoids generating impossible candidates, and it is more efficient than traditional ones.

Since data mining can successfully find information from large sets of items, it is useful to achieve goals of parameter learning in SAG-SEFA. Therefore, the data-mining method called FP-growth algorithm is adopted since the FP-growth algorithm can find items that occur frequently in transactions without candidate generations. After the parameter learning with FP-growth algorithm is performed, the population can search for a better solution from the combination of individuals that perform well and explore other combinations of individuals. Moreover, suitable groups will cooperate to perform the crossover steps. Therefore, the better chromosomes of suitable group will be selected to perform crossover in the next generation.

When compared with SAGC-SE, the proposed SAG-SEFA not only selects the suitable groups form candidate groups to perform selection steps but also allows the better solutions from different groups to cooperate for generating better solutions in the next generation.

The second part of the proposed ISRL-SAEAs is an improved safe reinforcement learning (ISRL). In the ISRL, the feedback takes the form of an accumulator. The accumulator determines by two different strategies (judgment and evaluation). The judgment strategy determines the reinforcement signal when the plant fails entering a predefined goal set and the evaluation strategy applies under the condition that the plant enters the goal set. Moreover the safe reinforcement learning [32] is considered in ISRL. The key of the ISRL is using an accumulator determined by two different strategies as the fitness function of the SAEAs. It

will be observed that the advantage of the proposed ISRL is that it can meet global optimization capability.

1.4 Approach

To demonstrate the performance of the ISRL-SAEAs for temporal problems, this dissertation presents two examples and performance contrasts with some other models.

In the first example, the inverted pendulum control system is adopted to evaluate the performance of the proposed ISRL-SAEAs of this dissertation. This problem is often used as an example of inherently unstable and dynamic systems to demonstrate both modern and classical control techniques ([55]-[57]) or the reinforcement learning schemes ([15]-[21]), and is now used as a control benchmark.

In the second example, the tandem pendulum control system is adopted to evaluate the performance of the proposed method of this dissertation. Since the task of an inverted pendulum control system is too easy to find solutions quickly through random search, in this example, a variety of extensions to an inverted pendulum control system have been suggested. The most challenging extension of an inverted pendulum control system ([58]-[60]) is a tandem pendulum control system, where two pendulums of different length must be balanced synchronously.

1.5 Overview of Dissertation

This dissertation consists of six chapters. In **Chapter 1**, the introduction consists of motivation, review of previous works, research goal, approach, and overview of this dissertation.

In **Chapter 2**, the foundation for the four components of the proposed ISRL-SAEAs by providing background material on neuro-fuzzy controller, reinforcement learning, Lyapunov

stability, and evolutionary algorithm.

In **Chapter 3**, the first part of the proposed ISRL-SAEAs is self adaptive evolution algorithms (SAEAs). The SAEAs consist of a hybrid evolutionary algorithm (HEA), self adaptive groups' cooperation based symbiotic evolution (SAGC-SE), and self adaptive groups based symbiotic evolution using FP-growth algorithm (SAG-SEFA). These three algorithms are introduced in this chapter.

In **Chapter 4**, the second part of the proposed ISRL-SAEAs is the improved safe reinforcement learning (ISRL). The ISRL consists of novel reinforcement signal designed and Lyapunov stability analysis. Both of these two components will be introduced in this chapter.

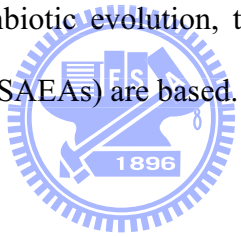
In **Chapter 5,** to demonstrate the performance of the ISRL-SAEAs for temporal problems, two examples and performance contrasts with some other models are presented. The examples of this chapter consist of the inverted pendulum control system and tandem pendulum control system.

In Chapter 6, the contributions and outlines some promising directions for future research are discussed.

Chapter 2

Foundations

The background material and literature review that relates to the major components of the research purpose outlined above (neuro-fuzzy controller, reinforcement learning, Lyapunov stability, and evolutionary algorithm) are introduced in this chapter. The concept of neuro-fuzzy controller is discussed in the first section. The reinforcement learning schema is introduced in Section 2.2. In Section 2.3, the Lyapunov stability that the improved safe reinforcement learning (ISRL) is discussed. The final section focuses on genetic algorithm, cooperative coevolution, and symbiotic evolution, the method on which the proposed self adaptive evolutionary algorithms (SAEAs) are based.



2.1 Neuro-Fuzzy Controller

Neuro-fuzzy modeling has been known as a powerful tool ([1]-[14]) which can facilitate the effective development of models by combining information from different sources, such as empirical models, heuristics and data. Neuro-fuzzy models describe systems by means of fuzzy if–then rules represented in a network structure, to which learning algorithms known from the area of artificial neural networks can be applied.

A neuro-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 output. Two typical types of neuro-fuzzy controllers are Mamdani-type and TSK-type neuro-fuzzy controllers.

For Mamdani-type neuro-fuzzy controllers ([1]), the minimum fuzzy implication is used in fuzzy reasoning. The neuro-fuzzy controllers employ the inference method proposed by Mamdani in which the consequence parts are defined by fuzzy sets. A Mamdani-type fuzzy rule has the form:

$$\textbf{IF } x_1 \textbf{ is } A_{1j}(m_{1j}, \sigma_{1j}) \textbf{ and } x_2 \textbf{ is } A_{2j}(m_{2j}, \sigma_{2j}) \ldots \textbf{and } x_n \textbf{ is } A_{nj}(m_{nj}, \sigma_{nj})$$

$$\textbf{THEN } y' \textbf{ is } B_j(m_j, \sigma_j) \qquad (2.1)$$

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

Besides the more common fuzzy inference method proposed by Mamdani, Takagi, Sugeno and Kang introduced a modified inference scheme ([5]). 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 fuzzy model employs different implication and aggregation methods than the standard Mamdani's type. For TSK-type neuro-fuzzy controllers ([5]), the consequence of each rule is a function input variable. The general adopted function is a linear combination of input variables plus a constant term. A TSK-type fuzzy rule has the form:

$$\textbf{IF } x_1 \textbf{ is } A_{1j}(m_{1j}, \sigma_{1j}) \textbf{ and } x_2 \textbf{ is } A_{2j}(m_{2j}, \sigma_{2j}) \ldots \textbf{and } x_n \textbf{ is } A_{nj}(m_{nj}, \sigma_{nj})$$

$$\textbf{THEN } y' = w_{0j} + w_{1j}x_1 + \ldots + w_{nj}x_n \qquad (2.2)$$

where w_{0j} represents the first parameter of a linear combination of input variables with jth rule node and w_{ij} represents the ith parameter of a linear combination of ith input variable. Since

the consequence of a rule is crisp, the defuzzification step becomes obsolete in the TSK inference scheme. Instead, the model output is computed as the weighted average of the crisp rule outputs, which is computationally less expensive then calculating the center of gravity. Recently, there are many researchers ([5], [35], and [44]) to show that using a TSK-type neuro-fuzzy controller achieves superior performance in network size and learning accuracy than that of Mamdani-type neuro-fuzzy controllers. According to this reason, in this dissertation, a TSK-type neuro-fuzzy controller (TNFC) is adopted to perform various dynamic problems. Therefore, the proposed SAEAs are used to tune free parameters of a TNFC.

The structure of a TNFC is shown in Fig. 2.1, where n and R 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.

$$u_i^{(1)} = x_i, \tag{2.3}$$

where $u_i^{(k)}$ denotes the ith node's input in the kth layer and x_i denotes ith input dimension.

Layer 2 *(Membership Function Node)****:*** Nodes in this layer correspond to one linguistic label of the input variables in layer1; that is, the membership value specifying the degree to which an input value belongs to a fuzzy set ([3]-[4]) is calculated in this layer. In this dissertation, the Gaussian membership function is adopted in this layer. Therefore, for an external input x_i, the following Gaussian membership function is used:

$$u_{ij}^{(2)} = \exp\left(-\frac{\left[u_i^{(1)} - m_{ij}\right]^2}{\sigma_{ij}^2} \right) \tag{2.4}$$

where m_{ij} and σ_{ij} are, respectively, the center and the width of the Gaussian membership

function of the *j*th term of the *i*th 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

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \tag{2.5}$$

Layer 4 *(Consequent Node)**:*** Nodes in this layer are called consequent nodes. The input to a node in layer 4 is the output derived from layer 3, and the other inputs are the input variables from layer 1 as depicted in Fig. 2.1. The function of a node in this layer is

$$u_j^{(4)} = u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i) \tag{2.6}$$

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 single output variable. The node integrates all the actions recommended by layers 3 and 4 and acts as a defuzzifier with

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i)}{\sum_{j=1}^{R} u_j^{(3)}} \tag{2.7}$$

where *R* is the number of fuzzy rule.

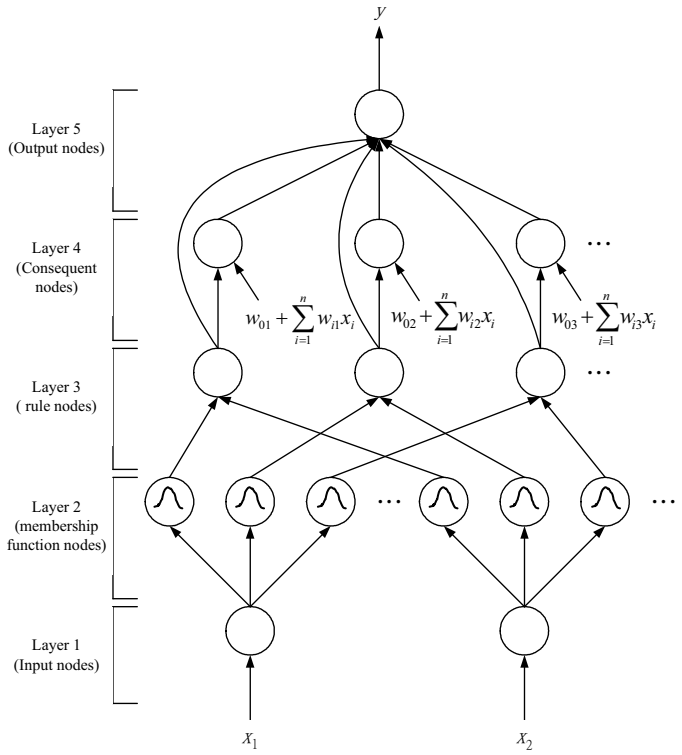


Figure 2. 1: Structure of the TSK-type neuro-fuzzy controller.

2.2 Reinforcement Learning

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. Reinforcement learning algorithm is proposed for determining a sequence of decisions to maximize a reinforcement signal. At each time step, the agent in state $s_t \in S$, chooses an action $a_t \in A$ that transfers the environment to the state s_{t+1} and returns a numerical reward, r_t, to the agent. To lack of knowledge of how to solve the problem, the agent should explore the environment by trial-and-error learning strategy. Unlike supervised learning, the desired output in each state is not known in advance in reinforcement learning. In such trial-and-error learning strategy, an action performs well in the current states may perform badly in the future states, and vice versa.

The well-known learning methods for solving control problems are dynamic

programming ([61]). These methods are similarly to the reinforcement learning ([62]). The necessary component of reinforcement learning methods is shown in Fig. 2.2. The agent consists of a value function and a strategy. The value function represents how much reward can be expected from each state if the best known strategy is performed. The strategy represents how to choose suitable actions from the value function to environment. As shown in Fig. 2.2, at time step t, the agent selects an action a_t. The action is applied to the environment, causing a state transition from s_t to s_{t+1}, and a reward r_t is received. The goal of a reinforcement learning method is to find the optimal value function for a given environment.

There are several reinforcement learning algorithms such as the Q-learning ([63]-[64]) and Sarsa ([65]) algorithms are proposed for computing the value function. These methods are developed based on the temporal difference learning algorithm. In the temporal difference learning method, the value function of each state ($V(s_t)$) is updated using the value function of the next state ($V(s_{t+1})$). The value function of each state is shown as follows.

$$V(s_t) = V(s_t) + \alpha\left[r_t + \lambda V(s_{t+1}) - V(s_t)\right], \tag{2.8}$$

where $V(s_t)$ increases by the reward r_t plus the difference between the next state $\lambda V(s_{t+1})$ and $V(s_t)$; α is the learning rate between 0 to 1; and λ is the discount factor between 0 to 1.

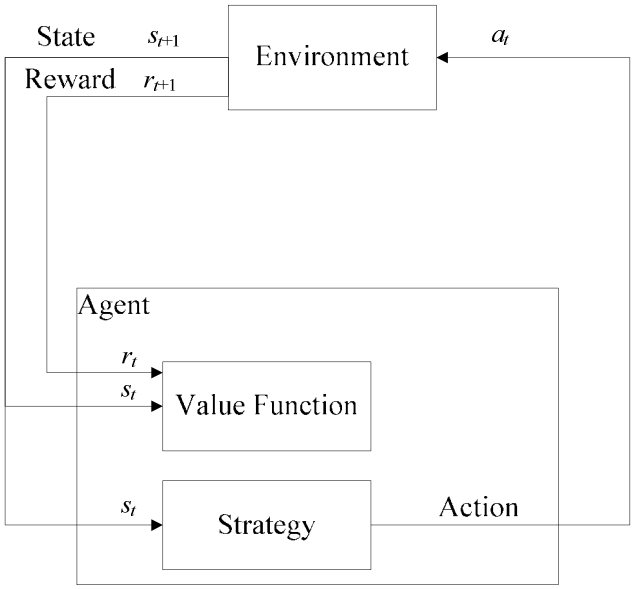


Figure 2. 2: Reinforcement learning method.

In early research, these reinforcement learning algorithms were proposed in simple environments. Recently, the reinforcement learning algorithms focus on larger, high-dimensional environments. These reinforcement learning algorithms are developed base on the neural networks ([66]-[67]), radial basis functions ([68]), and neuro-fuzzy network ([18]-[20]).

More recently, there are several researches proposed time-step reinforcement architectures to provide an easier way to implement the reinforcement learning architecture when compared with temporal difference learning architectures ([18]-[20]). In time-step reinforcement architecture, the only available feedback is a reinforcement signal that notifies the model only when a failure occurs. An accumulator accumulates the number of time steps before a failure occurs. The goal of the time-step reinforcement method is to maximize the value function V. The fitness function is defined by:

$$V = TIME\text{-}STEP \tag{2.9}$$

where *TIME-STEP* represents how long the experiment is still a "success". Equation 2.9 reflects the fact that long-time steps before a failure occurs means the controller can control

the plat well. For example, in evolutionary algorithm, Eq. 2.9 reflects the fact that long-time steps before a failure occurs means higher fitness of the evolutionary algorithm.

2.3 Lyapunov Stability

Consider a following system:

$$\dot{x} = f(x) \tag{2.10}$$

where $f : D \to R^n$ represents a locally Lipschitz that maps from an open set $D \subset R^n$ into R^n. Suppose that $x = 0$ is an equilibrium point for Eq. 2.10; that is, $f(0) = 0$.

According to [69], we have the following definition of stability

Definition 2.3.1:

1. Stable, if $\exists \delta(\varepsilon) > 0$, for $\forall \varepsilon > 0$, such that

$$\|x(0)\| \le \delta \Rightarrow \|x(t)\| < \varepsilon, \ \forall t \ge 0 \tag{2.11}$$

2. Asymptotically stable, if it is stable and there exists some $\gamma > 0$ such that

$$\|x(0)\| < \gamma \Rightarrow \lim_{t \to \infty} x(t) = 0 \tag{2.12}$$

3. Globally asymptotically stable, if it is asymptotically stable and there exists $\lim_{t \to \infty} x(t) = 0$ holds for all $x(0)$.

4. Unstable, if not stable.

The Lyapunov stability theorems ([69]) are introduced as follows.

Theorem 2.3.1 **(**[69]):

Suppose $x = 0$ is an equilibrium point of Eq. 2.10 and $D \subset R^n$ is an open set containing $x = 0$. Let $V : D \to R$ to be a continuously differentiable function as following

$$V(0) = 0 \ \text{ and } \ V(x) > 0 \ \text{ in } \ D \setminus \{0\}. \tag{2.13}$$

According to Eq. 2.13, the following equations hold:

1. If $\dot{V}(x) \le 0$ in D,then $x = 0$ is stable, where $\dot{V}(x)$ is defined by

$$\dot{V}(x) = \frac{\partial V}{\partial x}(f(x)). \tag{2.14}$$

2. If $D = R^n$, $\dot{V}(x) < 0$ in $D \setminus \{0\}$, and $V(x)$ is radially unbounded. An equilibrium point $x = 0$ is globally asymptotically stable, when $\|x\| \to \infty$ and $V(x) \to \infty$.

3. If $\dot{V}(x) < 0$ in $D \setminus \{0\}$, then $x = 0$ is asymptotically stable.

In reinforcement learning, the most well-known algorithm is Barto and his colleagues' actor-critic architecture ([17]), which consists of a control network and a critic network. However, Barto's architecture is complicated and is not easy to implement. Therefore, there are several researches proposed time-step reinforcement architectures to improve Barto's architecture ([18]-[20]). In time-step reinforcement architecture, the only available feedback is a reinforcement signal that notifies the model only when a failure occurs. An accumulator accumulates the number of time steps before a failure occurs. Even though the time-step reinforcement architecture is easier to implement when compared with Barto's architecture, it only measures the number of time steps before a failure occurs; in other words, it only evaluates how long the controller works well instead of how soon the system can enter the desired state, which is also very important. In [32], Perkins and Barto proposed safe reinforcement learning based on Lyapunov function design. Once the system's Lyapunov function is identified, under Lyapunov-based manipulations on control laws, the architecture can drive the system to reach and remain in a predefined desired state with probability 1. Then, the time step for the system entering the desired state can indicate the concept of how soon the system becomes stable. In this dissertation, the Lyapunov stability theorem is used to design the reinforcement signal; therefore, the improved safe reinforcement learning (ISRL) is based on the Lyapunov stability theorem. The details of the ISRL can be found in Chapter 4.

2.4 Evolution Learning

In this section, the foundations of evolutionary algorithm are introduced. This section focuses on genetic algorithm, cooperative coevolution and symbiotic evolution, the methods on which the proposed self adaptive evolutionary algorithms (SAEAs) are based.

2.4.1 Genetic algorithm

Genetic algorithms (GAs) ([22]) are search algorithms inspired by the mechanics of natural selection, genetics, and evolution. It is widely accepted that the evolution of living beings is a process that operates on chromosome-organic devices for encoding the structure of living beings.

The flowchart of the learning process is shown in Fig. 2.3, where *Nc* is the size of population, *G* denote *G*th generation. The learning process of the GAs involves three major steps: reproduction, crossover, and mutation. Reproduction ([70]-[72]) is a process in which individual strings are copied according to their fitness value. This operator is an artificial version of neural selection. In GAs, a high fitness value denotes a good fit. In the reproduction step, the well-known method is the roulette-wheel selection method ([72]) (see Fig.2.4). In Fig.2.4, the intermediate population is *P'*, which is generated from identical copies of a chromosome sampled by spinning the roulette wheel a sufficient number of times.

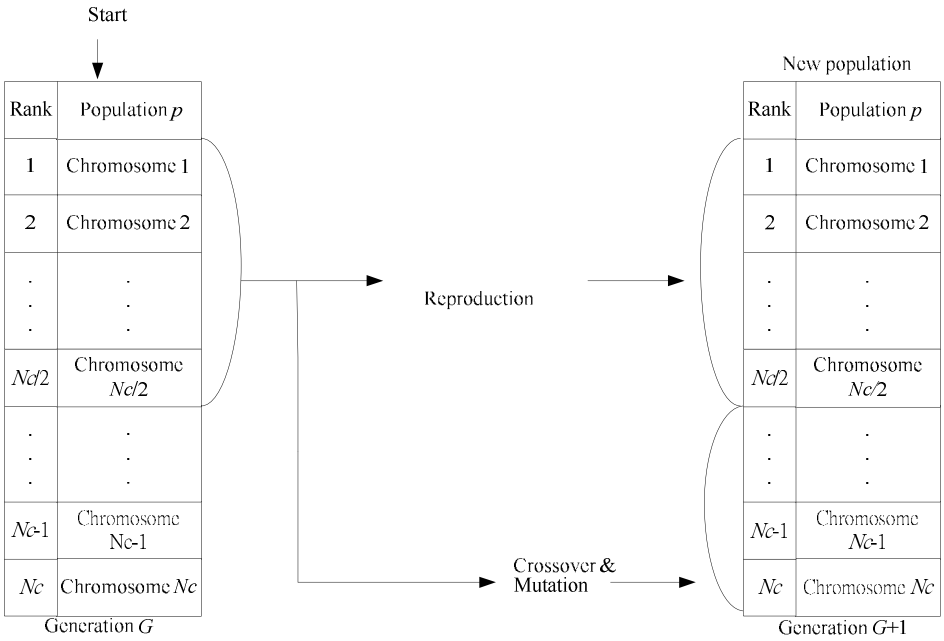


Figure 2. 3: Flowchart of the genetic algorithm.

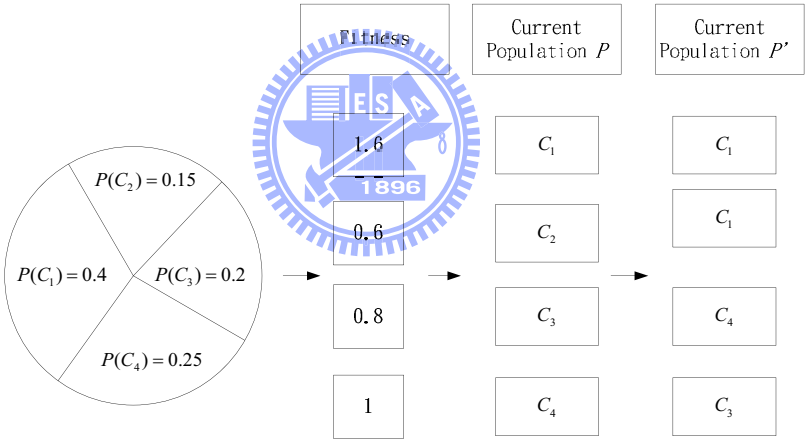


Figure 2. 4: The roulette wheel selection.

In crossover step ([73]-[77]), although reproduction step directs the search toward the best existing individuals, it cannot 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 occurred for a selected pair with a crossover rate. Figure 2.5 illustrates how the crossover works. Crossover produces two offspring from their parents by exchanging chromosomal genes on either side of a crossover point generated randomly.

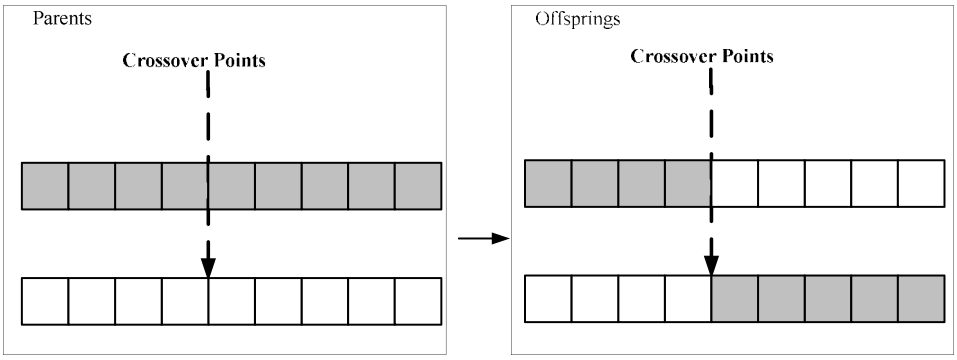


Figure 2.5: Crossover operator.

In mutation step ([78]-[84]), although the reproduction and crossover would produce many new strings, they do not introduce any new information to the population at the site of an individual. Mutation can randomly alter the allele of a gene. The operation is occurred with a mutation rate. Figure 2.6 illustrates how the mutation works. When an offspring is mutated, one of its genes selected randomly is changed to a new value.

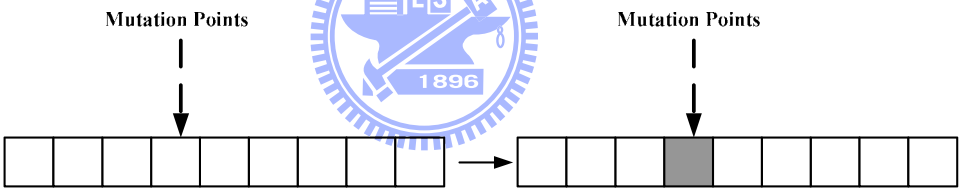


Figure 2.6: Mutation operator.

Since GAs search many points in the space simultaneously, they have less chance to reach the local minima than single solution methods. The advantages of GAs are (1) some individuals have a better chance to come close to the global optima solution, and (2) the genetic operators allow the GA to search optima solution. According to above reasons, GAs are suitable for searching the parameters space of neuro-fuzzy controller. For solving the problem that a neuro-fuzzy controller which performs gradient-descent based learning algorithms may reach the local minima very fast but never find the global solution, the GAs sample the parameters space of neuro-fuzzy controllers and recombine those that perform best on the control problem.

2.4.2 Cooperative Coevolution

In natural evolution, individuals may compete and/or cooperate with each other for resources and survival. The fitness of each individual may changes each generation. The reason is that individuals compete and/or cooperate with other individuals in the environment. About this phenomenon, recently, many researches try to propose coevolutionary algorithms to improve the traditional evolutionary algorithm. Most coevolutionary algorithms focus on competition between individuals in the population ([85]-[87]). Therefore, individuals generate stronger and stronger strategies to defeat others in every generation.

Another kind of coevolutionary algorithms is proposed to improve cooperation of GAs. Cooperative coevolution is proposed for reducing the difficult problems through modularization ([88]). Therefore, in cooperative coevolutionary algorithms, the individuals represent only partial solutions. The partial solutions are evolved by evaluating their performance to complete solutions and recombining the partial solutions with well performance to solve the problem. Cooperative coevolution algorithms can improve the performance of traditional evolution by dividing the problem into several small problems.

In [89], Holland and Reitman proposed cooperative coevolution algorithms to apply in classifier systems. The fitness value is assigned to each individual on how well it cooperates with others. This approach is implemented by a neural network. Recently, there are several researches try to use coevolution algorithms to radial basis functions ([90]-[93]). In [94], the authors proposed a cooperative coevolutionary GA that each individual is evaluated independently on its own population.

More recently, several researches try to propose algorithms to combine cooperative coevolution with neural networks ([95]-[96]) and neuro-fuzzy controller ([29], [31], and [44]) to improve the performance. The approach called symbiotic evolution will be introduced in next section.

2.4.3 Symbiotic Evolution

In this section, an approach of cooperative coevolution is introduced. Therefore, the symbiotic evolution is discussed. The idea of symbiotic evolution was first proposed in an implicit fitness sharing algorithm that was used in an immune system model ([97]). The authors developed artificial antibodies to identify artificial antigens. Because each antibody can match only one antigen, a different population of antibodies was required to effectively defend against a variety of antigens.

Unlike traditional GAs that use each individual in a population as a full solution to a problem, symbiotic evolution assumes that each individual in a population represents only a partial solution to a problem; complete solutions combine several individuals in the population. In a normal evolution algorithm, a single individual is responsible for the overall performance, with a fitness value assigned to that individual according to its performance. In symbiotic evolution, the fitness of an individual (a partial solution) is calculated by summing up the fitness values of all possible combinations of that individual with other current individuals (partial solutions) and dividing the sum by the total number of combinations.

As shown in [29], [31], [44], and [95]-[96], partial solutions can be characterized as *specializations*. The specialization property ensures diversity, which prevents a population from converging to suboptimal solutions. A single partial solution cannot "take over" a population since there must be other specializations 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 ([29], [31], [44], and [95]-[96]).

The basic idea of symbiotic evolution is that an individual (i.e., a chromosome) is used to represent a partial solution. A complete solution is formed when several individuals,

which are randomly selected from a population, are combined. With the fitness assignment performed by symbiotic evolution, and with the local property of a fuzzy rule, symbiotic evolution and the fuzzy system design can complement each other. If a normal GA evolution scheme is adopted, only the overall performance of the complete solution is known, not the performance of each partial solution. The best method to replace the unsuitable partial solutions that degrade the overall performance of a fuzzy system is to use crossover operations, followed by observing the performance of the offspring.

The structure of the symbiotic evolution is shown in Fig. 2.7, where *N* is the number of complete solutions the symbiotic evolution will select individuals to form. The complete solution is constructed by selecting the individuals from a population. The learning flowchart is shown in Fig. 2.8, where *Nc* is the size of population, and *G* denotes *G*th generation. Compare with genetic algorithm, the difference of symbiotic evolution is selecting individuals to form a complete solution (selection step) and to evaluate the performance of each individual (fitness assignments step).

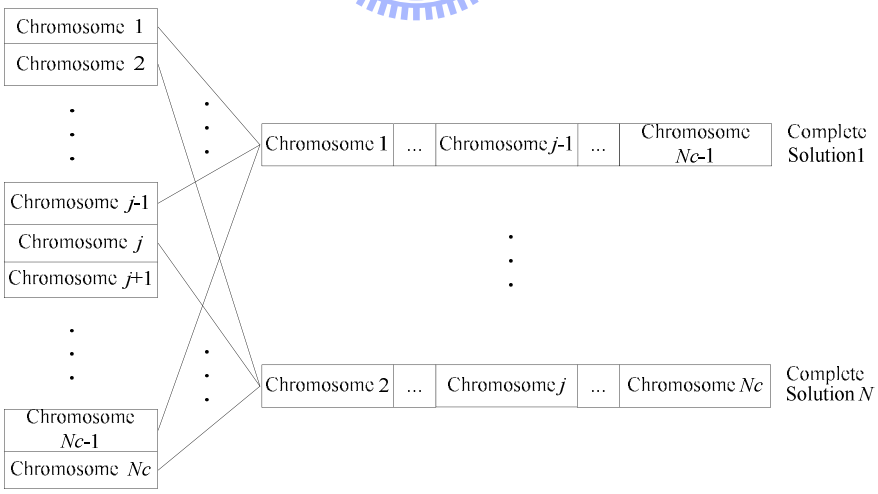


Figure 2.7: Structure of a chromosome in a symbiotic evolution.

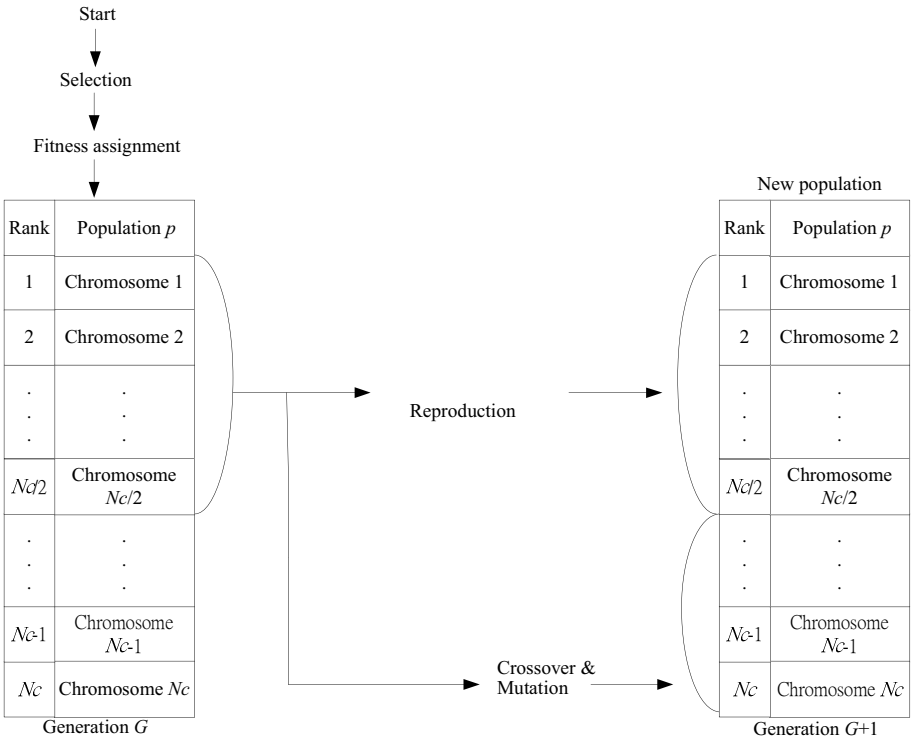


Figure 2.8: Flowchart of the symbiotic evolution.

In the selection step, the sufficient times are needed to select individuals. Therefore, the selection times must be large enough to let every individual in population to be selected sufficiently.

In the fitness assignments step, the fitness value is defined as follows:

Step 1. Randomly choose *R* individuals in population to form complete solution. The *R* represents the number of individuals that the complete solution needs (in this dissertation, *R* represents the number of fuzzy rules).

Step 2. Evaluate every complete solution with *R* individuals, which are selected from step1, to obtain a fitness value.

Step 3. Divide the fitness value by *R* and accumulate the divided fitness value to the selected individuals with their fitness value records are set for zero initially.

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

single individual.

Although the symbiotic evolution uses the specialization property to ensure diversity, which prevents a population from converging to suboptimal solutions, there still has a problem that the population cannot evaluate each partial solution locally. For example, the partial solutions are all encode in a population. In the evolutionary operator, the chromosomes with better fitness have most chance to reproduce in the offspring. Therefore, the chromosomes of population may become similarly. This will cause the complete solution hard to find by combining of the chromosomes in the population. For solving this problem, the SAEAs are proposed in this dissertation to let the population can evaluate each partial solution locally. The details of the SAEAs can be found in Chapter 3.

Chapter 3
Self Adaptive Evolutionary Algorithms

In this chapter, the first part of the proposed ISRL-SAEAs, that is, the self adaptive evolution algorithms (SAEAs), is introduced. The three methods contained in the SAEAs will be introduced in the following sections. In the first section, a hybrid evolutionary algorithm (HEA) is introduced for improving the traditional evolution that the length of individuals in the population must be predefined. In the second section, a self adaptive group cooperation based symbiotic evolution (SAGC-SE) is proposed for providing the local consideration of the population in the HEA. In the final section, a self adaptive group based symbiotic evolution using FP-growth algorithm (SAG-SEFA) is discussed to provide the methodology of selecting groups for performing selection and crossover steps in SAGC-SE.

3.1 Self Adaptive Hybrid Evolutionary Algorithm

In this section, the proposed hybrid evolutionary algorithm (HEA) is introduced. Recently, many efforts to enhance the traditional GAs have been made ([98]). Among them, one category focuses on modifying the structure of a population or the role an individual plays in it ([85]-[87] and [95]-[100]), such as the coevolutionary algorithms ([85]-[87]), distributed GA ([99]), cellular GA ([100]), and symbiotic evolution ([95]-[97]).

In a traditional evolution algorithm, the number of rules in a model must be predefined. The length of individuals (chromosomes) in a population is the same and is defined by

trail-and-error testing. Therefore, for different control problem, the length of an individual (chromosome) must be redefined. For solving this problem, the hybrid evolutionary algorithm (HEA) is proposed to decide the number of fuzzy rules automatically.

The proposed HEA combines the modified compact genetic algorithm (MCGA) and the modified variable-length genetic algorithm (MVGA). In the MVGA, the initial length of each individual may be different from each other, depending on the total number of rules encoded in it. Thus, the number of rules does not need to be predefined. In HEA, individuals with an equal number of rules constitute the same group. Initially, there are several groups in a population. Unlike the traditional variable-length genetic algorithm (VGA), Bandyopadhyay *et. al.* ([30]) used "#" to mean, "does not care". In the HEA, the variable two-part crossover (VTC) and variable two-part mutation (VTM) are adopted to make the traditional crossover and mutation operators applicable to different lengths of chromosomes. Therefore, using "#" means that "does not care" is not needed in the VTC and VTM.

In the HEA, a chromosome is divided into two parts. The first part of the chromosome gives the antecedent parameters of a TSK-type neuro-fuzzy controller while the second part of the chromosome gives the consequent parameters of a TSK-type neuro-fuzzy controller. Each part of the chromosome can be performed using the VTC on the overlapping genes of two chromosomes. In the traditional VGA, Bandyopadhyay *et. al.* ([30]) only evaluated the performance of each chromosome in a population. The performance of the number of rules was not evaluated ([30]). In HEA, the elite-based reproduction strategy is proposed to keep the best group with the same length chromosomes. Therefore, the best group can be reproduced many times for each generation. The elite-based reproduction strategy is similar to the maturing phenomenon in society, where individuals become more suitable to the environment as they acquire knowledge from society.

In the proposed HEA, the modified compact genetic algorithm (MCGA) is proposed to carry out the elite-based reproduction strategy. The concept of compact genetic algorithm

(CGA) ([101]) represents a population as a probability distribution over the set of solutions and is operationally equivalent to the order-one behavior of the simple GA ([102]). The advantage of the CGA is that it processes each gene independently and requires less memory than the normal GA. In the proposed MCGA, the building blocks (BBs) in the MCGA represent the suitable lengths of the chromosomes and reproduce the individuals according to the BBs. The coding scheme consists of the coding done by the MVGA and the MCGA. The MVGA codes the adjustable parameters of a TSK-type neuro-fuzzy controller (TNFC) into an individual (chromosome), as shown in Fig. 3.1; where MS_j represents the parameters of the antecedent of the jth rule in the TNFC, W_j represents the parameters of the consequent of the jth rule, w_{ij} is the corresponding parameter of the consequent part with the jth rule and ith input variable, and R_k represents that there are R_k fuzzy rules in a TNFC. In Fig. 3.2, the MCGA codes the probability vector into the building blocks (BBs), where each probability vector represents the suitability of the number of fuzzy rules in a TNFC. In MCGA, the maximum number of rules ($R_{\max}$) and minimum number of rules ($R_{\min}$) must be predefined to prevent the number of fuzzy rules to generate beyond a certain bound (i.e., [$R_{\min}$, $R_{\max}$]).

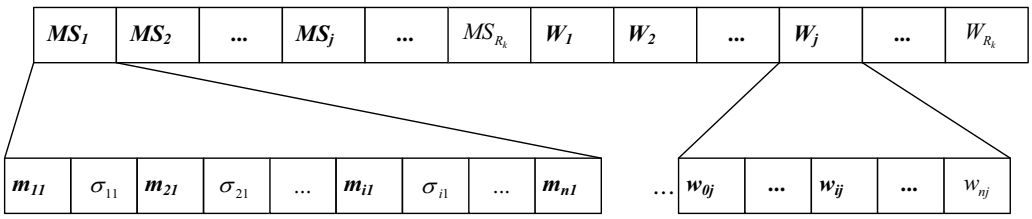


Figure 3.1: Coding the adjustable parameters of a TNFC into a chromosome in the MVGA.

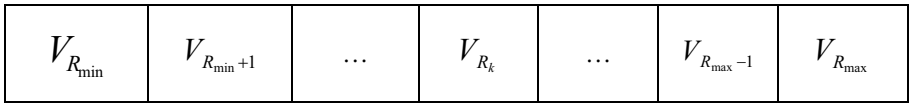


Figure 3.2: Coding the probability vector into the building blocks (BBs) in the MCGA.

The learning process of the HEA involves six major operators: initializing, evaluating, sorting, elite-based reproduction strategy, variable two-part crossover, and variable two-part

mutation. Figure 3.3 shows the flowchart of the learning process. The whole learning process is described step-by-step as follows:

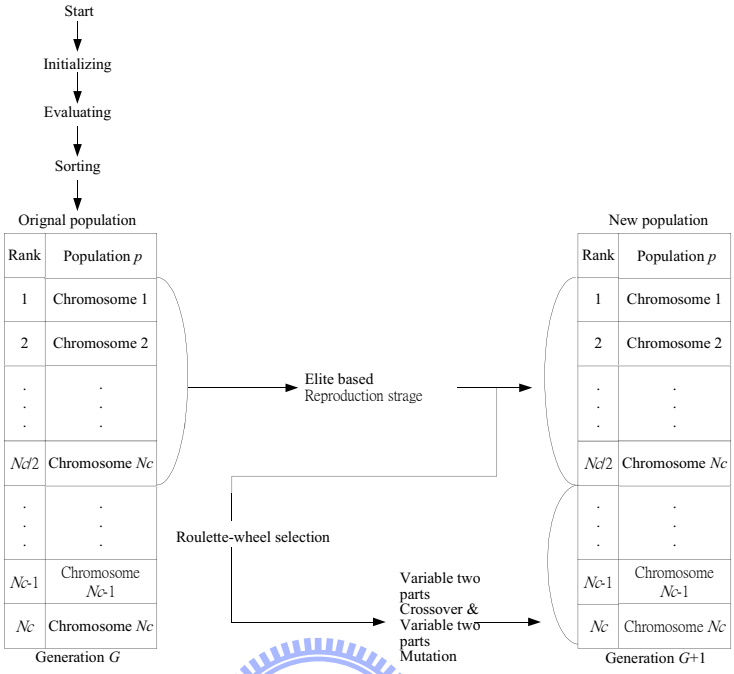


Figure 3.3: Flowchart of the parameter learning in the HEA.

1. Initializing:

The initializing step sets initial values of the MVGA and MCGA. In the MVGA, individuals should be generated randomly to construct an initial population. The initial values of MVGA are generated randomly within a fixed range. The following formulations show how to generate the initial chromosomes:

$$\begin{aligned} &\text{Mean: } Chr_c[p] = random[m_{\min},\ m_{\max}] \\ &\quad \text{where } c = 1,\ 2,\ \cdots,\ Nc; \\ &\qquad p = 1,\ 3,\ \cdots,\ 2(n \times R_k) - 1; \\ &\qquad R_k = R_{\min},\ R_{\min}+1,\ \cdots,\ R_{\max}, \end{aligned} \tag{3.1}$$

$$\begin{aligned} &\text{Deviation: } Chr_c[p] = random[\sigma_{\min},\ \sigma_{\max}] \\ &\quad \text{where } p = 2,\ 4,\ \cdots,\ 2(n \times R_k), \end{aligned} \tag{3.2}$$

$$\begin{aligned} &\text{Weight: } Chr_c[p] = random[w_{\min},\ w_{\max}] \\ &\quad \text{where } p = 2(n \times R_k)+1,\ 2(n \times R_k)+2,\ \cdots,\ 2(n \times R_k)+3n+1, \end{aligned} \tag{3.3}$$

where Chr_c represents cth chromosome in a population; R_k represents they are R_k rules in a TNFC and N_C is the total number of chromosomes in each group; p

represents the pth gene in a Chr_c; and [$\sigma_{\min}$, $\sigma_{\max}$], [$m_{\min}$, $m_{\max}$], and [$w_{\min}$, $w_{\max}$] represent the predefined range.

In order to keep the same number of rules in a TNFC, the number of the rules of each chromosome needs to be generated η chromosomes. That is, the number of chromosomes generated of each group (η) must be predefined. Therefore, the population size Nc is set to η*($R_{\max}$-$R_{\min}$ +1). In the MCGA, the probability vectors of the BBs are set to 0.5 initially.

2. Evaluating:

The evaluating step is to evaluate each chromosome in a population. The goal of the HEA is to maximize the fitness value. The higher a fitness value, the better the fitness. The fitness function is used by a reinforcement signal defined in Chapter 4 will be introduced later.

3. Sorting:

After the evaluating step, the chromosomes in the population are sorted. After sorting the whole population, chromosomes in the top half of population are also sorted in each group. The sorting step can help performing the reproduction step because of the best chromosome in each group can be stayed. After sorting the chromosomes in the population, the algorithm goes to next step.

4. Elite-Based Reproduction Strategy (ERS):

Reproduction is a process in which individual strings are copied according to their fitness value. In the HEA, an elite-based reproduction strategy (ERS) is proposed to mimic the maturing phenomenon in society, where individuals become more suitable to the environment as they acquire more knowledge from society. The MCGA uses the BBs to represent the suitable length of the chromosomes and reproduces the chromosomes according to the probability vector in the BBs. The best performing individuals where in the top half of each population are using to perform

the ERS. According to the results of the ERS, using the crossover and the mutation generate the other half individuals. After the ERS, the suitable length of chromosomes will be preserved and the unsuitable length of chromosomes will be removed. The detailed of the ERS is shown as follows:

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

$$\begin{cases} V_{R_k} = V_{R_k} + (Upt\_value_{R_k} \times \lambda), & if \quad Avg \le Max\_fit_{R_k} \\ V_{R_k} = V_{R_k} - (Upt\_value_{R_k} \times \lambda), & otherwise \end{cases} \tag{3.4}$$

where $R_k = R_{\min}, R_{\min}+1, \cdots, R_{\max}$,

$$Avg = \sum_{p=1}^{Nc} fit_p / Nc, \tag{3.5}$$

$$Upt\_value_{R_k} = Total\_fit_{R_k} \Big/ \sum_{p=1}^{Nc} fit_p, \tag{3.6}$$

$$Total\_fit_{R_k} = \sum_{q=1}^{N_{R_k}} fit_{R_k q}, \tag{3.7}$$

where V_{R_k} is the probability vector in the BBs and represents the suitable chromosomes in the group with R_k rules in a population; λ is a predefined threshold value; *Avg* represents the average fitness value in the whole population; *Nc* is the population size; N_{R_k} represents that there are N_{R_k} chromosomes in the group that individuals with R_k rules; fit_p is the fitness value of the *p*th chromosome in all *Nc* populations; $fit_{R_k q}$ is the fitness value of the *q*th chromosome in the group with R_k rules; and $Max\_fit_{R_k}$ is the best fitness value in the group with R_k rules. As shown in Eq. 3.4, if $Max\_fit_{R_k} \ge Avg$, then the suitable chromosomes in the group with R_k rules should be increased. On the other hand, if $Max\_fit_{R_k} < Avg$, then the suitable chromosomes in the group with R_k rules should be decreased. Eq. 3.7 represents the

sum of the fitness values of the chromosomes in the group with R_k rules.

Step 2. Determine the reproduction number according to the probability vectors of the BBs as follows:

$$Rp_{R_k} = (Nc/2) * (V_{R_k} / Total\_Velocy)$$

$$\text{where } R_k = R_{\min}, R_{\min}+1, \cdots, R_{\max}, \tag{3.8}$$

$$Total\_Velocy = \sum_{R_k=R_{\min}}^{R_{\max}} V_{R_k} \tag{3.9}$$

where Nc represents the population size; Rp_{R_k} is the recorder, and a chromosome has R_k rules for constructing a TNFC.

Step 3. After step 2, the reproduction number of each group in the top half of a population is obtained. Then Rp_{R_k} chromosomes in each group are generated using the roulette-wheel selection method ([72]).

Step 4. In the proposed ERS, for avoiding suitable number of fuzzy rules may fall in the local optima solution, the two different actions to update the V_{R_k} are adopted. The two deferent actions are defined according to the following equations:

$$\begin{aligned} &\text{if} \quad Accumulator \le ERSTimes \\ &\text{then} \quad \text{do} \quad \text{Steps} \quad 1 \quad \text{to} \quad 3 \end{aligned} \tag{3.10}$$

$$\begin{aligned} &\text{if} \quad Best\_Fitness_g = Best\_Fitness \\ &\text{then} \quad Accumulator = Accumulator + 1; \end{aligned} \tag{3.11}$$

$$\begin{aligned} &\text{if} \quad Accumulator > ERSTimes \\ &\text{then} \quad \text{do} \quad \text{Step} \quad 0 \quad \text{and} \quad Accumulator = 0, \end{aligned} \tag{3.12}$$

where $ERSTimes$ is a predefined value; $Best\_Fitness_g$ represents the best fitness value of the best combination of chromosomes in the gth generation; $Best\_Fitness$

represents the best fitness value of the best combination of chromosomes in current generations. Eqs. 3.10-3.12 represent that if the best fitness is not changed for a sufficient generations, the suitable number of fuzzy rules may fall in the local optima solution.

5. Variable two-part crossover:

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 the HEA, the variable two-part crossover (VTC) is proposed to perform this step. In the VTC, the parents are selected from the enhanced elites. In the VTC, two parents are selected by using the roulette-wheel selection method ([72]). The two parents may be selected from the same or different groups. Performing crossover on the selected parents creates the offspring. Since the parents may be of different lengths, the misalignment of individuals must be avoided in the crossover operation. Therefore, a variable two-part crossover is proposed to solve this problem. The first part of the chromosome gives the antecedent parameters of a TNFC while the second part of the chromosome gives the consequent parameters of a TNFC. The two-point crossover ([76]) is adopted in each part of the chromosome. Thus, new individuals are created by exchanging the site's values between the selected sites of the parents' individuals. To avoid the misalignment of individuals in the crossover, in the VTC, the selection of the crossover points in each part will not exceed the shortest length chromosome of two parents. Two individuals of different lengths using the variable two-part crossover operation are shown in Fig. 3.4. MS_j represents the parameters of the antecedent part of the jth rule in the TNFC, W_j represents the parameters of the consequent of the jth rule in the TNFC, and R_k represents that there are k fuzzy rules

in a TNFC. After the VTC, the individuals with poor performance are replaced by the new offspring.

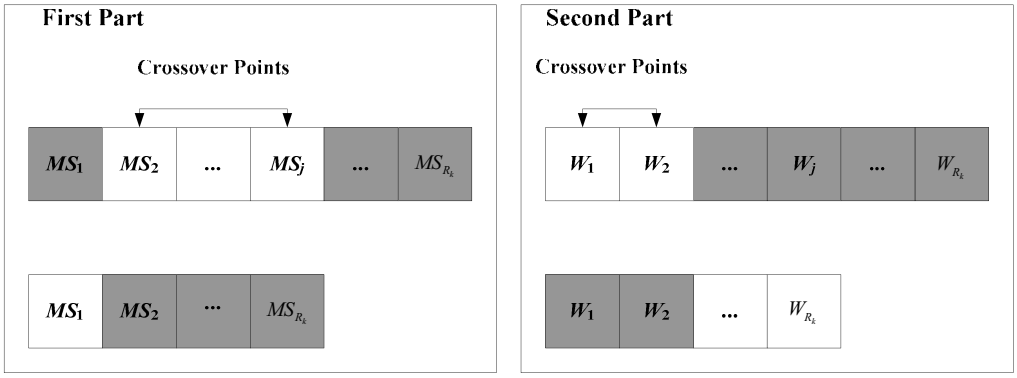


Figure 3.4: The variable two-part crossover operation in the HEA.

6. Variable two-part mutation:

Although the ERS and the VTC produce many new strings; these strings do not provide any new information to every population at the site of an individual. Mutation can randomly alter the allele of a gene. In the HEA, the variable two-part mutation (VTM) is proposed to perform the mutation operation. The proposed VTM is different from the traditional mutation and is applicable to chromosomes of different length. The first and second parts of the chromosome are the same as the VTC. In each part of a chromosome, uniform mutation ([84]) is adopted, and the mutated gene is drawn randomly from the domain of the corresponding variable. The VTM operation of each individual is shown in Fig. 3.5.

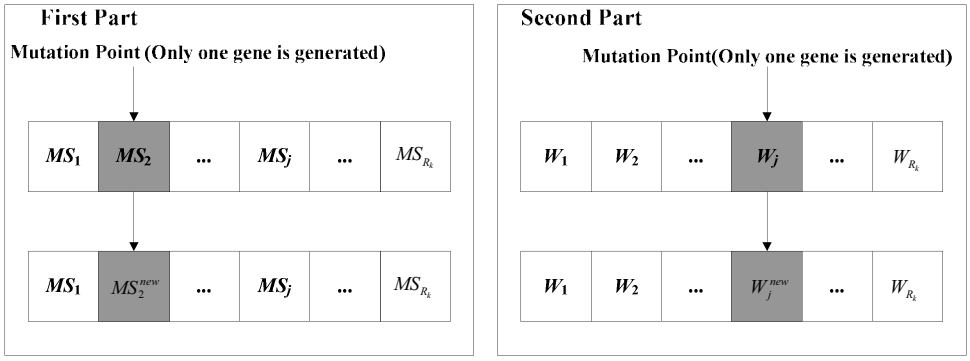


Figure 3.5: The variable two-part mutation operation in the HEA.

After the above-mentioned operations, the problem of groups constituted by the most suitable number of rules will be solved. The number of elites in other groups will decrease and most of them will become zero (in most cases, there will be no elites). That is, the proposed HEA indeed can eliminate unsuitable groups and rules.

As mention above, the proposed HEA has structure-and-parameter learning ability. That is, it can determine the average optima number of fuzzy rules and tune the free parameters in a TNFC. The proposed HEA also processes variable lengths of the chromosomes in a population.

3.2 Self Adaptive Groups Cooperation Based Symbiotic Evolution

Although the HEA can solve the problems about deciding the number of fuzzy rules, there still has a limitation of the proposed HEA. That is, all the fuzzy rules are encoded into one chromosome. In the HEA, partial solutions cannot be evaluated independently in the population. The partial solutions can be characterized as *specializations*. The specialization property ensures diversity and prevents a population from converging to suboptimal solutions. A single partial solution cannot “take over” a population since it must correspond with other specializations. For solving this problem, the self adaptive group cooperation based symbiotic evolution (SAGC-SE) is proposed.

In this section, the self adaptive group cooperation based symbiotic evolution (SAGC-SE) will be discussed. In the proposed SAGC-SE, the algorithm is developed from symbiotic evolution. Unlike the standard evolutionary approach which always causes a given population to converge, hopefully at the global optimum, the symbiotic evolution finds solutions in different, unconverted populations ([95]-[97]).

Although the symbiotic evolution uses the specialization property to ensure diversity,

which prevents a population from converging to suboptimal solutions, there still has a problem that the population cannot evaluate each partial solution locally. For example, the partial solutions are all encode into a population. In the evolutionary operator, the chromosomes with better fitness have most chance to keep in the offspring. Therefore, chromosomes of a population may become more similarly. The complete solution is hard to fine by combining of chromosomes in a population.

Recently, Gomez and Schmidhuber proposed lots of work to solve this problem ([40] and [41]). The proposed enforced sub-populations (ESP) used sub-populations of neurons for the fitness evaluation and overall control. As shown in [40] and [41], the sub-populations that use to evaluate the solution locally can obtain better performance compared to systems of only one population be used to evaluate the solution.

Same with the ESP, the SAGC-SE is proposed for solving the problem that that the population cannot evaluate each fuzzy rule locally. In the SAGC-SE, each chromosome represents only one fuzzy rule and an n-rules fuzzy system is constructed by selecting and combining n chromosomes from several groups. The SAGC-SE, which promotes both cooperation and specialization, ensures diversity and prevents a population from converging to suboptimal solutions. In SAGC-SE, compared with normal symbiotic evolution, there are several groups in a population. Each group formed by a set of chromosomes represents a fuzzy rule. Compare with the ESP, to let the well-performing groups of individuals to cooperate for generating better generation, an elite-based compensatory of crossover strategy (ECCS) is proposed. In the ECCS, each group will cooperate to perform the crossover steps. Therefore, the better chromosomes of each group will be selected to perform crossover in the next generation.

The proposed SAGC-SE consists of structure and parameter learning. In structure learning, as well as HEA, the SAGC-SE determines the number of fuzzy rules automatically and 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 TNFC. To deal with this, in the SAGC-SE, the numbers of rules in TNFCs are variable. The structure of the chromosome in the SAGC-SE is shown in Fig. 3.6. As shown in Fig. 3.6, each rule represents a chromosome that is selected from a group, P_{size} represents that there are P_{size} groups in a population, and "R_k" means that there are R_k rules used to construct a TNFC.

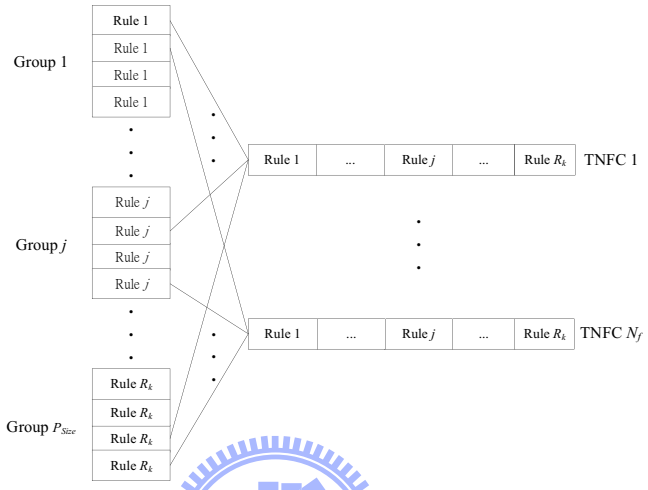


Figure 3.6: The structure of the chromosome in the SAGC-SE.

For determining the suitable number of fuzzy rules, in the SAGC-SE, the two-step self-adaptive algorithm (TSSA) is proposed. In the proposed TSSA, as well as HEA, the compact genetic algorithm (CGA) is adopted. The TSSA is different from the MCGA in that the building blocks (BBs) are used to represent the suitability of TNFCs with different numbers of fuzzy rules and determine the number of TNFCs with each different number of fuzzy rules to evaluate the chromosomes. As shown in Fig. 3.7, the TSSA codes the probability vector into the building blocks (BBs), where R_k represents R_k rules (chromosomes) that are used to form a TNFC, and V_{R_k} is a probability vector that represents the suitability of a TNFC with R_k fuzzy rules. In the TSSA, as well as MCGA, the maximum and minimum number of rules must be predefined to prevent the number of fuzzy rules from generating beyond a certain bound (i.e., $[R_{\min}, R_{\max}]$).

In parameter learning of the SAGC-SE, to allow groups to cooperate with each other for generating better solutions, an elite-based compensatory of crossover strategy (ECCS) is proposed. In SAGC-SE, the coding structure of chromosomes must be suitable for the concept of each chromosome represents only one fuzzy rule. A fuzzy rule with the form introduced in Eq. 2.2 is described in Fig. 3.8.

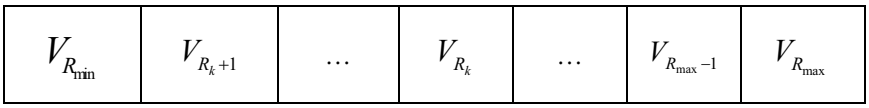


Figure 3.7: Coding the probability vector into the building blocks (BBs) in the TSSA.

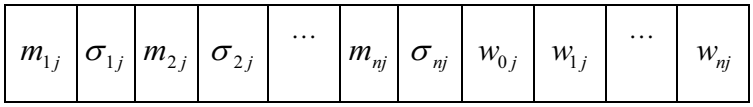


Figure 3.8: Coding a rule of a TNFC into a chromosome in SAGC-SE.

The learning process of the SAGC-SE in each group involves six major steps: initialization, fitness assignment, sorting, elite-based reproduction strategy (ERS), elite-based compensatory of crossover strategy (ECCS), and mutation strategy. The flowchart of the learning process is shown in Fig. 3.9 where *Nc* represents that there are *Nc* chromosomes in each group. The learning process is described step-by-step as follows:

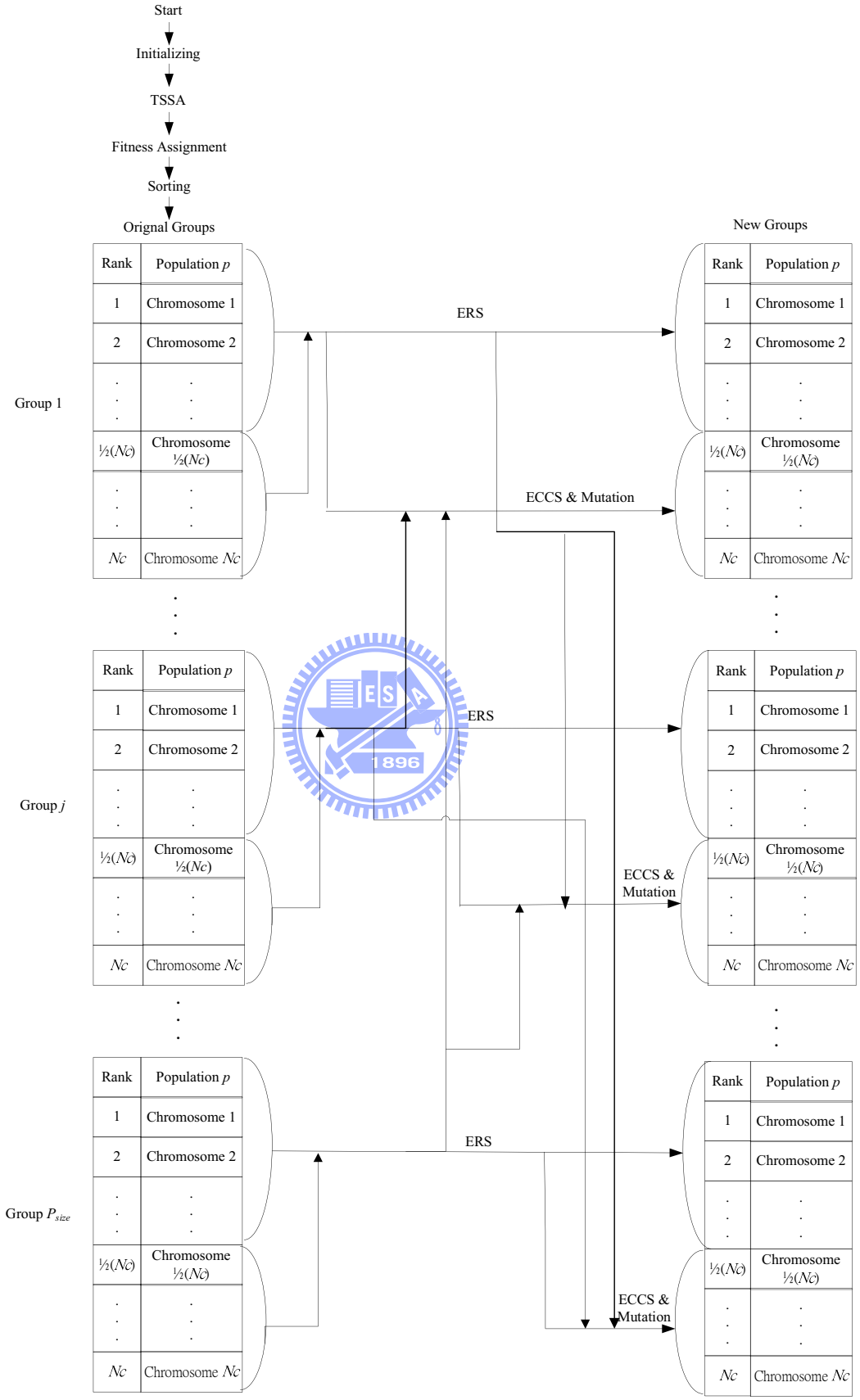


Figure 3.9: The learning process of SAGC-SE.

1. Initialization:

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

De*viation*: $Chr_{g,c}[p]=random[\sigma_{\min}, \sigma_{\max}]$

where $p=2, 4, \ldots, 2n$; $g=1, 2, \ldots, P_{size}$; $c=1, 2, \ldots, N_C$. (3.13)

Mean: $Chr_{g,c}[p]= random[m_{\min}, m_{\max}]$

where $p=1, 3, \ldots, 2n-1$. (3.14)

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

where $p=2n+1, 2n+2, \ldots, 3n+2$. (3.15)

where $Chr_{g,c}$ represents cth chromosome in gth group; P_{size} represents total number of groups 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 predefined range.

2. Two-step self-adaptive algorithm (TSSA):

After every group is initialized, the SAGC-SE uses the two-step self-adaptive algorithm (TSSA) to determine the suitable selection times of each number of rules (In the SAGC-SE, the numbers of rules are between $[R_{\min}, R_{\max}]$.). "Selection times" in the SAGC-SE indicates how many TNFCs should be generated in one generation. In the SAGC-SE, one chromosome represents only one fuzzy rule; several chromosomes are selected to combine and make a TNFC. The chromosomes should be evaluated and selected to construct TNFCs. The selection times using in the SAGC-SE are represented the number of TNFCs (a set of chromosomes) with different rules in one generation. In traditional symbiotic evolution ([95]-[97]) and ESP ([41] and [42]), the number of rules is predefined, so only the total selection

times are defined in all generations. However, in the SAGC-SE, the number of fuzzy rules is variable; therefore, the selection times with each different number of rules in the TNFCs in every generation must be determined to evaluate the performance of each group. To solve this problem, the TSSA is proposed to determine the suitable selection times (the number of TNFCs with different rules in a generation). The TSSA is a process that determines the number of TNFCs with R_k rules that are selected from R_k groups in every generation. The TSSA is similar to the maturing phenomenon in society, where individuals become more suited to society as they acquire more knowledge.

In the TSSA, the modified compact genetic algorithm (MCGA) is adopted. The MCGA adopts the building blocks (BBs) to represent the suitable length of chromosomes and reproduce the chromosomes according to the BBs. The TSSA is different from the MCGA in that the building blocks (BBs) are used to represent the suitability of TNFCs with different numbers of fuzzy rules and to determine how many TNFCs with each different number of fuzzy rules should be selected to evaluate the chromosomes in every generation. After the TSSA is carried out, the selection times of the suitable number of rules in a TNFC will increase, and the selection times of the unsuitable number of rules in a TNFC will decrease. The details of the TSSA are as follows:

Step 1. Initialize the probability vectors of the BBs:

$$\begin{aligned} &V_{R_k} = 0.5 \\ &\text{where } R_k = R_{\min}, R_{\min}+1, \cdots, R_{\max}, \end{aligned} \tag{3.16}$$

Step 2. Update the probability vectors of the BBs according to the following equations:

$$\begin{cases} V_{R_k} = V_{R_k} + (Upt\_value_{R_k} * \lambda), & \text{if} \quad Avg \le fit_{R_k} \\ V_{R_k} = V_{R_k} - (Upt\_value_{R_k} * \lambda), & \text{otherwise} \end{cases} \tag{3.17}$$

where $R_k = R_{\min}, \ R_{\min}+1, \ \cdots, \ R_{\max}$

$$Avg = \sum_{R_k = R_{\min}}^{R_{\max}} fit_{R_k} / (R_{\max} - R_{\min}) \tag{3.18}$$

$$Upt\_value_{R_k} = fit_{R_k} \Big/ \sum_{R_k = R_{\min}}^{R_{\max}} fit_{R_k} \tag{3.19}$$

$$\begin{aligned} &\text{if} \quad Fitness_{R_k} \ge (Best\_Fitness_{R_k} - ThreadFitnessvalue) \\ &\text{then} \quad fit_{R_k} = fit_{R_k} + Fitness_{R_k}; \end{aligned} \tag{3.20}$$

where V_{R_k} is the probability vector in the BBs and represents the suitability of TNFCs with R_k rules; λ is a predefined threshold value; *Avg* represents the average fitness value in the whole population; $Best\_Fitness_{R_k}$ represents the best fitness value of TNFCs with R_k rules; fit_{R_k} is the sum of the fitness values of the TNFCs with R_k rules when the fitness values of TNFCs with R_k rules greater than $Best\_Fitness_{R_k}$ minus a predefined threshold value named $ThreadFitnessvalue$. As shown in Eq. 3.17, if $fit_{R_k} \ge Avg$, then the suitability of TNFCs with R_k rules should be increased. On the other hand, if $fit_{R_k} < Avg$, then the suitability of TNFCs with R_k rules should be decreased.

Step 3. Determine the selection times of TNFCs with different fuzzy rules according to the probability vectors of the BBs as follows:

$$Rp_{R_k} = (Selection\_Times) * (V_{R_k} / Total\_Velocy)$$

$$\text{where} \quad R_k = R_{\min}, \ R_{\min}+1, \ \cdots, \ R_{\max} \tag{3.21}$$

$$Total\_Velocy = \sum_{R_k = R_{\min}}^{R_{\max}} V_{R_k} \tag{3.22}$$

where $Selection\_Times$ represents the total selection times in each generation; Rp_{R_k} represents the selection times of TNFCs with R_k rules that are selected from R_k group in a generation.

Step 4. After step 3, the selection times of TNFCs with different rules are obtained. Rp_{R_k} represents to select Rp_{R_k} TNFCs, and each TNFC selects R_k chromosomes from R_k groups in one generation.

Step 5. In the TSSA, as well as MCGA, to prevent suitable selection times from falling in the local optimal solution, the TSSA uses two different actions to update the V_{R_k}. The deferent actions are defined according to the following equations:

$$\text{if} \quad Accumulator \le TSSATimes \quad \text{then do Steps 1 to 3} \tag{3.23}$$

$$\text{if} \quad Best\_Fitness_g = Best\_Fitness \quad \text{then} \quad Accumulator = Accumulator + 1; \tag{3.24}$$

$$\text{if} \quad Accumulator > TSSATimes \quad \text{then do Step 1 and} \quad Accumulator = 0. \tag{3.25}$$

3. Fitness assignment:

As premised, in SAGC-SE, the fitness value of a single rule (an individual) is calculated by summing up the fitness values of all the possible combinations which contains that single rule. The details for assigning the fitness value are described step by step as follows:

Step 1. Randomly choose R_k fuzzy rules from the R_{max} groups with size N_C to form a

TNFC Rp_{R_k} times. The R_k represents the number of fuzzy rules. The numbers of fuzzy rules are variable in SAGC-SE.

Step 2. Evaluate every TNFC with R_k rules, which is generated from step1, to obtain a fitness value.

Step 3. Divide the fitness value by R_k and accumulate the divided fitness value to the selected rules with their fitness value records are set for zero initially.

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

4. Sorting:

After the Fitness assignment step, the chromosomes in each group are sorted. This step can improve the reproduction step because the well-performing chromosomes in each group will be kept. After the chromosomes in each group were sorted, the algorithm goes to the next step.

5. Elites-based Reproduction Strategy (ERS):

Reproduction is a process in which individuals are copied according to their fitness values. A fitness value is assigned to each chromosome according to a fitness assignment step in which a high value denote a good fit. The goal of the SAGC-SE is to maximize the fitness value. For keeping the stability, an elite-based reproduction strategy (ERS) is proposed to allow the best combination of chromosomes can be kept in the next generation. In SAGC-SE, the chromosome with the best fitness value may not be in the best combination. Therefore, every chromosome in the best combination must be kept by applying ERS. Other chromosomes in each group, as well as HEA, are selected under roulette-wheel selection method ([72]). The best performing chromosomes in the top half of each group ([96]) advance to the next

generation. The other half is generated by applying crossover and mutation operations on chromosomes in the top half of the parent generation. In the reproduction step, the top half of each group must be kept the same number of chromosomes.

6. Elite-based compensatory of crossover strategy (ECCS):

Although the ERS 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 step working on the parents is the crossover step, which occurs on a selected pair under a crossover rate. Therefore, an elite-based compensatory of crossover strategy (ECCS) is proposed to improve the crossover operation. The ECCS mimics the cooperation phenomenon in society, in which individuals become more suitable for the environment as they acquire and share more knowledge of their surroundings. The best performing individuals in the top half of each group that are called elites are used to select the parents for applying the ECCS. Details of the ECCS are shown below.

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

$$Fitness\_Ratio_{g,t} = \frac{\sum_{u=1}^{t} fitness_{g,u}}{\sum_{c=1}^{Nc} fitness_{g,c}}, \quad \text{where} \quad t = 1,\ 2,\ \cdots,\ Nc; \tag{3.26}$$

$$Rand\_Value[g] = Random[0,1], \quad \text{where} \quad g = 1,\ 2,\ \cdots,\ P_{size}; \tag{3.27}$$

$$Parent\_SiteA[g] = t, \quad \text{if} \quad Fitness\_Ratio_{g,t-1} < Rand\_Value[g] \le Fitness\_Ratio_{g,t}, \tag{3.28}$$

where $Fitness\_Ratio_{g,t}$ is a fitness ratio of tth chromosome in the gth group;

$Rand\_Value[g] \in [0,\ 1]$ is a random value of gth group; $Parent\_SiteA[g]$ is the site of the first parent. According to Eq. 3.28, if the $Rand\_Value[g]$ is greater than the fitness ratio at (t-1)th chromosome in gth group and equal to or smaller than 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 best performing elites in every group is used to determine the other parent. In this step, the total fitness ratio of every group is computed as follows:

$$Total\_Fitness_g = \sum_{c=1}^{Nc} fitness_{g,c}, \quad \text{where} \quad g = 1,\ 2,\ \cdots,\ P_{size}; \tag{3.29}$$

$$Total\_Fitness\_Ratio_q = \frac{\sum_{b=1}^{q} Total\_Fitness_b}{\sum_{g=1}^{R_{\max}} Total\_Fitness_g}, \tag{3.30}$$

$$\text{where} \quad q = 1,\ 2,\ \cdots,\ P_{size};$$

where $Total\_Fitness_g$ represents the summation of all chromosomes' fitness value in gth group; $Total\_Fitness\_Ratio_q$ is a total fitness ratio of qth group.

Step 3. Determine the other parental group for applying crossover with the $Parent_SiteA[g]$th chromosome according to the following equations:

$$Group\_Rand\_Value[g] = Random[0,1] \quad \text{where} \quad g = 1,\ 2,\ \cdots,\ P_{size}; \tag{3.31}$$

$$Parent\_Group\_SiteB[g] = q, \quad \text{if} \tag{3.32}$$

$$Total\_Fitness\_Ratio_{q-1} < Group\_Rand\_Value[g] \le Total\_Fitness\_Ratio_q.$$

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

Step 4. After the $Parent_Group_SiteB[g]$th group is selected, the other parent which is selected from $Parent_Group_SiteB[g]$th group is determined by ECCS according to the following equations:

$$Fitness\_Ratio_{Selected\_g,t} = \frac{\sum_{b=1}^{t} fitness_{Selected\_g,b}}{\sum_{c=1}^{Nc} fitness_{Selected\_g,c}}, \tag{3.33}$$

where $t = 1,\ 2,\ \cdots,\ Nc;\ Selected\_g = Parent\_Group\_SiteB[g];$

$$Rand\_Value[g] = Random[0,1], \quad \text{where} \quad g = 1,\ 2,\ \cdots,\ P_{size}; \tag{3.34}$$

$$Parent\_SiteB[g] = l, \quad \text{if} \quad Fitness\_Ratio_{Selected\_g,l-1} < Rand\_Value[g] \le Fitness\_Ratio_{Selected\_g,l}, \tag{3.35}$$

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

After selecting the parents from the gth group and $Parent_Group_SiteB[g]$th group by ECCS, the individuals ($Parent_SiteA[g]$th chromosome and the $Parent_SiteB[g]$th chromosome) are crossed and separated by using a two-point crossover ([76]) in the gth group, as shown in Fig. 3.10. In Fig. 3.10, exchanging the site's values between the selected sites of parents' individuals creates new individuals. After this operation, the individuals with poor performances are replaced by the newly produced offspring.

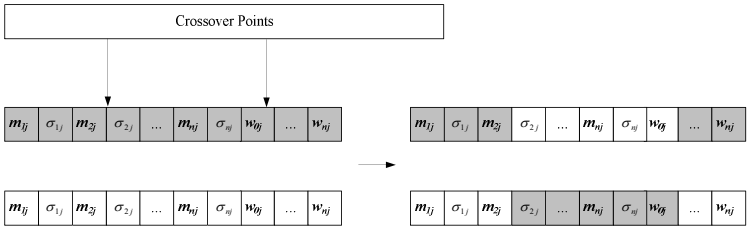


Figure 3.10: Two-point crossover.

7. Mutation strategy:

   For emphasizing the capability of the ECCS, the proposed SAGC-SE tries to simplify the mutation operation. Therefore, a uniform mutation ([84]) is adopted, and the mutated gene is generated randomly from the domain of the corresponding variable.

The aforementioned steps are done repeatedly and stopped when the predetermined condition is achieved. As mention above, a self adaptive group cooperation based symbiotic evolution (SAGC-SE) is proposed for considering the local evaluation of the population. The SAGC-SE can determine the suitable number of fuzzy rules and evaluate the fuzzy rule locally. Moreover, the SAGC-SE can make groups to cooperate with each other for generating the better chromosomes by using an elites-base compensation crossover strategy (ECCS).

3.3 Self adaptive Groups Based Symbiotic Evolution using FP-growth Algorithm

Although the SAGC-SE could solve the problem of the HEA that all the fuzzy rules are encoded into one chromosome. Moreover the SAGC-SE not only evaluates the fuzzy rule locally but also makes groups to cooperate with each other for generating the better chromosomes. However, in the SAGC-SE, how to select groups to choose individuals for constructing a TNFC with different number of rules is a major problem. Therefore, for determining the number of fuzzy rules automatically, the SAGC-SE selects different number of groups to construct complete solution. In this way, the SAGC-SE selects groups randomly. It's obvious that the performance of the SAGC-SE dependents on how to select individuals from groups.

In this section, the self adaptive groups based symbiotic evolution using FP-growth algorithm (SAG-SEFA) will be discussed. The SAG-SEFA is proposed for providing a

method of how to select groups to select individuals for constructing a TSK-type neuro-fuzzy controller (TNFC) with different number of rules in the SAGC-SE. Therefore, the SAG-SEFA is used to determine the suitable number of rules in a TNFC and the suitable groups used to perform the selection of groups. Moreover, the SAG-SEFA adopts a different way to select suitable groups to perform crossover steps.

The SAG-SEFA is proposed to improve the SAGC-SE. The purpose of the SAG-SEFA is to determine not only the suitable number of rules in a TNFC but also the suitable rules that are used to construct a TNFC. Therefore, the SAG-SEFA, as well as the SAGC-SE, consists of structure and parameter learning.

In structure learning, as well as SAGC-SE, the SAG-SEFA determines the number of fuzzy rules automatically and processes the variable length of a combination of chromosomes by using the TSSA.

In parameter learning, to solve the problem of the SAGC-SE that the chromosomes are selected randomly to perform selection step. The proposed SAG-SEFA determines which suitable groups should be selected the chromosomes that will form TNFCs with different rules and which suitable groups that should be selected to perform selected step. Furthermore, the SAG-SEFA also provides a different way to determine the suitable groups to perform crossover step. The SAG-SEFA proposes using the data mining based selection strategy (DMSS) and the data mining based crossover strategy (DMCS) to determine which groups should be used to select individuals to form a TNFC with each different rules and to determine which groups should be used to select individuals to perform crossover steps using the frequent pattern growth (FP-growth) ([50]) data mining method.

The goal of FP-growth is to find the frequent patterns that do not have candidate generation. In the proposed DMSS, the FP-growth is used to find from transactions the sets of groups that occur frequently. In SAG-SEFA, a “transaction” refers to the collection of groups that perform well. After the candidate sets of frequently-occurring groups have been found,

the DMSS uses three actions to determine R_k groups that are used to select R_k chromosomes to form TNFCs with R_k rules. The three actions defined in the DMSS are normal, search, and exploration. In the normal action, as well as SAGC-SE, R_k groups that are used to select R_k chromosomes to form a TNFC are chosen randomly. In the search action, R_k groups are chosen from the set of frequently-occurring groups which chosen from the candidate sets of frequently-occurring groups. In the exploration action, R_k groups are chosen without using the set of frequently-occurring groups. As well as the DMSS, in the DMCS, the suitable groups used to select chromosomes to perform the crossover steps are decided based on the three actions (normal, search, or exploration). Compare with SAGC-SE, the SAG-SEFA provides a robust way to select groups to perform selection and crossover step. Therefore, the three actions (normal, search, or exploration) can improve the combination of solutions to avoid fall in the local optimal solution.

The structure of the chromosome in the SAG-SEFA, as well as SAGC-SE, is shown in Fig. 3.6. In the proposed SAG-SEFA, as well as the SAGC-SE, the coding structure of the chromosomes must be suitable for symbiotic evolution. The coding structure is shown in Fig. 3.8.

For determining the suitable number of fuzzy rules, the two-step self-adaptive algorithm (TSSA) proposed in SAGC-SE is adopted. As well as SAGC-SE, the building blocks (BBs) are used to represent the suitability of TNFCs with different number of fuzzy rules and to determine to the number of TNFCs with each different fuzzy rules should be selected to evaluate the chromosomes. The TSSA codes the probability vector into the building blocks (BBs) is shown in Fig. 3.7. In the TSSA, the maximum and minimum number of rules must be predefined to prevent the number of fuzzy rules from generating beyond a certain bound (i.e., $[R_{\min}, R_{\max}]$).

The learning process of the SAG-SEFA is shown in Fig. 3.11. As shown in Fig. 3.11, each group involves eight major operators: the initialization, two-step self-adaptive algorithm

(TSSA), data mining based selection strategy (DMSS), fitness assignment, sorting, elite-based reproduction strategy (ERS), data mining-based crossover strategy (DMCS), and the mutation strategy. In the SAG-SEFA, the operators of the initialization, two-step self-adaptive algorithm (TSSA), sorting, elite-based reproduction strategy (ERS), and the mutation strategy are same as the SAGC-SE introduced in Section 3.2. About this, the only three operators of the data mining based selection strategy (DMSS), fitness assignment, and data mining-based crossover strategy (DMCS) are described step-by-step as follows:

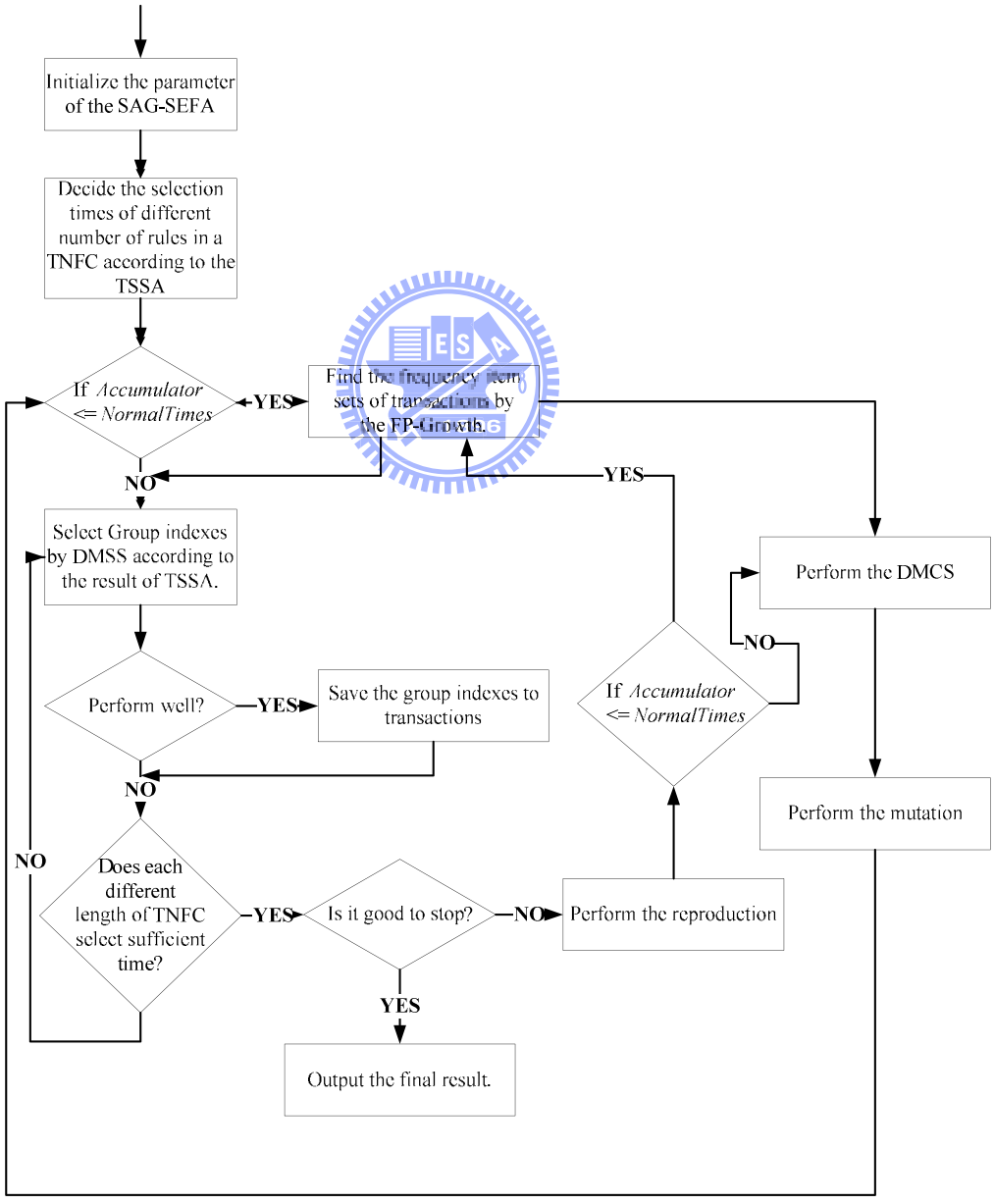


Figure 3.11: Learning process of the SAG-SEFA.

1. The data mining based selection strategy (DMSS):

After the TSSA, the selection times of the TNFCs with different rules are determined. The SAG-SEFA then performs the selection step. The selection step in the SAG-SEFA can be divided into the selection of groups and the selection of chromosomes. In the selection of groups, the data mining-based selection strategy (DMSS) is proposed to improve the selection of the SAGC-SE in which chromosomes are selected randomly to form TNFCs. In the DMSS, the groups are selected according to the groups that frequently obtain the best performance. To defend the groups that frequently obtain the best performance, the FP-growth ([50]) data mining method is adopted. The FP-growth was proposed by Han et al. ([50]). The goal of FP-growth is to find the frequently-occurring patterns that do not have candidate generation. In the proposed DMSS, the FP-growth is used to find the frequently-occurring groups from transactions (in the SAG-SEFA, a transaction means a set of the groups that performs well). After the groups that occur frequently have been found, the DMSS selects the R_k groups that are used to select chromosomes to form TNFCs with R_k rules according to the frequently-occurring groups. To avoid the frequently-occurring groups that may fall in the local optimal solution, the DMSS uses three actions to select R_k groups. The three actions defined in this paper are normal, search, and exploration. The details of the DMSS are as follows:

Step 1. The transactions are built in the following equation:

$$\begin{aligned}
&\text{if} \quad Fitness_{R_k} \geq (Best_Fitness_{R_k} - ThreadFitnessvalue) \\
&\text{then} \quad Transaction_j[i] = TNFCRuleSet_{R_k}[i] \\
&\text{where} \quad i = 1,\ 2,\ \cdots, R_k; \\
&\qquad R_k = R_{\min},\ R_{\min}+1,\ \cdots,\ R_{\max}; \\
&\qquad j = 1,\ 2,\ \cdots,\ TransactionNum.
\end{aligned} \tag{3.36}$$

where the $Fitness_{R_k}$ represents the fitness value of TNFC with R_k rules; $ThreadFitnessvalue$ is the predefined value; $TransactionNum$ is the total number of transactions; $Transaction_j[i]$ represents the *i*th item in the *j*th transaction; and $TNFCRuleSet_{R_k}[i]$ denotes the *i*th group of the selected R_k groups used to select chromosomes to form a TNFC with R_k rules. The transactions have the form shown in Table 3.1. As shown in Table 3.1, every transaction represents the R_k groups that form a TNFC with R_k rules. For example, as shown in Table 3.1, the first transaction of the transaction set means that the 3-rule TNFC that is selected from the first group, fourth group, and eighth group performs well. The step of building transactions continues in the normal, search, and exploration actions.

Table 3.1: Transactions in a FP-growth.

Transaction index	Groups
1	1, 4, 8
2	2, 4, 7, 10
…	…
TransactionNum	1, 3, 4, 6, 8, 9

Step 2. Normal action:

After the transactions are built, the DMSS selects groups according to different action types. If the action type is normal, the DMSS selects the groups, using the

following equation:

$$\begin{aligned} &\text{if} \quad Accumulator \le NormalTimes \\ &\text{then} \quad GroupIndex[i] = Random[1,\ P_{Size}]; \\ &\text{where} \quad i = 1,\ 2,\ \cdots,\ R_k; R_k = R_{\min},\ R_{\min}+1,\ \cdots,\ R_{\max}. \end{aligned} \tag{3.37}$$

where *Accumulator* defined in Eq. 3.23-3.25 is used to determine what action should be adopted; *GroupIndex*[*i*] represents the selected *i*th group of the R_k groups; and P_{Size} indicates that there are P_{Size} groups in a population in the SAG-SEFA. In this action, the algorithm is used to accumulate the transaction set. Therefore, the groups that perform well will be stored in a transaction if the groups fit Eq. 3.36. If the best fitness value does not improve for a sufficient number of generations (*NormalTimes*), the DMSS selects the groups according to another action type (which go to the next steps).

Step 3. Find the groups that occur frequently:

If the action is the search or exploration action (the *Accumulator* exceeds the *NormalTimes*), the DMSS uses FP-growth to find the groups that occur frequently in transactions. The frequently-occurring groups are found according to the predefined *Minimum_Support*. *Minimum_Support* represents the minimum fraction of transactions that contain an item set. After *Minimum_Support* is defined, data mining using FP-growth will be performed. The FP-growth algorithm can be viewed as having two parts: construction of the FP-tree and FP-growth. The sample transactions shown in Table 3.2 are given as examples. *Minimum_Support*=3 is considered in this example.

(1) Construction of FP-tree:

To construct the FP-tree, the first step is to scan the transactions and retrieve the frequent 1-groupset in transactions. The frequent 1-groupset

represents the set of one group which has support counts bigger than *Minimum_Suppor* in transactions. The result is shown in Table 3.3. Then the retrieved frequently-occurring groups are ordered by descending order based on their supports, as shown in Table 3.4. The ordered list in Table 3.4 is called the F-list. After the F-list is obtained, the next step is to discard the infrequently-occurring groups and sort the remaining groups in the same order as in the F-list in each transaction. The result is shown in Table 3.5. The ordered transactions are then used to construct the FP-tree. The steps for constructing the FP-tree are illustrated in Fig. 3.12 (a). In Fig. 3.12 (a), 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. Then the complete 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. 3.12 (b). This FP-tree is constructed from the transactions shown in Table 3.2.

(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, 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. Then accumulate the count for each group in the base to construct the conditional FP-tree of the corresponding suffix group. After mining the frequently-occurring groups on the conditional

FP-tree, FP-growth data mining is completed by the concatenation of the suffix group with the generated frequently-occurring groups. The groups generated by the FP-growth, shown in Table 3.6, are then thrown into the pool that is called *FrequentPool*. *FrequentPool* represents the candidate sets of the frequently-occurring groups.

Table 3.2: 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 3.3: Frequent 1-groupset of sample transactions.

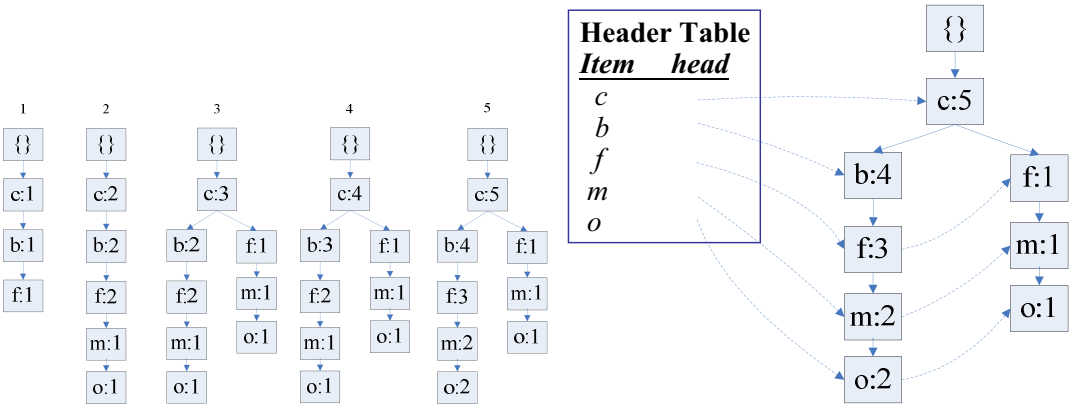
Group name	count	Group name	count
B	4	M	3
C	5	O	3
F	4		

Table 3.4: F-list of sample transactions.

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

Table 3.5: 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}



(a) (b)

Figure 3.12: (a) Steps for constructing the FP-tree of sample transactions. (b) FP-tree of sample transactions.

Table 3.6: Frequently-occurring groups generated by FP-growth 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

Step 4. Select the groups according to two different actions:

After the groups that frequently occur are identified, the DMSS selects groups according to two different actions as follows:

(1) In the search action, the groups are selected from the items that frequently occur by using the following equation:

if $NormalTimes < Accumulator \leq SearchingTimes$

then $GroupIndex[i] = w$,

where

$$w = Random[1, P_{Size}] \text{ and } w \in FrequentItemSet[q]; \quad (3.38)$$

$$FrequentItemSet[q] = Random[FrequentPool];$$

$$q = 1,\ 2,\ \cdots,\ FrequentPoolNum;$$

$$i = 1,\ 2,\ \cdots,\ R_k; R_k = R_{\min},\ R_{\min}+1,\ \cdots,\ R_{\max},$$

where $SearchingTimes$ is a predefined value that is used to judge whether the search action needs to be taken or not; $FrequentPool$ represents the candidate sets of groups that frequently occur and which are obtained from FP-growth; $FrequentPoolNum$ represents the total number of sets in $FrequentPool$; and $FrequentItemSet[i]$ represents a set of frequently-occurring groups selected from $FrequentPool$ randomly. In Eq. 3.38, if R_k is greater than the size of $FrequentItemSet[q]$, the remaining groups are selected by using Eq. 3.37. In this action, the algorithm tries to search for a better solution in the set of solutions that performs well frequently. Therefore, the groups are selected according to the candidate groups that perform well frequently. In this action, the groups that perform well will also be stored in a transaction if the groups fit Eq. 3.36. If the best fitness value does not improve for a sufficient number of generations (*SearchingTimes*), the DMSS selects groups according to the exploration action type.

(2) In the exploration action, the groups are selected according to the groups that occur frequently by using the following equation:

if $SearchingTimes < Accumulator \leq ExploringTimes$

then $GroupIndex[i] = w$,

where

$$w = Random[1, PulationSize] \quad \text{and} \quad w \notin FrequentItemSet[i]; \tag{3.39}$$

$$FrequentItemSet[i] = Random[FrequentPool];$$

$$i = 1,\ 2,\ \cdots,\ R_k;\ R_k = R_{\min},\ R_{\min} + 1,\ \cdots,\ R_{\max},$$

where $ExploringTimes$ is a predefined value that is used to judge whether the exploration action needs to be taken or not. In this action, the algorithm tries to find a better solution without using the solutions that perform well frequently. This is because when the candidate groups that perform well frequently do not improve for a sufficient number of generations, the candidate groups may fall in the local optimal solutions. Therefore, the groups are selected without the candidate groups that perform well frequently. In this action, the groups that perform well will also be stored in a transaction if the groups fit Eq. 3.36. If the best fitness value does not improve for a sufficient number of generations (*ExploringTimes*), the DMSS selects groups according to the normal action type. In the SAG-SEFA, *ExploringTimes* is equal to *TSSATimes* of the TSSA.

Step 5. After the R_k groups are selected, the R_k chromosomes are selected from the R_k groups as the following equation:

$$\begin{aligned} &ChromosomeIndex[i] = q, \\ &\text{where} \\ &\qquad q = Random[1,\ N_c]; \\ &\qquad i = 1,\ 2,\ \cdots,\ R_k, \end{aligned} \tag{3.40}$$

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 *i*th group.

2. Fitness assignment:

As previously stated, for the SAG-SEFA, the fitness value of a rule (an individual) is calculated by summing up the fitness values of all the possible combinations in the chromosomes that are selected from the R_k groups that are decided according to the DMSS. The details for assigning the fitness value are described step by step below:

Step 1. Choose R_k fuzzy rules to construct a TNFC Rp_{R_k} times from the R_k groups with size N_C. The R_k groups are obtained from the DMSS.

Step 2. Evaluate every TNFC that is generated from step1 to obtain a fitness value.

Step 3. Divide the fitness value by R_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 the R_k groups by the number of times it has been selected. The average fitness value represents the performance of a rule.

3. The data mining based crossover strategy (DMCS):

In the SAG-SEFA, the data mining based crossover strategy (DMCS) is proposed to perform the crossover operation. The DMCS mimics the cooperation phenomenon in society, in which individuals become more suited to the environment as they acquire and share more knowledge of their surroundings. Similar to the DMSS, the DMCS uses FP-growth to select the parental groups to perform crossover operations in the next generation. Moreover, the DMCS also uses threes actions to select parental groups from the *FrequentPool* according to the set of frequently-occurring groups. The best performing individuals in the top half of the selected parental groups that are called elites are used to select the parents for

performing with the DMCS. Details of the DMCS are given 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:

$$Fitness\_Ratio_{g,\,t} = \frac{\sum_{u=1}^{t} fitness_{g,\,u}}{\sum_{c=1}^{Nc} fitness_{g,\,c}}, \quad \text{where} \quad t = 1,\ 2,\ \cdots,\ Nc; \tag{3.41}$$

$$Rand\_Value[g] = Random[0,1], \quad \text{where} \quad g = 1,\ 2,\ \cdots,\ P_{Size}; \tag{3.42}$$

$$Parent\_SiteA[g] = t, \quad \text{if} \quad Fitness\_Ratio_{g,\,t-1} < Rand\_Value[g] \le Fitness\_Ratio_{g,\,t}, \tag{3.43}$$

where $Fitness\_Ratio_{g,\,t}$ is a fitness ratio of the fitness value of the tth chromosome in the gth group; $Rand\_Value[g] \in [0,\ 1]$ is the random values of the gth group; $Parent\_SiteA[g]$ is the site where the first parent is. According to Eq. 3.43, if the $Rand\_Value[g]$ is greater than the fitness ratio at the (t-1)th chromosome in the gth group and smaller or equal to the fitness ratio at the tth chromosome in the gth group, the site of the first parent of the gth group is assigned to t.

Step 2. After the first parent is determined, the second parental group is decided according to different actions as follows:

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

$$Total\_Fitness_{g} = \sum_{c=1}^{Nc} fitness_{g,\,c}, \quad \text{where } g = 1, 2, \cdots, P_{Size}; \tag{3.44}$$

$$Total\_Fitness\_Ratio_q = \frac{\sum_{b=1}^{q} Total\_Fitness_b}{\sum_{g=1}^{P_{Size}} Total\_Fitness_g}, \quad \text{where } q = 1, 2, \cdots, P_{Size}, \tag{3.45}$$

where $Total\_Fitness_g$ represents the summation of the fitness value of every chromosome in the gth group and $Total\_Fitness\_Ratio_q$ is a total fitness ratio of the qth group. After the total fitness ratio is computed, the group from which the chromosome is selected to be the other parent to perform crossover with the $Parent_SiteA[g]$th chromosome in the gth group is determined according to the following equations:

$$Group\_Rand\_Value[g] = Random[0, 1] \quad \text{where} \quad g = 1, 2, \cdots, P_{Size}; \tag{3.46}$$

$$Parent\_Group\_SiteB[g] = q, \quad \text{if} \tag{3.47}$$

$$Total\_Fitness\_Ratio_{q-1} < Group\_Rand\_Value[g] \le Total\_Fitness\_Ratio_q,$$

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

(2) In the search action, the second parent is decided according to the following equations:

$$FrequentItemSet[q] = Random[FrequentPool]$$
$$\text{where} \quad q = 1, 2, \cdots, FrequentPoolNum; \tag{3.48}$$

$$Parent\_Group\_SiteB[g] = w, \quad \text{if} \quad w \in FrequentItemSet[q]. \tag{3.49}$$

(3) In the exploration action, the second parent is decided according to the following equations:

$$FrequentItemSet[q] = Random[FrequentPool]$$
$$where \quad q = 1, 2, \cdots, FrequentPoolNum; \tag{3.50}$$

$$Parent\_Group\_SiteB[g] = w, \quad \text{if} \quad w \notin FrequentItemSet[q]. \tag{3.51}$$

Step 3. After the $Parent_Group_SiteB[g]$th group is selected, the DMCS selects the other parents in the selected $Parent_Group_SiteB[g]$th group according to the following equations:

$$Fitness\_Ratio_{Selected\_g,\ t} = \frac{\sum_{b=1}^{t} fitness_{Selected\_g,\ b}}{\sum_{c=1}^{Nc} fitness_{Selected\_g,\ c}}, \tag{3.52}$$

where $t = 1,\ 2,\ \cdots,\ Nc;\ Selected\_g = Parent\_Group\_SiteB[g];$

$$Rand\_Value[g] = Random[0,\ 1], \quad \text{where} \quad g = 1,\ 2,\ \cdots,\ P_{Size}; \tag{3.53}$$

$$Parent\_SiteB[g] = l, \quad \text{if} \quad Fitness\_Ratio_{Selected\_g,\ l-1} < Rand\_Value[g] \le Fitness\_Ratio_{Selected\_g,\ l}, \tag{3.54}$$

where $Fitness\_Ratio_{Selected\_g,\ t}$ is a fitness ratio of the fitness value of the 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 from the gth group and the $Parent_Group_SiteB[g]$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 ([76]) in the gth group, as shown in Fig. 3.10.

The aforementioned steps are done repeatedly and stopped when the predetermined condition is achieved. In this section, a SAG-SEFA with a TNFC is proposed. The SAG-SEFA has structure and parameter learning ability. That is, it can determine the suitable number of fuzzy rules and efficiently tune the parameters in the TNFC. The goal of using the SAS-SEFA is to determine the suitable groups for performing the selection and crossover steps for improving the problem of the SAGC-SE. The DMSS and DMCS are proposed to select the suitable groups for performing the selection and crossover steps.

Chapter 4
Improved Safe Reinforcement Learning

In this chapter, the other part of the proposed ISRL-SAEAs is introduced. Therefore, improved safe reinforcement learning (ISRL) is discussed. In this dissertation, the self adaptive evolution algorithms (SAEAs) are trained by using the ISRL. The ISRL will be introduced in the following sections. In Section 4.1, the safe reinforcement learning that the ISRL is based is introduced. The safe reinforcement learning is based on Lyapunov function design ([32]). Once the system's Lyapunov function is identified, under Lyapunov-based manipulations on control laws, the architecture can drive the plant to reach and remain in a predefined desired set of states with probability 1. The structure of the ISRL is introduced in Section 4.2. Therefore, the schematic diagram and flowchart of the ISRL are introduced. Moreover, the Lyapunov function used in ISRL is also introduced. In the final section, the fitness function of the SAEAs is introduced. Therefore, two strategies of the ISRL are discussed. There are two strategies in the ISRL, judgment and evaluation strategies. The judgment strategy measures the fitness value of controller that fails to guide the system into the goal set. The evaluation strategy measures the fitness value of controller that successfully guide the system into the goal set. In the ISRL, the control laws of the system are designed according to Lyapunov function. However, for different control systems, different control laws of the system are needed. Therefore, the Lyapunov-based manipulations on control laws are defined in Chapter 5.

4.1 Safe Reinforcement Learning

Although supervised learning is a powerful training technique that can be applied to networks. However, if the precise training data can be obtained easily, the supervised learning algorithm may be efficient in many applications. For some real-world applications, precise training data are usually difficult and expensive to obtain. For this reason, there has been a growing interest in reinforcement learning problems ([15]-[21]). For the reinforcement learning problems, training data are very rough and coarse and there are only "evaluative" when compared with the "instructive" feedback in the supervised learning problem.

Unlike the supervised learning problem, in which the correct "target" output values are given for each input pattern to perform the fuzzy controller learning, the reinforcement learning problem has only very simple "evaluative" or "critical" information, rather than "instructive" information, available for learning. In the extreme case, there is only a single bit of information to indicate whether the output is right or wrong. To solve reinforcement learning problems, the most popular approach is temporal difference (TD) reinforcement learning ([17]). The well-known TD-based reinforcement learning is the adaptive heuristic critic (AHC). The AHC consists of an action network and an evaluation network. Based on the AHC, in [17], Barto and his colleagues proposed an actor-critic architecture which consists of a control network and a critic network. However, the Barto's architecture is complicated and is not easy to implement. For solving this problem, several researches proposed time-step reinforcement architecture to improve the Barto's architecture ([18]-[20]). In [18]-[20], the time-step reinforcement architecture has a structure in which the only available feedback is a reinforcement signal that notifies the model only when a failure occurs. An accumulator accumulates the number of time steps before a failure occurs. The goal of the time-step reinforcement method is to maximize the value function V. The fitness function is defined by Eq. 2.9. Equation 2.9 reflects the fact that long-time steps before a failure occurs

(to keep the desired control goal longer) mean the controller can control the plat well. For example, in evolutionary algorithm, Eq. 2.9 reflects the fact that long-time steps before a failure occurs mean higher fitness of the evolutionary algorithm. As shown in [18]-[20], time-step reinforcement architecture is simpler and easier to implement than [17].

Even though time-step reinforcement architecture is easier to implement when compared with Barto's architecture, it can only measure the number of time steps before a failure occurs; in other words, it only evaluates how long the system can enter the desired state, which is also very important. Moreover, Eq. 2.9 of the time-step reinforcement architecture only indicates the system does not perform out of the boundaries. Therefore, the system could not evaluate how well the system controls the plant. For example, in the ball and beam balance system, Eq. 2.9 of the time-step reinforcement architecture represents how long before the beam deviates beyond a certain angle or the ball reaches the end of the beam. However, the system cannot evaluate how well the plant controls the ball near the center of the beam. Recently, Perkins and Barto proposed a safe reinforcement learning based on Lyapunov function design ([32]). Once the system's Lyapunov function is identified, under Lyapunov-based manipulations on control laws, the architecture can drive the plant to reach and remain in a predefined desired set of states with probability 1. The purpose of [32] is to guide the system to reach and remain in a set of goal states. Several properties defined in [32] are listed below to express a safety constraint that the controller uses to satisfy. Let S denotes the state set, $T \subset S$ and G denotes the set of goal states.

Property 4.1 (Remain in T) With probability 1, if $s_0 \in T$, then $s_t \in T$, for $\forall t \geq 0$.

Property 4.2 (Reach T) With probability 1, $\exists t \geq 0$ such that $s_t \in T$.

Property 4.3 (Reach and Remain in T) With probability 1, $\exists \tau \geq 0$ such that $s_t \in T$ for $\forall t \geq \tau$.

Property 4.4 (Converge in T) With probability 1, $\lim_{t \to \infty} \delta_T(s_t) = 0$, where $\delta_T : S \to \mathbb{R}^+$ is a distance-to-T function which satisfies $\delta_T(s) = 0$ for $s \in T$ and $\delta_T(s) > 0$ for $s \notin T$.

In [32], when $T=G$, Properties 4.1, 4.3, 4.4 represent of an achievement of stability. Lyapunov's direct method is mainly used to study the stability of systems of differential equations. Authors in [32] extend this method to the reinforcement learning problem. If the controller is designed such that it reduces the Lyapunov function of the plant in each time step, the controller could reach and remain in the goal sets.

4.2 Structure of the ISRL

Although safe reinforcement learning can let the control system to reach and remain in the goal set, it cannot evaluate how soon the system meets the control goal. The system using safe reinforcement learning only can make sure the control system to reach and remain in the goal set. However how soon the control system reaches the goal set is not considered in safe reinforcement learning. It is important to indicate the control system how soon to reach the goal set. For solving above problem, in this dissertation, the improved safe reinforcement learning is proposed. In this section, the other part of the proposed ISRL-SAEAs, that is, improved safe reinforcement learning (ISRL) is discussed. In this dissertation, the self adaptive evolution algorithms (SAEAs) are trained by using the improved safe reinforcement learning (ISRL).

As shown in safe reinforcement, once the system's Lyapunov function is identified, under Lyapunov-based manipulations on control laws, the architecture can drive the plant to reach and remain in a predefined desired set of states with probability 1. About this, in the proposed ISRL, the time step for the plant entering the desired set of states can be modified to indicate the concept of how soon the system becomes stable.

In the proposed ISRL, a reinforcement signal is designed based on Lyapunov function. The purpose of ISRL is to guide the system to reach and remain in a set of goal states. Several properties defined in [32] are listed above to express a safety constraint that we want the

controller to satisfy. Therefore, the improved safe reinforcement learning with self adaptive evolution algorithms (ISRL-SAEAs), which are constructed on a TNFC, are based on Lyapunov analysis. The schematic diagram of the ISRL-SAEAs is shown in Fig. 4.1. The TNFC acts as a control network to determine a proper action according to the current input vector (environment state). The feedback signal in Fig. 4.1 is the reinforcement fitness value that plays a role as a performance measurement. The reinforcement fitness value is evaluating how soon the plant can meet the desired set of states. The reinforcement fitness value is also used as the fitness function of the SAEAs. Each string with higher fitness value represents the better-fitted individual in the population. It will be observed that the advantage of the proposed ISRL-SAEAs is that its capability of meeting global optimum.

The flowchart of the ISRL-SAEAs is shown in Fig. 4.2. The proposed ISRL-SAEAs runs in a feed forward fashion to control the environment (plant) until the controller guides the plant into a predefined goal set. The concept of "goal set" proposed in this paper is referenced from [32]. In [32], authors proposed a Lyapunov-based design for reinforcement learning. The purpose of [32] is to guide the system to reach and remain in a goal set comprising goal states.

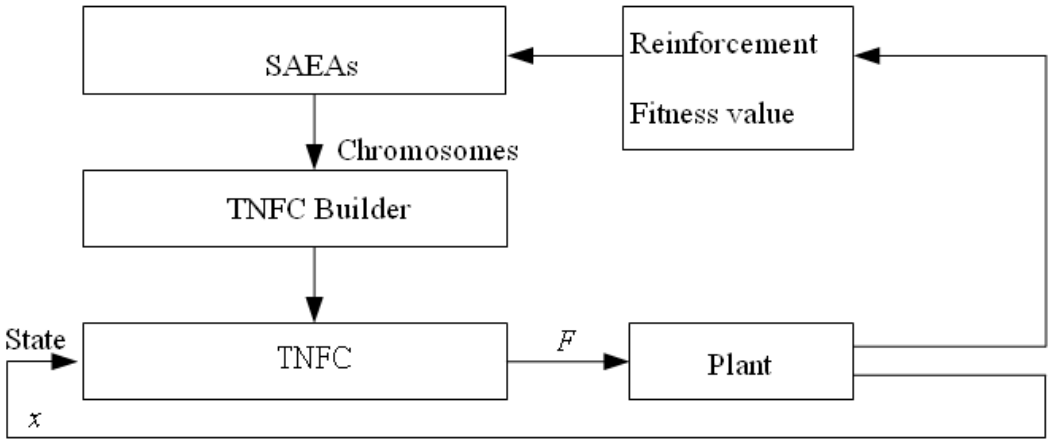


Figure 4.1: Schematic diagram of the ISRL-SAEAs for the TNFC.

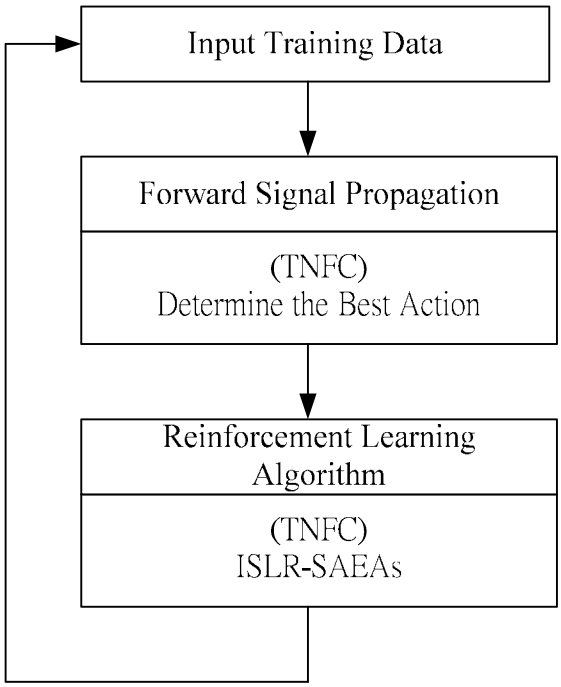


Figure 4.2: Flowchart of the ISRL-SAEAs.

In the simulations of ISRL-SAEAs, when T=G, Properties 4.1, 4.3, 4.4 represent of an achievement of stability. Lyapunov's direct method is mainly used to study the stability of systems of differential equations. Authors in [32] extend this method to the reinforcement learning problem. The proposed ISRL conducts one Lyapunov-style theorem proposed in [32], which provides criterion for designing the reinforcement learning agent. The theorem is listed below. Let $L : S \to \mathbb{R}$ denotes a function positive on $T^c = S - T$, and Δ denotes a fixed real number.

Theorem 4.1 *If* $\forall s \notin T$, *actions* $a \in A(s)$, *all possible next state* s' *(either* $s' \in T$ *or* $L(s) - L(s') \geq \Delta$*), then from any* $s_t \notin T$, *the environment enters* T *within* $\lceil L(s_t)/\Delta \rceil$ *time steps.*

The proof of Theorem 4.1 can be found in [32]. Theorem 4.1 provides a guarantee of a plant's meeting the goal state, if the controller is designed such that it reduces the Lyapunov function of the plant in each time step. Therefore, the main concept of the proposed ISRL is to identify a Lyapunov function of a control plant then design the action choices so that the

reinforcement learning satisfies the above theorem.

4.3 Two Strategies in the ISRL

The main concept of the proposed ISRL is to identify a Lyapunov function of a control plant then design the action choices so that the reinforcement learning satisfies the Theorem 4.1. The learning process of the ISRL is shown in Fig. 4.3. As shown in Fig. 4.3, *Thres _TimeStep* is a predefined parameter that represents the controller that is deemed unsuccessful if it is not able to guide the system into the goal set before *Thres _TimeStep*; *Stable_TimeSteps* is the predefined parameter that indicates the success of a controller if it controls the plant for such a period; successful range represents the boundary of the parameters of the control plant; the strict constraint will be defined later. There are two strategies in the ISRL, judgment and evaluation strategies. The judgment strategy measures the fitness value of controller that fails to guide the system into the goal set. The evaluation strategy measures the fitness value of controller that successfully guide the system into the goal set. The details of the two strategies in the ISRL are shown as follows:

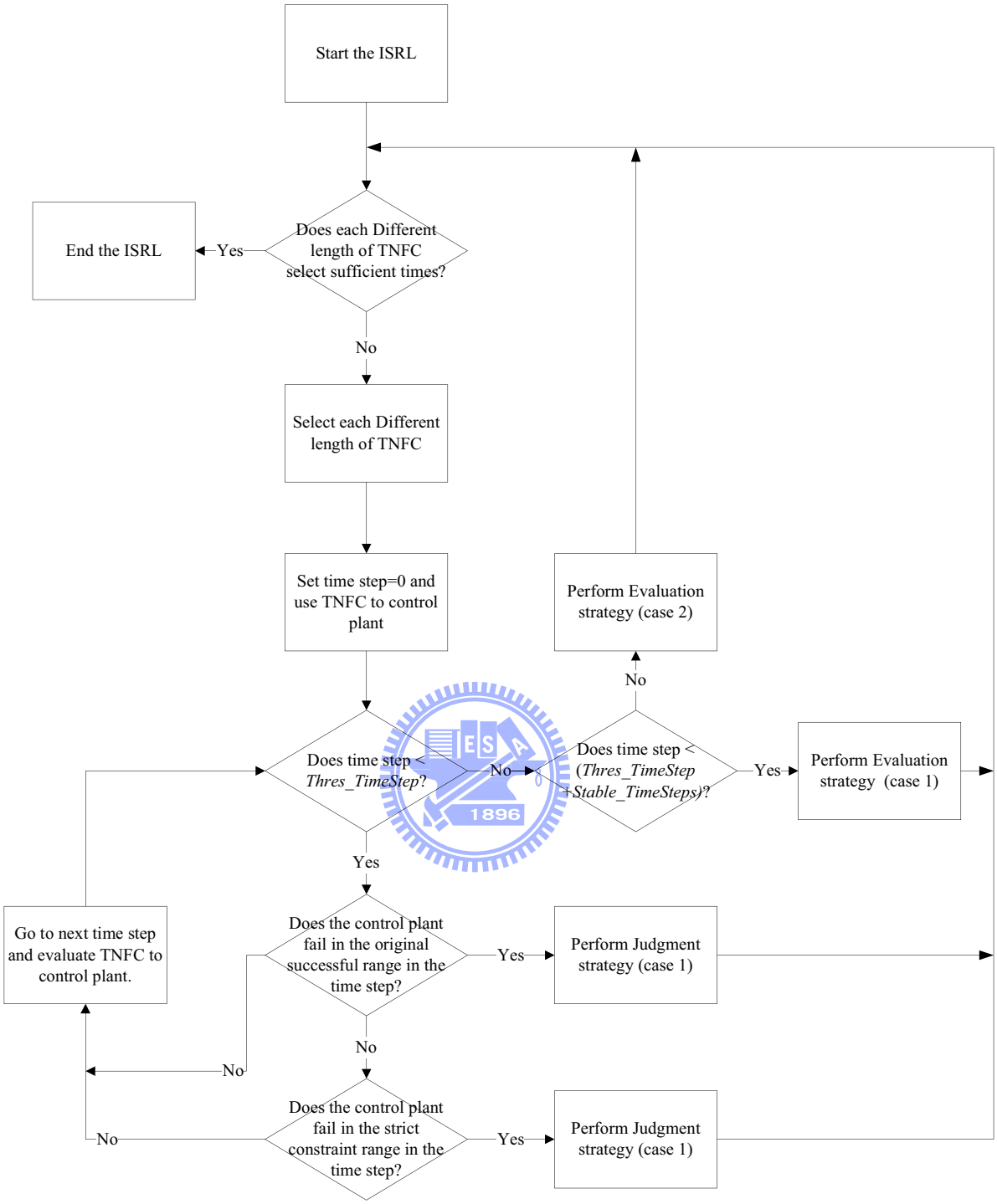


Figure 4.3: Learning process of the ISRL.

Staregy 1. Judgment strategy:

Under the condition that the controller fails to control the plant into the goal set, the fitness value is calculated by the following two cases. Case 1 represents the controller fails under a relative looser constraint. Case 2, on the contrary, represents the failure under the

strict constraint defined in this dissertation.

Case 1. If the system fails at time step t under a loose constraint, before entering goal set, then

$$Fitness\_Value = \frac{1}{Thres\_TimeStep} \times \frac{t\text{-}1}{Thres\_TimeStep}\ ; \tag{4.1}$$

where the $Thres\_TimeStep$ is a predefined parameter. The controller is deemed unsuccessful if it is not able to guide the system into the goal set before $Thres\_TimeStep$.

Case 2. If the plant works under the original successful range, but fails at time step t under a strict constraint before entering the goal set, then

$$Fitness\_Value = \frac{1}{Thres\_TimeStep} \times t. \tag{4.2}$$

The strict constraint is defined by Eq. 4.3. It shrinks the successful range as the time step increases.

$$Strict\_Range = Original\_Range \times \delta, \text{ where}$$
$$\delta = \begin{cases} \left(Thres\_TimeStep + A - t \middle/ Thres\_TimeStep\right), & \text{if } t \le Thres\_TimeStep \\ \left(A \middle/ Thres\_TimeStep\right), & otherwise \end{cases}; \tag{4.3}$$

where A is a parameter that simply prevents the modified range from becoming zero. According to Eq. 4.3, when $t \le Thres\_TimeStep$, this equation provides a better fitness value for the controller that guides the plant into the goal set sooner. When $t > Thres\_TimeStep$, this equation provides a penalty for the controller that exceeds the defined range.

Strategy 2. Evaluation strategy

Under the condition that the controller successfully controls the system into the goal set, the fitness value is calculated by the following two cases. Case 1 represents the system enters the goal set, but falls beyond the range defined in Eq. 4.3. Case 2 represents the controller successfully controls the system.

Case 1. If the system enters the goal set at time step t_1 and falls beyond the range defined in Eq. 4.3 at time step t_2, then

$$Fitness\_Value = \frac{1}{t_1} \times (t_2 - t_1)\ ; \tag{4.4}$$

Case 2. If the system enters the goal set at time step t_1 and stabilizes the system for *Stable_TimeSteps* then

$$Fitness\_Value = Thres\_StableTimeSteps + (\frac{1}{t_1} \times Stable\_TimeSteps); \tag{4.5}$$

where *Thres_StableTimeSteps is* the predefined parameter; *Stable_TimeSteps* is the predefined parameter that indicates the success of a controller if it controls the plant for such a period.

In the ISRL, the control laws of the system are designed according to Lyapunov function. However, for different control systems, we need to define different control laws of the system. Therefore, the Lyapunov-based manipulations on control laws are defined in Chapter 5.

Chapter 5
Control Illustration

To demonstrate the performance of the ISRL-SAEAs for temporal problems, in this chapter, two examples and performance contrasts with some other models are presented. The examples used in this chapter are described as follows.

In Section 5.1, the inverted pendulum control system is adopted to evaluate the performance of the proposed ISRL-SAEAs. Therefore, the three methods of the ISRL-SAEAs are evaluated in this example. This problem is often used as an example of inherently unstable and dynamic systems to demonstrate both modern and classical control techniques ([55]-[57]) or the reinforcement learning schemes ([15]-[21]), and is now used as a control benchmark.

In Section 5.2, the tandem pendulum control system is adopted to evaluate the performance of the three methods of the ISRL-SAEAs. Since the task of an inverted pendulum control system is too easy to find solutions quickly through random search, in this example, a variety of extensions to a basic inverted pendulum control problem have been suggested. The most challenging extension of an inverted pendulum control system is a tandem pendulum control system ([58]-[60]), where two pendulums of different length must be balanced synchronously.

In the experiments, a Pentium 4 chip with a 1.5GHz CPU, a 512MB memory, and the visual C++ 6.0 simulation software are used to implement the control systems.

5.1 Inverted Pendulum Control System

In this section, the classic control problem of an inverted pendulum control system is adopted to evaluate the performance of the ISRL-SAEAs. Figure 5.1 depicts the inverted pendulum control system. This system is often used as an example of inherently unstable and dynamic system to demonstrate both modern and the classic control techniques ([55]-[57]), or the reinforcement learning schemes ([15]-[21]), and is now used as a control benchmark. The bottom of the pendulum is hinged to a cart that travels along a finite-length track to its right or left. Both the cart and the pendulum can move only in the vertical plane; that is, each has only one degree of freedom.

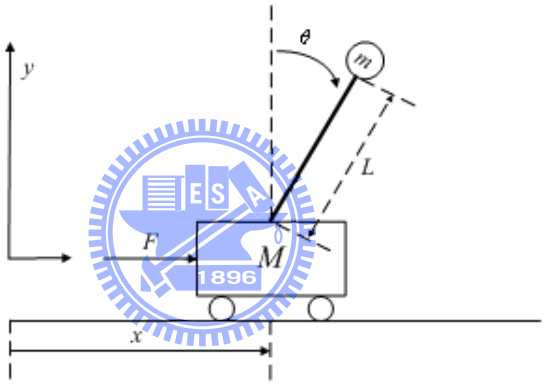


Figure 5.1: The inverted pendulum control system.

The only control action is F, which is the amount of force (in *Newtons*) applied to cart to move it toward left or right. The system fails when the pendulum falls past a certain angle ($\pm 12^{\circ}$ is used here) or the cart runs into the bounds of its track (the distance is 2.4 m from the center to each bound of the track). Using Lagrange's method, the model of the inverted pendulum control system can be obtained as follows:

$$x: \quad (m+M)\ddot{x} + mL(\ddot{\theta}\cos\theta - \dot{\theta}^2 \sin\theta) = F \tag{5.1}$$

$$\theta: \quad \ddot{x}\cos\theta + L\ddot{\theta} - g\sin\theta = 0 \tag{5.2}$$

where

$L = 0.5$ m, the length of the pendulum;

M = 1.0 kg, the mass of the cart;

m = 0.1 kg, the mass of the pendulum;

g = 9.8 m/s , acceleration due to the gravity;

Let $q = (x, \theta)^T$, we can rewrite Eqs. 5.1 and 5.2 into general dynamic form as follows:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau \tag{5.3}$$

where

$$D(q) = \begin{bmatrix} m+M & mL\cos\theta \\ mL\cos\theta & mL^2 \end{bmatrix} \tag{5.4}$$

$$C(q, \dot{q}) = \begin{bmatrix} 0 & -mL\dot{\theta}\sin\theta \\ 0 & 0 \end{bmatrix} \tag{5.5}$$

$$G(q) = \begin{bmatrix} 0 \\ -mgL\sin\theta \end{bmatrix} \tag{5.6}$$

$$\tau = \begin{bmatrix} F & 0 \end{bmatrix}^T \tag{5.7}$$

The total mechanical energy of the system can be derived from

$$E(q, \dot{q}) = \frac{1}{2}\dot{q}^T D(q)\dot{q} + P(q) \tag{5.8}$$

where $P(q)$ denotes the potential energy of the system, $mgL\cos\theta$ in this case, and $G(q) = \frac{\partial P(q)}{\partial q}$. The purpose of this control task is to determine a sequence of forces applying to the cart to balance the pendulum upright, and maintain the cart as stationary as possible. Hence, we define a goal set comprising near-upright and near-stationary states as $G_1 = \left\{(\dot{x}, \theta, \dot{\theta}) : \left\|(\dot{x}, \theta, \dot{\theta})\right\| \le 0.001\right\}$. When the state of inverted pendulum control system is in G_1, according to Eq. 5.8, the total mechanical energy E of the system is mgL equaling 0.49. We define a Lyapunov function $V(q, \dot{q}) = 0.49 - E(q, \dot{q})$. The purpose of this control problem can be transformed to the problem of achieving $V(q, \dot{q}) = 0$. So we define another goal set $G_2 = \{(q, \dot{q}) : V(q, \dot{q}) = 0\}$.

The time derivative of E with respect to time is

$$\begin{aligned}\dot{E}(q,\ \dot{q}) &= \dot{q}^T D(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \dot{q}^T G(q) \\ &= \dot{q}^T\left(-C(q,\ \dot{q})\dot{q} - G(q) + \tau\right) + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \dot{q}^T G(q) \\ &= \dot{q}^T \tau \\ &= \dot{x}F\end{aligned} \tag{5.9}$$

that shows the derivative of E is proportional to the product of the speed of cart and input force. Hence in this paper, following [32], a control law for the learning agent based on Lyapunov analysis is proposed as follows:

$$P(q,\ \dot{q}) = \begin{cases} sgn(\dot{x})F, & \text{if } \mathrm{E}(q,\ \dot{q}) < 0.49 \\ 0, & \text{if } \mathrm{E}(q,\ \dot{q}) \geq 0.49 \end{cases} \tag{5.10}$$

where $sgn(\dot{x}) = \{1 \text{ if } x \geq 0, \text{ and -1 otherwise}\}$ and F is the output of the TNFC that is limited in [-10,10]. $P(q,\ \dot{q})$ guarantees the descent of the Lyapunov function; as a result, this control law satisfies the Theorem 4.1 defined in [32]. Denote the state of the environment at time step $t = (q_t,\ \dot{q}_t) = s_t$. Theorem 4.1 tells that the agent will bring the environment to G_1 within $\lceil L(s_0)/\Delta \rceil$ steps, and remains the environment in the set $\{s:\ L(s_{t+1}) \leq L(s_t)\}$. In the simulation, since the descent step size can not be ensured. Theorem 4.1 can be reduced to the form that the agent will bring the environment to G_1 and remain the environment until it achieves G_2 eventually, if the controlling time step is long enough.

In the simulation of inverted pendulum control system, the original successful region of the variables are $-12^\circ \leq \theta \leq 12^\circ$ and -2.4m$\leq x \leq$2.4m. The strict successful region of θ is described in Eq. 4.3. The constraints on the variables are $-12^\circ \leq \theta \leq 12^\circ$, -2.4m$\leq x \leq$ 2.4m, and -10N$\leq f \leq$ 10N. A control strategy is deemed successful if it can meet the control goal (θ and $\dot{\theta}$ decade to zero). The four input variables $(\theta,\ \dot{\theta},\ x,\ \dot{x})$ and the output $f(t)$ are

normalized between 0 and 1 over the following ranges: θ: [-12, 12], $\dot{\theta}$: [-240, 240], x: [-2.4, 2.4], $\dot{x}$: [-3.26, 3.26], and $f(t)$: [-10, 10]. The ranges of $\dot{\theta}$ and $\dot{x}$ are calculated by experiments with extreme boundary conditions. The car is placed the location of 2.4m (or -2.4m) with pendulum angle set for -12° (or 12°) respectively, then applies the maximum force of -10N (or 10N) to the cart. When the system fails, the observed $\dot{\theta}$ and $\dot{x}$ are the boundaries. The four normalized state variables are used as inputs to the TNFC. The values are floating-point numbers assigned to the SAEAs initially. The fitness function is defined according to Eq. 4.1-4.5.

In this example, the performance evaluation of the SAEAs consists of the HEA, SACG-SE, and SAG-SEFA. In the following sections, the performances of three methods are discussed.

5.1.1 Evaluating performance of the HEA

The initial parameters of the proposed ISRL-HEA in this example are determined by parameter exploration ([103]). The first study in parameter exploration was proposed by De Jong ([103]). As shown in [103], a small population size is good for the initial performance, and a large population size is good for long-term performance. Moreover, a low mutation rate is good for on-line performance, and a high mutation rate is good for off-line performance. In [104], the author found from his simulation that the best population size and mutation rate were 30 and 0.01, respectively. How parameters affect the methods in this study are as follows: 1) the population size affects both the final performance and the efficiency of GA's; 2) the crossover rate deals with the frequency to which the crossover step is applied; 3) the mutation rate deals with the second search step which increases the variability of the population. In this study, the parameters are found using the method given in [104]. Therefore, the number of fuzzy rules has the range from 2 to 20 in increments of 1, the group size has the range from 10 to 100 in increments of 10, the crossover rate has the range from 0.25 to 1 in increments of 0.05, and the mutation rate has the range from 0 to 0.3 in exponential increments. The other

parameters listed in the ISRL-HEA are defined as the same way. The parameters set for the proposed ISRL-HEA are shown in Table 5.1.

Table 5.1 : The initial parameters of the ISRL-HEA before training.

Parameters	Value	Parameters	Value
N_c	100	*Stable _TimeSteps*	5000
Crossover Rate	0.5	*Thres _TimeStep*	1000
Mutation Rate	0.3	*ERSTimes*	50
$[\sigma_{\min}, \sigma_{\max}]$	[0, 2]	A	10
$[m_{\min}, m_{\max}]$	[0, 2]	λ	0.01
$[w_{\min}, w_{\max}]$	[-20, 20]	η	7
$[R_{\min}, R_{\max}]$	[3, 12]	*Generations*	300
Thres _StableTimeSteps	500		

In this example, the coding of a rule in a chromosome is the form in Fig. 3.31 in Section 3.1. A total of thirty runs were performed. Each run started at the different initial state ($\dot{\theta}$ and $\dot{x}$ are set for 0, θ and x are set randomly within a predefined range). The learning curves of ISRL-HEA are shown in Fig. 5.2. In this figure, there are thirty runs each run represents that how soon the TNFC can meet the goal state. The fitness value is defined according to Eqs. 4.1-4.5. The higher fitness value by the end of each run represents that the sooner the plant meets the goal set. When the ISRL-HEA is stopped, the best string from the population in the final generation is selected and applied on the testing phase of the inverted pendulum control system. The results of the probability vectors in MCGA are shown in Fig. 5.3. In this figure, the final average number of rules is 5.

The testing results, which consist of the pendulum angle, pendulum angular velocity (in degrees/seconds), and cart velocity (in meters/seconds) are shown in Fig. 5.4. A total of thirty runs were executed in the testing phase. Each line in Fig. 5.4 represents a single run that starts form a different initial state. The results shown in this figure are the first 1,000 of 6,000 control time steps (*Thres _TimeStep* + *Stable _TimeSteps*). As shown in Fig. 5.4, the ISRL-HEA successfully controlled the inverted pendulum control system in all thirty runs

(the pendulum angle, pendulum angular velocity, and cart velocity decrease to 0).

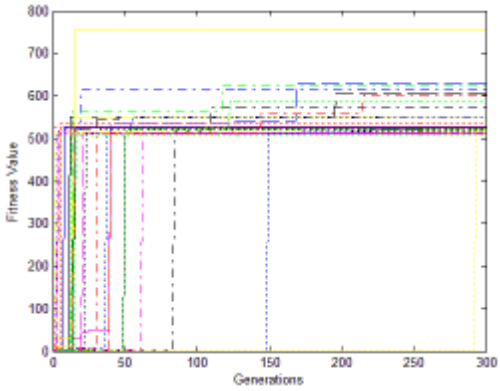


Figure 5.2: The learning curves of the ISRL-HEA.

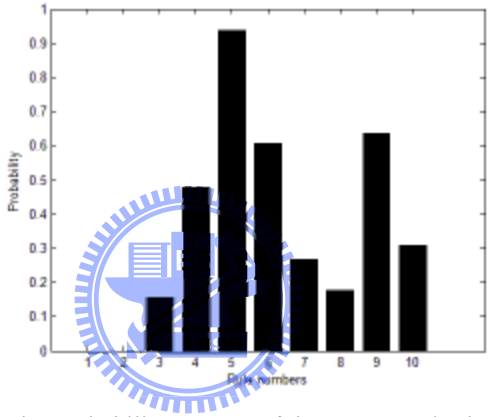
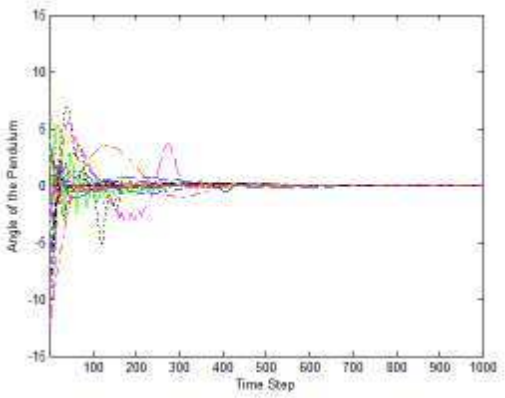
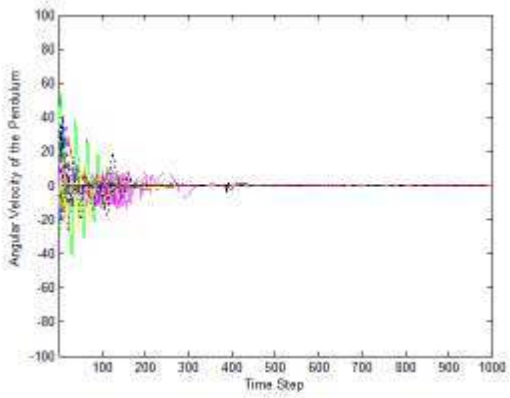


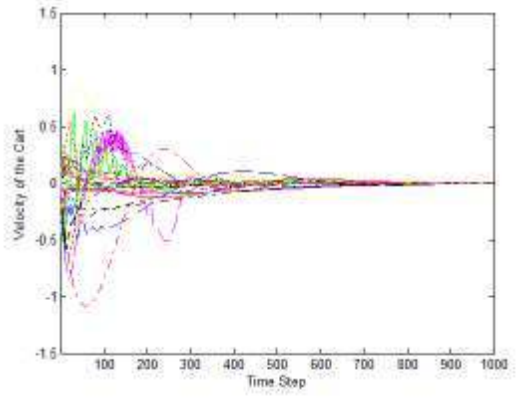
Figure 5.3: The probability vectors of the ERS step in the ISRL-HEA.



(a)



(b)



(c)

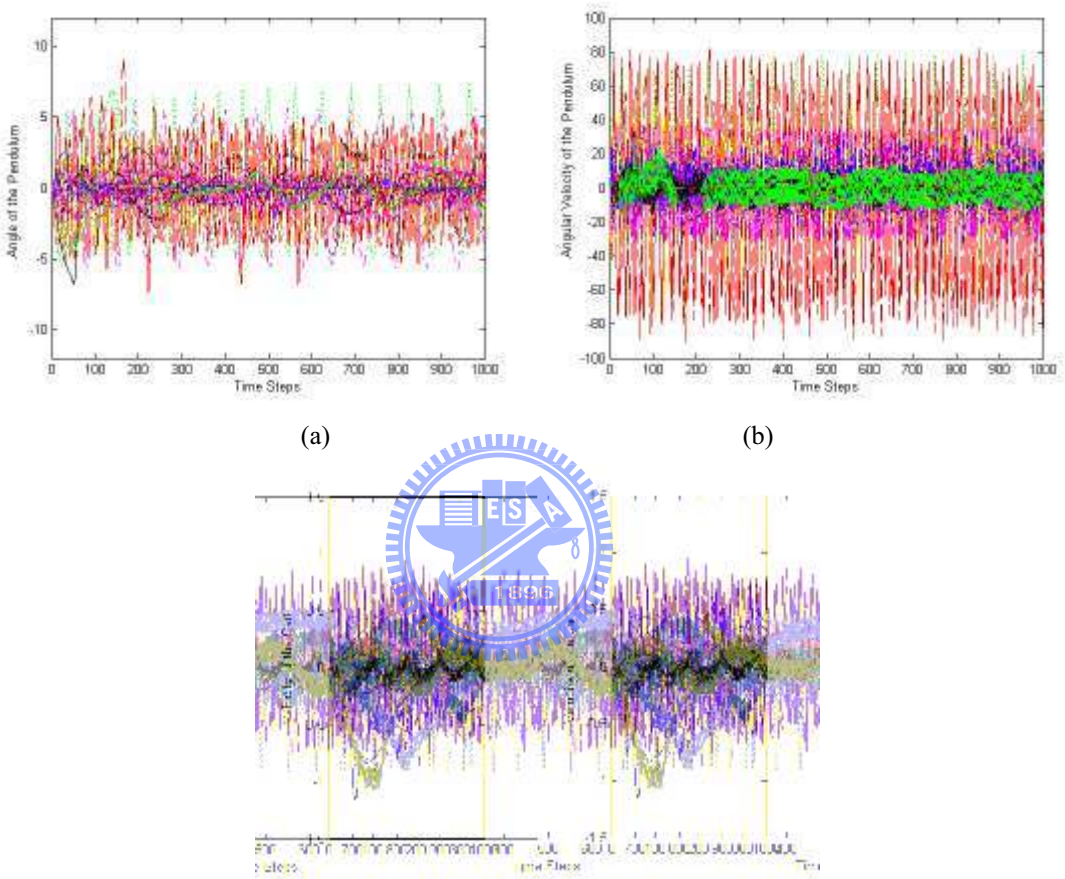
Figure 5.4: Control results of the inverted pendulum control system using the ISRL-HEA in Example 1. (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.

The reinforcement symbiotic evolution (R-SE) ([29]) and the reinforcement genetic algorithm (R-GA) ([26]) were applied to the same control task to compare with the performance of the ISRL-HEA. In the simulation of [29] and [26], parameters of learning are found by using the method given in [104]. Therefore, four rules were set for the R-SE and R-GA, the population size ranges from 10 to 250 in increments of 10, the crossover rate ranges from 0.25 to 1 in increments of 0.05, and the mutation rate ranges from 0 to 0.3 in exponential increments. The resulting parameters set for these methods (R-SE and the R-GA) are shown as follows: 1) the population sizes of the R-SE and the R-GA were 170 and 70, respectively; 2) the crossover rates of the R-SE and the R-GA were 0.55 and 0.6, respectively; 3) the mutation rate of the R-SE and the R-GA were 0.08 and 0.02, respectively. In R-SE ([29]) and R-GA ([26]), the reinforcement signal is designed base on time-step reinforcement architecture ([18]-[20]). The fitness function in R-SE and R-GA is defined according to

$$Fitness\_Value = TIME\_STEP \tag{5.11}$$

where *TIME_STEP* represents how long the experiment is a "success" in one generation. In this example, Eq. 5.11 represents how long before the pendulum falls apart a certain angle ($|\theta| > 12°$) or the cart runs into the bounds of its track ($|x| > 2.4\text{m}$). A control strategy is deemed successful if it can balance a pendulum for 6,000 time steps.

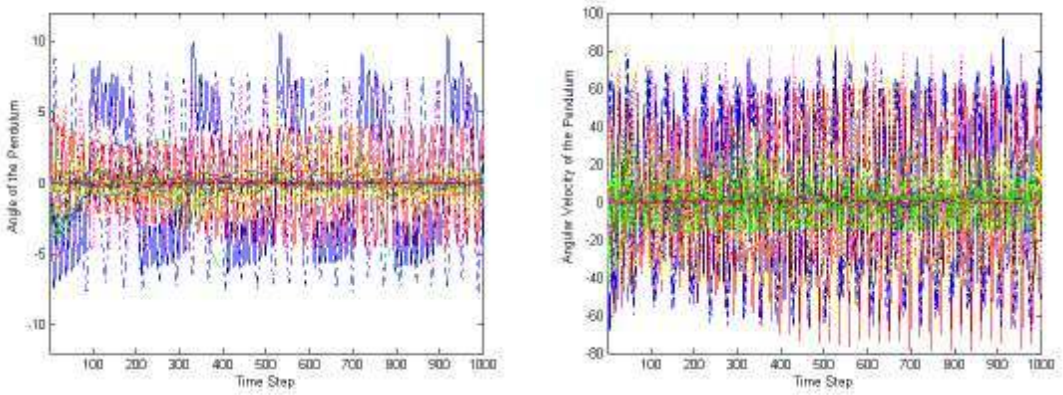
The simulation was carried out for 30 runs. The testing results of the R-SE and R-GA are shown in Figs. 5.5 and 5.6. The results shown in these figures are the first 1,000 of 6,000 control time steps. As shown in Figs. 5.5 and 5.6, not every line meets the control goal ($\dot{x}$, θ and $\dot{\theta}$ decay to zero). It's obvious that the ISRL-HEA obtains better result when compared with [29] and [26], since the $\dot{x}$, θ and $\dot{\theta}$ of the ISRL swing in a smaller range near zero.



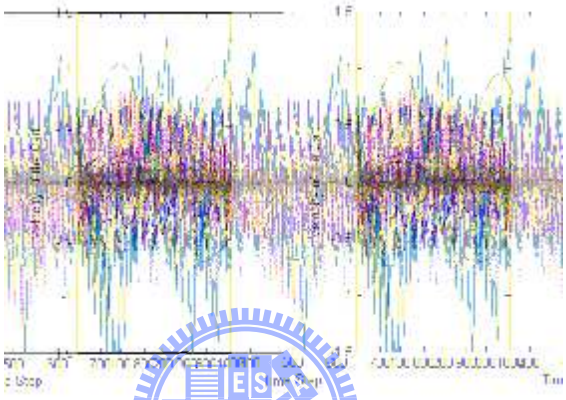
(a) (b)

(c)

Figure 5.5: Control results of the inverted pendulum control system using the R-SE in Example 1. (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.



(a) (b)

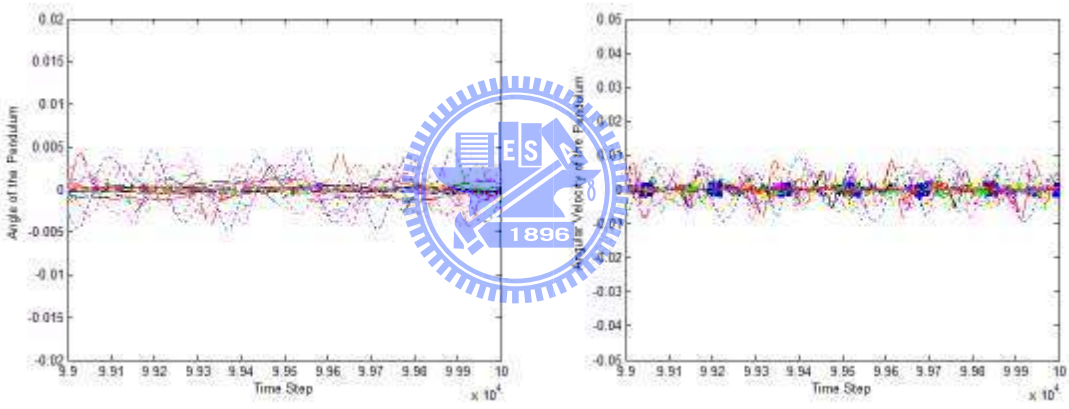


(c)

Figure 5.6: Control results of the inverted pendulum control system using the R-GA in Example 1. (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (d) Velocity of the cart.

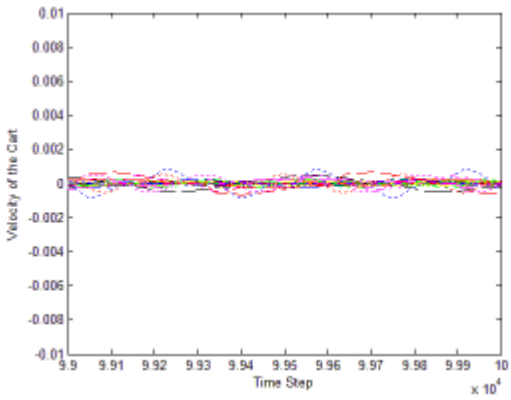
In the further simulation, we select the best-trained individual of the ISRL-HEA, R-GA and R-SE in the training phase, and extend the control time steps to 100,000 in the testing phase. The simulation results, which consist of the pendulum angle, angular velocity of the pendulum, and the cart velocity, are shown in Fig. 5.7. Each line in Fig. 5.7 represents the result of the last 1000 time steps in a run that starts from the different initial state. As shown in Fig. 5.7 (d)-(i), not every line meets the control goal G_1 in the R-SE and R-GA. Moreover, the pendulum angle may swing outside the boundary at the last 1000 time steps. However, in the proposed ISRL-HEA, each line can meet the control goal G_1 and the pendulum is kept upright during the last 1000 time steps. The percentage that the R-SE and the R-GA controls the plant to G_1 are 56% (with 13 runs that the plant unreach G_1 and 4 out of 13 runs that the

pendulum swings outside the boundary) and 54% (with 14 runs that the plant unreach G_1 and 5 out of 14 runs that the pendulum swings outside the boundary) respectively. The reason is that the fitness function used in the R-SE and R-GA only evaluates how long before the pendulum falls apart a certain angle ($|\theta| > 12°$) or the cart runs into the bounds of its track ($|x| > 2.4\text{m}$). Therefore, the system may not reach G_1 and when the control time steps are extend to 100,000 in the testing phase the pendulum may swing outside the boundary. However, in the ISRL-HEA, the percentage that the plant remains in G_1 during the last 1000 time steps is 100%. It's obvious that the ISRL allows the pendulum angle, angular velocity of the pendulum and the cart velocity to swing a small range near zero and stabilize the control system.

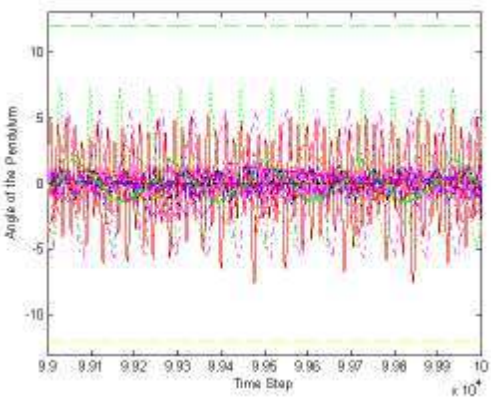


(a)

(b)



(c)



(d)

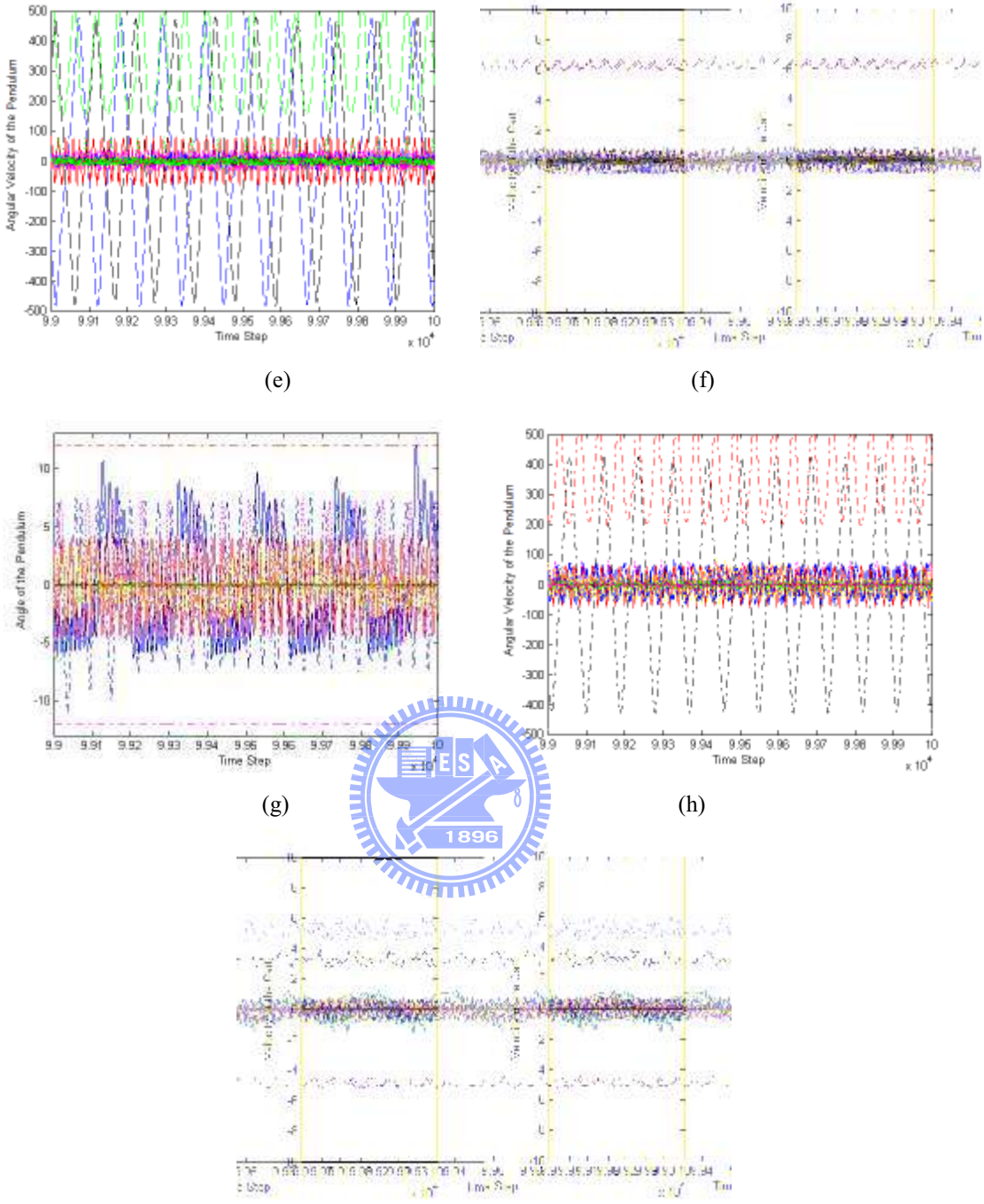


Figure 5.7: Control results of the inverted pendulum control system in Example 1. (a) Angle of the pendulum of ISRL-HEA. (b) Angular velocity of the pendulum of ISRL-HEA. (c) Velocity of the cart of ISRL-HEA. (d) Angle of the pendulum of R-SE. (e) Angular velocity of the pendulum of R-SE. (f) Velocity of the cart of R-SE. (g) Angle of the pendulum of R-GA. (h) Angular velocity of the pendulum of R-GA. (i) Velocity of the cart of R-GA.

The accuracy and CPU time comparison of ISRL-HEA, R-SE, and R-GA are shown in Table 5.2. The ISRL-HEA needs less CPU time than R-SE and R-GA. The reason is that the

ISRL adopts a strict restriction in the earlier time steps and evaluates the control system by how soon the plant can meet the control goal. The individual in ISRL-HEA with better performance means it controls the plant to the goal set sooner. As a result, the CPU time of ISRL-HEA is dramatic less than that of R-SE and R-GA. For example, in the R-SE and R-GA, if one individual fails around 5000 time step, this individual is set with a high fitness value and causes other individuals in the population to approach. At the time when most individuals fail around 5000 time step, the evolution in one generation becomes very time-consuming. As shown in the Table 5.2, when compared with the traditional reinforcement signal design, the proposed ISRL can reduce the CPU time and always control the plant to the goal set. Moreover, the HEA can determine the fuzzy rules automatically without trail and error testing.

The genetic reinforcement learning for neuro control (GENITOR) ([57]), symbiotic adaptive neuro-evolution (SANE) ([96]), temporal difference and genetic algorithm-based reinforcement learning (TDGAR) ([20]), combination of online clustering and Q-value based GA for reinforcement fuzzy system (CQGAF) ([43]), efficient reinforcement learning through dynamical symbiotic evolution (ERDSE) ([44]), and enforce sub-population (ESP) ([40]) have been applied to the same control task and the simulation results are listed in Table 5.2. The accuracy of the controller meet the control goal and keep the pendulum in 100000 time steps and the CPU time are shown in Table 5.2. A total of thirty runs were executed. Each run started at the different initial state. The initial parameters of these methods ([57], [96], [20], [43], [44], and [40]) are determined according to [104]. In these methods, the network size has the range from R_{min} to R_{max} in increments of 1. This dissertation determines the network sizes by executing an evolutionary algorithm with fixed string length for each specification (R_{min} to R_{max} in increments of 1) of the number of network sizes and then computes the average of the generations. In [57], the normal evolutionary algorithm is used to evolve the weights of a fully-connected two-layer neural network, with additional connections from each input unit to

the output layer. After trial-and-error tests, the network size is ten. In [96], the symbiotic evolutionary algorithm is used to evolve a two-layer neural network. In [96], the network size is ten. The TDGAR ([20]) that consists of the critic network and action network to the learning system. The critic network is a standard three-layer feedforward network using sigmoid functions in the hidden layer and output layer. The action network is a fuzzy neural network with five layers of nodes and each layer performs one stage of the fuzzy inference process. There are five hidden nodes and five fuzzy rules in the critic network and the action network. In CQGAF ([43]), the fuzzy controller with Q-value based genetic algorithm is proposed to solve controller problems. After trial-and-error tests, the final average number of rules in CQGAF of thirty runs is 8 by using the on-line clustering algorithm. In ERDSE ([44]), the TSK type neuro-fuzzy controller is adopted to solve controller problems. After trial-and-error tests, the number of rules in ERDSE is 7. In ESP ([40]), the author proposed enforced sub-populations to evaluate solution locally. There are five sub-populations in ESP. The other parameters set for six methods ([57], [96], [20], [43], [44], and [40]) are as follows: 1) the population sizes of the six methods are 130, 170, 100, 130, 80 and 40, respectively; 2) the crossover rates of the six methods are 0.45, 0.55, 0.35, 0.45, 0.8 and 0.5, respectively; 3) the mutation rate of the six methods are 0.21, 0.17, 0.16, 0.24, 0.1 and 0.18, respectively. When each training step is stopped, the best combination of strings from the population in the final generation is selected and tested with different initial states in thirty times.

As shown in Table 5.2, the proposed ISRL-HEA is more feasible and effective when compared with other existing models ([26], [29], [57], [96], [20], [43], [44], and [40]). The advantages of the ISRL-HEA can be listed as follows:

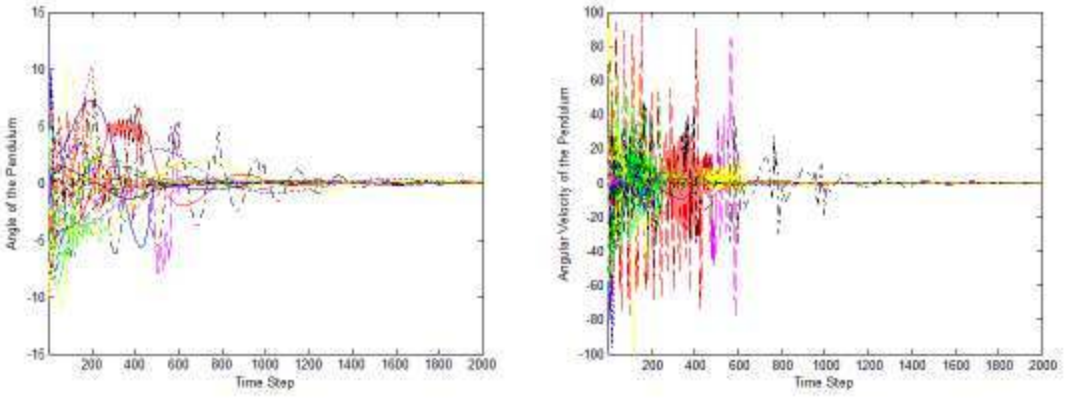
1. Using the concept of statistics, the ISRL-HEA computes the suitable number of fuzzy rules by probability to avoid the flaw that the number of fuzzy rules has to be assigned in advance under different environments.
2. The ISRL enhances the stability of the control system by using the design of

Lyapunov-based safe reinforcement learning. It has better capability to stabilize the plant under different initial states.

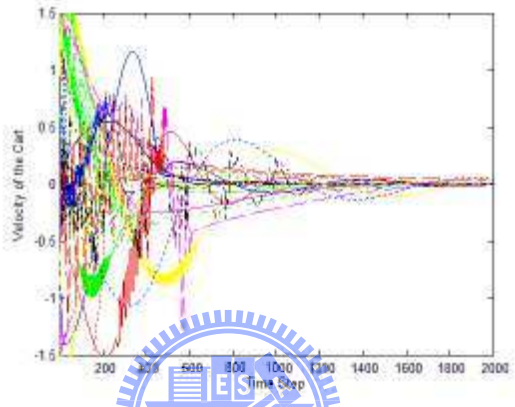
Table 5. 2: Performance comparison of various existing models.

Method	CPU time				Accuracy
	Mean	Best	Worst	Std.	
GENITOR ([57])	120.95	61.34	320.36	92.95	50%
SANE ([96])	97.56	48.54	254.84	83.56	61%
R-GA ([26])	89.83	34.85	192.93	69.94	54%
R-SE ([29])	73.14	28.66	149.43	57.87	56%
TDGAR ([20])	69.13	26.54	112.73	41.58	53%
ESP ([40])	58.32	22.08	95.57	35.27	56%
ERDSE ([44])	51.19	20.77	88.53	30.74	67%
CQGAF ([43])	48.82	18.79	84.39	26.31	59%
ISRL-HEA	**39.97**	**15.10**	**71.01**	**18.23**	**100%**

To demonstrate the proposed ISRL, in this example the safe reinforcement learning (SRL) ([32]) is used. Therefore, the SRL-HEA is used to compare performance with the proposed ISRL-HEA. The simulation was carried out for 30 runs. The goal sets are defined same as the ISRL-HEA. The testing results of the SRL-HEA are shown in Fig. 5.8. The results shown in Fig. 5.8 are the first 1,000 of 100,000 control time steps. As shown in Fig. 5.8, although each line can meet the control goal. The time steps the SRL-HEA needs to meet the control goal are longer than the ISRL-HEA (as shown in Fig. 5.4). The accuracy, CPU time, and time steps that the systems need to meet the control goal of the ISRL-HEA and SRL-HEA are shown in Table 5.3. As shown in this table, the proposed ISRL-HEA is more feasible and effective when compared with SRL-HEA. The reason is that the ISRL-HEA can evaluate how soon the system meets the control goal.



(a) (b)



(c)

Figure 5. 8: Control results of the inverted pendulum control system using the SRL-HEA in Example 1. (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (d) Velocity of the cart.

Table 5. 3: Performance comparison of ISRL-HEA and SRL-HEA.

Method	Time steps				CPU Time				Accuracy
	Best	Mean	Worst	Std.	Best	Mean	Worst	Std.	
ISRL-HEA	**85**	**287**	**421**	**92**	**15.10**	**39.97**	**71.01**	**18.23**	**100%**
SRL-HEA	257	754	1483	338	21.37	56.64	92.71	33.52	100%

Moreover, to demonstrate the efficiency of the proposed ERS, in this example the three different methods are used such as: the proposed ISRL-HEA (Type I), the proposed ISRL-HEA without ERS (Type II), and the fixed length genetic algorithm (Type III). In Type I method, the proposed ISRL-HEA combines the MVGA and the ERS. In Type II method, the proposed ISRL-HEA without using the probability vectors to determine the number of fuzzy

rules. That is, only MVGA are used. In Type III method, the ISRL are used with a genetic algorithm that has fixed length. In Type III method, we determine the number of fuzzy rules by executing a genetic algorithm with fixed string length for each specification (R_{min} to R_{max} in increments of 1) of the number of fuzzy rules and then compute the average of the generations. All the three methods are designed base on ISRL. Table 5.4 shows the performance comparison of three methods.

As shown in Table 5.4, compare to Type I and Type II method, the Type I method needs few CPU time to balance the control system. The reason is that the ERS can determine the suitable number of fuzzy rules automatically. Although the Type II method can also find the number of fuzzy rules with MVGA, however, the number of individuals with same length is fixed. Therefore, the evolutionary algorithm needs more CPU time to search the solution. In the proposed ERS, the number of individuals with same length is determined according to their performance.

Compare to Type I and Type III method, the Type I needs few CPU time to balance the control system. The reason is that the Type I method uses ERS to determine the number of fuzzy rules automatically. However, in Type III method, the number of fuzzy rules is determined by trial-and-error testing. Therefore, the average of the generations of the Type I method is larger than the Type I method.

Table 5. 4: Performance comparison of three different methods.

Method	Mean	Best	Worst	Std.
Type I method	**39.97**	**15.10**	**71.01**	**18.23**
Type II method	57.31	25.45	98.82	30.93
Type III method	85.83	30.85	183.93	60.94

5.1.2 Evaluating performance of the SACG-SE

In this section, the inverted pendulum control system is used to evaluate the performance of the SACG-SE. The initial parameters of the proposed ISRL-SACG-SE in this example are

determined by parameter exploration ([104]). The parameters set for the ISRL-SACG-SE are shown in Table 5.5.

Table 5. 5: The initial parameters of the ISRL-SACG-SE before training.

Parameters	Value	Parameters	Value
N_c	20	$Stable\_TimeSteps$	5000
Crossover Rate	0.4	$Thres\_TimeStep$	1000
Mutation Rate	0.15	$TSSATimes$	50
$[\sigma_{\min}, \sigma_{\max}]$	[0, 2]	A	10
$[m_{\min}, m_{\max}]$	[0, 2]	λ	0.01
$[w_{\min}, w_{\max}]$	[-20, 20]	η	7
$[R_{\min}, R_{\max}]$	[3, 12]	*Generations*	300
P_{size}	18	$Thres\_StableTimeSteps$	500

A total of thirty runs were performed. Each run started at the different initial state ($\dot{\theta}$ and $\dot{x}$ are set for 0, θ and x are set randomly according to the predefined ranges). Figure 5.9 shows one run of the results of the probability vectors in the TSSA. In this figure, the final optimal number of rules is 4. Table 5.6 shows the mean, best, and worst of the optimal number of rules from thirty runs. The learning curve of the ISRL-SACG-SE after thirty runs is shown in Fig. 5.10. In this figure, there are thirty runs each run represents how soon the TNFC can meet the goal state. When ISRL-SACG-SE is stopped, the best combination of strings from the groups in the final generation is selected and tested on the inverted pendulum control system.

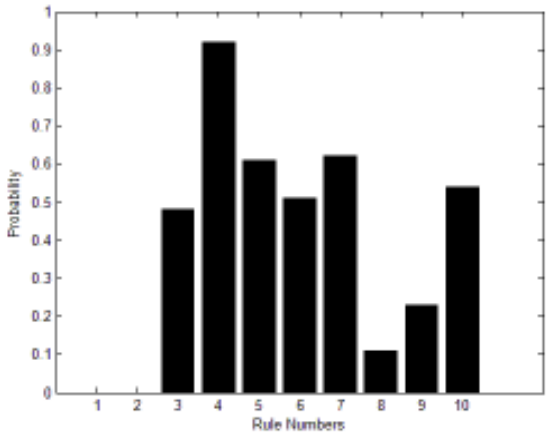


Figure 5. 9: The results of the probability vectors in the TSSA.

Table 5. 6: The number of rules from thirty runs of the TSSA.

Method	Mean	Best	Worst
ISRL-SACG-SE	**4**	**3**	**10**

The simulation was carried out for thirty runs. The successful results, which consist of the pendulum angle, angular velocity of the pendulum (in degrees/seconds), and the velocity of the cart (in meters/seconds) are shown in Fig. 5.11. Each line in Fig. 5.11 represents each run with a different initial state. The results shown in this figure are the first 1,000 of 6,000 control time steps ($Thres\_TimeStep + Stable\_TimeSteps$). As shown in Fig. 5.11, the ISRL-SACG-SE successfully controlled the inverted pendulum control system in all thirty runs (the pendulum angle, pendulum angular velocity, and cart velocity decrease to 0).

As well as ISRL-HEA, we select the best-trained individual of the proposed ISRL-SACG-SE in the training phase, and extend the control time steps to 100,000 in the testing phase. The simulation results, which consist of the pendulum angle, the pendulum angular velocity, and the cart velocity, are shown in Fig. 5.12. Each line in Fig. 5.12 represents the result of the last 1000 time steps in a run that starts from the different initial state. As shown in Fig. 5.12, each line can meet the control goal G_1 and the pendulum is kept upright during the last 1000 time steps. In the ISRL-SACG-SE, the percentage that the plant

remains in G_1 during the last 1000 time steps is 100%. It's obvious that the ISRL allows the pendulum angle, the pendulum angular velocity and the cart velocity to swing a small range near zero and stabilize the control system.

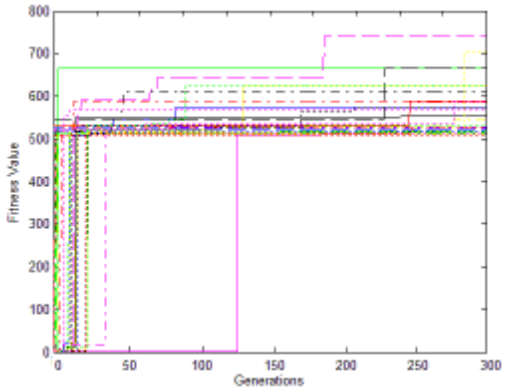
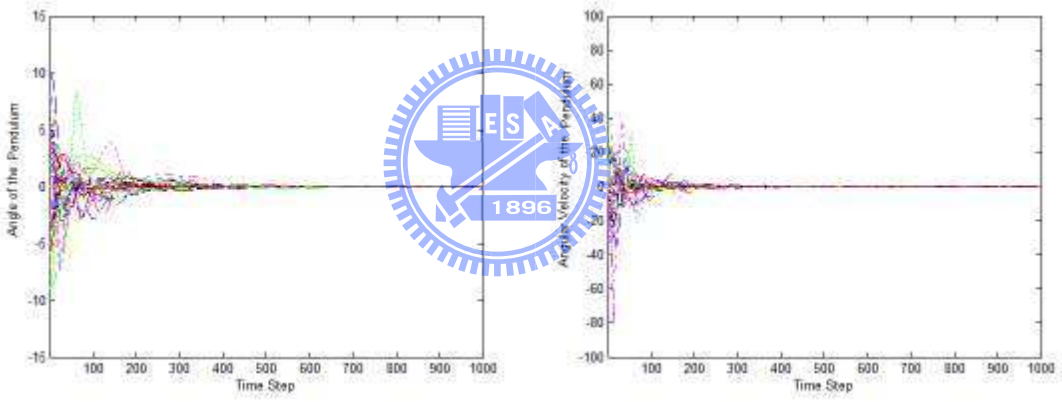
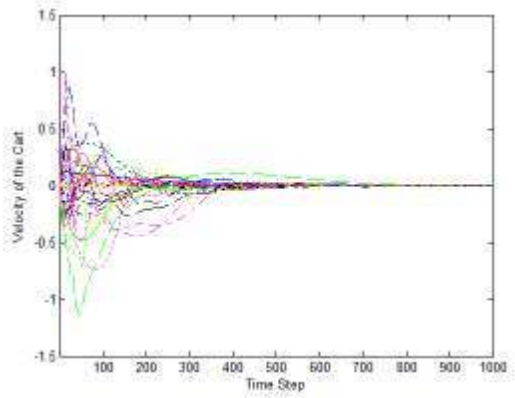


Figure 5. 10: The learning curve of the SACG-SE.



(a) (b)



(c)

Figure 5. 11: Control results of the inverted pendulum control system using the ISRL-SACG-SE in Example 1 (first 1000 time). (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.

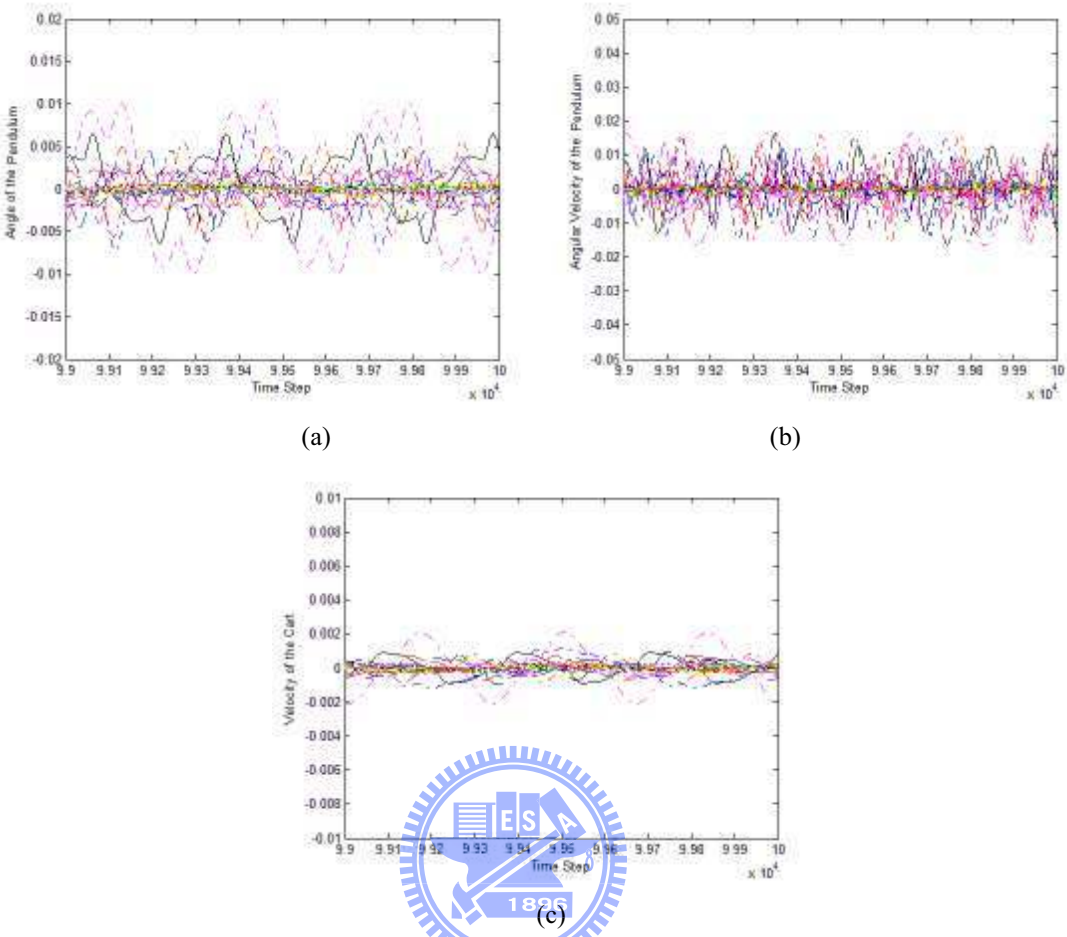


Figure 5. 12: Control results of the inverted pendulum control system using the ISRL-SACG-SE in Example 1 (last 1000 time). (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.

In this example, in order to demonstrate the effectiveness and efficiency of the proposed ISRL-SACG-SE, the R-SE ([29]) and R-GA ([26]) are used to compare with ISRL-SACG-SE. As shown in Fig. 5.7 (d)-(i), the accuracy of the TNFC with the R-SE and R-GA that the pendulum does not swing outside the boundary after 6,000 time steps are 56% and 54%. However, in the ISRL-SACG-SE, the accuracy of the TNFC success meet the control goal and keep the pendulum in 100,000 time steps is 100%. As shown in Fig. 5.12 and 5.7, the ISRL-SACG-SE can perform better than the R-SE and R-GA.

The accuracy and CPU time comparison of the ISRL-SACG-SE, R-SE, and R-GA are shown in Table 5.7. As shown in the Table 5.7, when compared with the traditional

reinforcement signal design, the proposed ISRL reduces the CPU time and always controls the plant to the goal set. Moreover, the SACG-SE can not only determine the fuzzy rules automatically without trail and error testing but also let the chromosomes that perform well to cooperate for generating better solutions.

Compare to HEA, the SACG-SE can obtain smaller CPU times because of the SACG-SE considers both of cooperation and specialization. As shown in Fig. 5.2 and 5.10, the learning curves of the SACG-SE converge more quickly than those of the HEA. The worst, mean, best and standard deviation of CPU time of the HEA and SACG-SE are shown in Table 5.7. As shown in this table, the SACG-SE obtains better performance than the HEA.

The GENITOR ([57]), SANE ([96]), TDGAR ([20]), CQGAF ([43]), ERDSE ([44]), and ESP ([40]) methods have been applied to the same control problem. The accuracy and CPU time are shown in Table 5.7. A total of thirty runs were performed. Each run started at the different initial state. The initial parameters of these methods ([57], [69], [20], [43], [44], and [40]) are determined according to Section 5.1.1. The control time steps for testing are extended to 100,000 time steps. As shown in Table 5.7, the proposed ISRL-SACG-SE is more feasible and effective when compared with other existing models ([26], [29], [57], [96], [20], [43], [44], and [40]). The advantages of the ISRL-SACG-SE can be listed as follows:

1. Using the TSSA, the ISRL-SACG-SE computes by probability the suitable number of fuzzy rules to avoid the flaw that the number of fuzzy rules has to be assigned in advance under different environments.
2. The ISRL enhances the stability of the control system by using the design of Lyapunov-based safe reinforcement learning. It has better capability to stabilize the plant under different initial states.
3. The ECCS lets the well-perform chromosomes to cooperate for generating better solutions in the generations.
4. The Group-based symbiotic evolution can evaluate the solution locally.

Table 5. 7: Performance comparison of various existing models in Example 1.

Method	CPU time				Accuracy
	Mean	Best	Worst	Std.	
GENITOR ([57])	120.95	61.34	320.36	92.95	50%
SANE ([96])	97.56	48.54	254.84	83.56	61%
R-GA ([26])	89.83	34.85	192.93	69.94	54%
R-SE ([29])	73.14	28.66	149.43	57.87	56%
TDGAR ([20])	69.13	26.54	112.73	41.58	53%
ESP ([40])	58.32	22.08	95.57	35.27	56%
ERDSE ([44])	51.19	20.77	88.53	30.74	67%
CQGAF ([43])	48.82	18.79	84.39	26.31	59%
ISRL-HEA	**39.97**	**15.10**	**71.01**	**18.23**	**100%**
ISRL-SACG-SE	**30.54**	**10.23**	**49.21**	**11.12**	**100%**

For demonstrating the efficiency of the each component of the proposed SACG-SE (the group-based symbiotic evolution (GSE), TSSA, and ECCS), in this example, five different methods: the proposed ISRL-SACG-SE without TSSA (Type I), the ISRL-SACG-SE without ECCS (Type II), the ISRL-SACG-SE without TSSA and ECCS(Type III), and the SE method (Type IV), and the proposed ISRL-SACG-SE (Type V) are used. In the Type I, III, IV methods, the number of fuzzy rules is determined according to trail and error testing and then compute the average of the generations. In Type II method, each group performs the two-point crossover strategy independently. In Type III, only GSE is used. In the Type IV method, the SE ([29]) with ISRL is adopted. In the Type V method, the ISRL-SACG-SE uses the TSSA to determine fuzzy rules automatically and the proposed ECCS is adopted to perform crossover strategy. In the Type I, III, IV methods, the parameters are set according to [104]. In Type I, III, IV methods, we determine the number of fuzzy rules by executing Type I, III, IV methods with fixed string length for each specification of the number of fuzzy rules and then compute the average of the generations. All the five methods are designed base on ISRL. The performance of five methods is shown in Table 5.8.

In Table 5.8, comparing Type IV with Type III, the GSE outperform than SE because of

the chromosomes that use to evaluate the solution locally can obtain better performance compared to systems of only one population be used to evaluate the solution. Comparing Type I with Type V method, the Type V method needs few CPU time to balance the control system. The reason is that the TSSA can determine the suitable number of fuzzy rules automatically. However, in Type I method, the number of fuzzy rules is determined by trial-and-error testing. Therefore, the average of the generations of the Type I method is larger than Type V method. Comparing Type II with Type V method, it is observed that the SACG-SE (Type V) performs better than Type II method. It is observed that ECCS can reduce CPU time. As shown in Table 5.8, the proposed ISRL-SACG-SE (Type V) performs better than other four types of methods.

Table 5. 8: Performance comparison of different methods.

Method	CPU Time			
	Mean	Best	Worst	Std.
Type I	51.54	18.23	83.21	31.12
Type II	45.93	16.41	76.87	27.39
Type III	58.43	22.18	98.91	38.55
Type IV	68.26	26.63	131.25	51.43
Type V	**30.54**	**10.23**	**49.21**	**11.12**

5.1.3 Evaluating performance of the SAG-SEFA

The initial parameters of the proposed ISRL-SAG-SEFA in this example are determined by parameter exploration ([104]). The parameters set for the ISRL-SAG-SEFA are shown in Table 5.9. As shown in Table 5.9, the *Minimum_Suppor* used in FP-growth is set as half number of transactions according to [52]. *Minimum_Suppor* effects the number of frequent item sets (the number of suitable group sets). If *Minimum_Suppor* is too small, large number of frequent item sets will be generated, which cause the system unable to estimate superior group sets. On the contrary, when the *Minimum_Suppor* is too large, few frequent item sets will be generated, which causes the system unable to pick chromosomes in a sufficient amount of suitable group sets. After experimenting in this paper, and referring to [52], we can

see that when *Minimum_Suppor* is set as half number of transactions, the performance is satisfactory.

Table 5. 9: The initial parameters of the ISRL-SAG-SEFA before training.

Parameters	Value	Parameters	Value
P_{size}	16	*Stable _TimeSteps*	5000
N_C	10	*Thres _TimeStep*	1000
Selection _Times	200	*TSSAimes*	30
NormalTimes	10	A	10
SearchingTimes	15	λ	0.01
ExploringTimes	20	η	7
Crossover Rate	0.5	*Generations*	300
Mutation Rate	0.2	*Minimum_Suppor*	*TransactionNum*/2
ThreadFitnessvalue	550	*Thres _StableTimeSteps*	500

The coding of a rule in a chromosome is the form given in Fig. 3.10. The values are floating-point numbers initially assigned using the ISRL-SAG-SEFA. A total of thirty runs were performed in this simulation. Each run started at the different initial state ($\dot{\theta}$ and $\dot{x}$ are set for 0, θ and x are set randomly according to the predefined ranges). The mean, best, and worst of the optimal number of rules by performing the TSSA from thirty runs is shown in Table 5.10.

Table 5. 10: The number of rules from thirty runs of the TSSA.

Method	Mean	Best	Worst
ISRL-SAG-SEFA	**5**	**3**	**10**

The learning curve of the ISRL-SAG-SEFA after thirty runs is shown in Fig. 5.13. In this figure, there are thirty runs each run represents that how soon the TNFC can meet the goal state. When ISRL-SAG-SEFA is stopped, the best combination of strings from the groups in the final generation is selected and tested on the inverted pendulum control system. The successful results, which consist of the pendulum angle, angular velocity of the pendulum (in degrees/seconds), and the velocity of the cart (in meters/seconds) are shown in Fig. 5.14. Each

line in Fig. 5.14 represents each run with a different initial state. The results shown in this figure are the first 1,000 of 6,000 control time steps ($Thres\_TimeStep + Stable\_TimeSteps$). As shown in Fig. 5.14, the ISRL-SAG-SEFA successfully controlled the inverted pendulum control system in all thirty runs (the pendulum angle, pendulum angular velocity, and cart velocity decrease to 0).

As well as the ISRL-HEA and ISRL-SACG-SE, we select the best-trained individual of the proposed ISRL-SAG-SEFA in the training phase, and extend the control time steps to 100,000 in the testing phase. The simulation results, which consist of the pendulum angle, the pendulum angular velocity, and the cart velocity, are shown in Fig. 5.15. Each line in Fig. 5.15 represents the result of the last 1000 time steps in a run that starts from the different initial state. As shown in Fig. 5.15, the proposed ISRL-SAG-SEFA, each line can meet the control goal G_1 and the pendulum is kept upright during the last 1000 time steps. Moreover, in the ISRL-SAG-SEFA, the percentage that the plant remains in G_1 during the last 1000 time steps is 100%. It's obvious that the ISRL allows the pendulum angle, the pendulum angular velocity, and the cart velocity to swing a small range near zero and stabilize the control system.

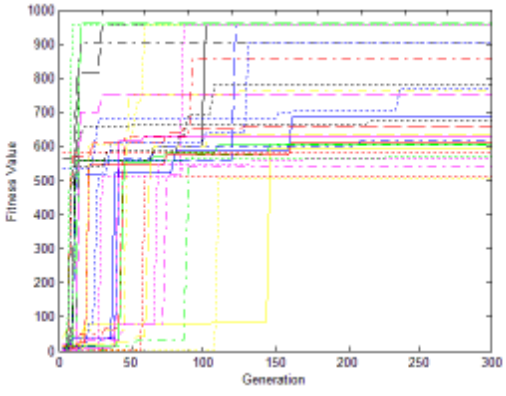
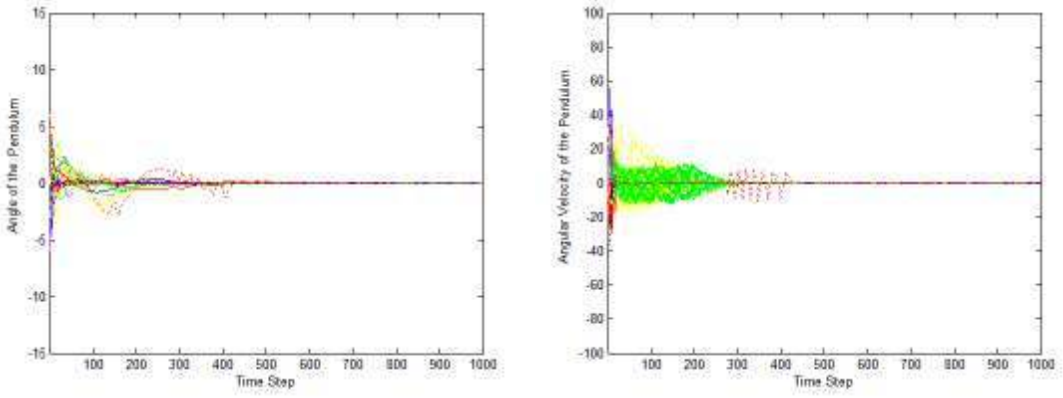
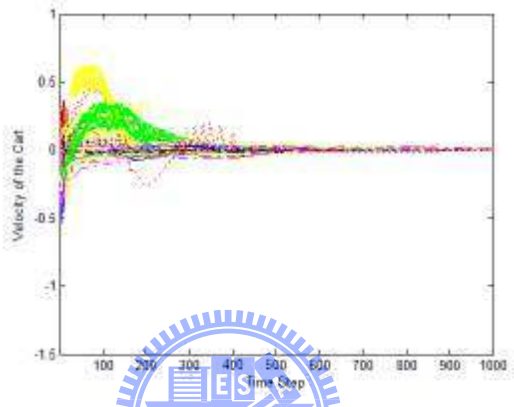


Figure 5. 13: The learning curve of the ISRL-SAG-SEFA.

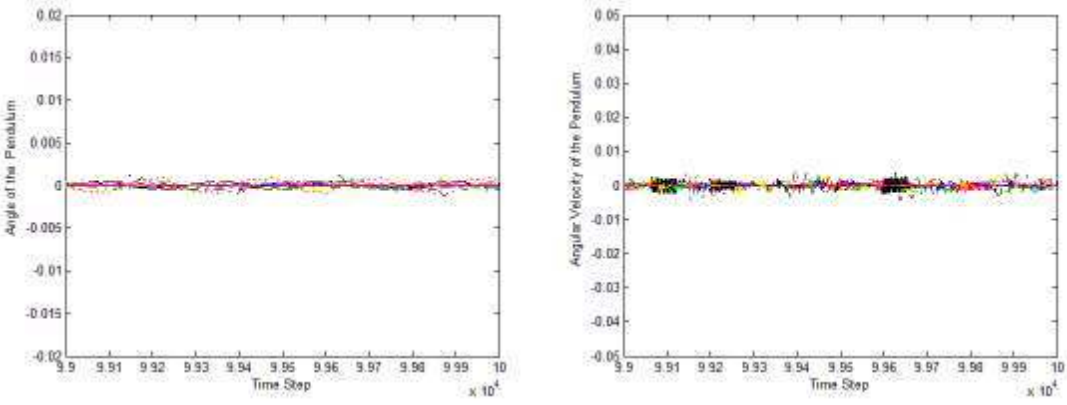


(a) (b)

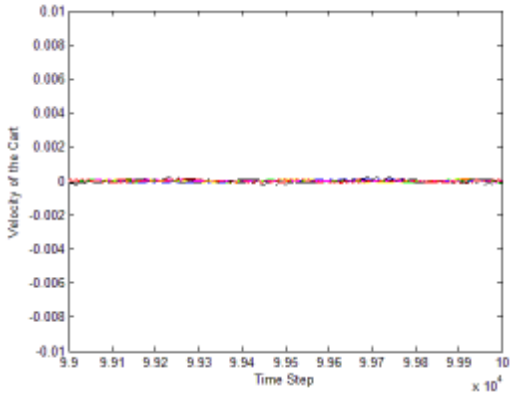


(c)

Figure 5. 14: Control results of the inverted pendulum control system using the ISRL-SAG-SEFA in Example 1 (first 1000 time). (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.



(a) (b)



(c)

Figure 5. 15: Control results of the inverted pendulum control system using the ISRL-SAG-SEFA in Example 1 (last 1000 time). (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.

In this example, in order to demonstrate the effectiveness and efficiency of the proposed ISRL-SAG-SEFA, the R-SE ([29]) and R-GA ([26]) are also used to compare with ISRL-SAG-SEFA. As shown in Fig. 5.7 (d)-(i), the accuracy of the TNFC with the R-SE and R-GA that the pendulum does not swing outside the boundary after 6,000 time steps are 56% and 54%. However, in the ISRL-SAG-SEFA, the accuracy of the TNFC success meet the control goal and keep the pendulum in 100,000 time steps is 100%. As shown in Fig. 5.7 and 5.15, the ISRL-SAG-SEFA can perform better than the R-SE and R-GA.

The accuracy and CPU time comparison of ISRL-SAG-SEFA, R-SE, and R-GA are shown in Table 5.11. The ISRL-SAG-SEFA needs less CPU time than R-SE and R-GA. The reason is that the ISRL adopt a strict restriction in earlier time steps and evaluate the control system by how soon the system can meet the control goal. Therefore, the individuals in ISRL-SAG-SEFA have the high performance and the system can reach and remain the control goal in the earlier time steps. About this, the CPU time of ISRL-SACG-SE is dramatic less than that of R-SE and R-GA.

Compare to SACG-SE, the SAG-SEFA can obtain smaller CPU times because of the SAG-SEFA not only considers both of cooperation and specialization but also selects suitable

groups to perform selection and crossover by using data-mining method. As shown in Fig. 5.10 and 5.13, the learning curves of the SAG-SEFA converge more quickly than those of the SACG-SE. The worst, mean, best, and standard deviation of CPU time of the SACG-SE and SAG-SEFA are shown in Table 5.11. As shown in this table, the SAG-SEFA obtains small CPU time than the SACG-SE.

The [57], [96], [20], [43], [44], and [40] have been applied to the same control problem. Their simulation results are listed in Table 5.11. Table 5.11 shows the accuracy and CPU time for the control model. The initial parameters of these methods ([57], [96], [20], [43], [44], and [40]) are determined according to Section 5.1.1. The control time steps for testing are extended to 100,000 time steps. As shown in Table 5.11, the proposed ISRL-SAG-SEFA is more feasible and effective when compared with other existing models ([29], [26], [57], [96], [20], [43], [44], and [40]). The advantages of the ISRL-SAG-SEFA can be listed as follows:

1. Using the TSSA, the ISRL-SAG-SEFA computes by probability the suitable number of fuzzy rules to avoid the flaw that the number of fuzzy rules has to be assigned in advance under different environments.
2. The ISRL enhances the stability of the control system by using the design of Lyapunov-based safe reinforcement learning. It has better capability to stabilize the plant under different initial states.
3. The Group-based symbiotic evolution can evaluate the solution locally.
4. The SAG-SEFA not only considers both of cooperation and specialization but also selects suitable groups to perform selection and crossover by using DMSS and DMCS.

Table 5. 11: Performance comparison of various existing models in Example 1.

Method	CPU time				Accuracy
	Mean	Best	Worst	Std.	
GENITOR ([57])	120.95	61.34	320.36	92.95	50%
SANE ([96])	97.56	48.54	254.84	83.56	61%
R-GA ([26])	89.83	34.85	192.93	69.94	54%
R-SE ([29])	73.14	28.66	149.43	57.87	56%
TDGAR ([20])	69.13	26.54	112.73	41.58	53%
ESP ([40])	58.32	22.08	95.57	35.27	56%
ERDSE ([44])	51.19	20.77	88.53	30.74	67%
CQGAF ([43])	48.82	18.79	84.39	26.31	59%
ISRL-HEA	**39.97**	**15.10**	**71.01**	**18.23**	**100%**
ISRL-SACG-SE	**30.54**	**10.23**	**49.21**	**11.12**	**100%**
ISRL- SAG-SEFA	**18.12**	**5.31**	**28.51**	**8.34**	**100%**

To demonstrate the efficiency of the proposed TSSA, DMSS, and DMCS, in this example, the six different methods, the SAG-SEFA using only TSSA (Type I), SE (Type II), GSE (Type III), SAG-SEFA (Type IV), SAG-SEFA using only TSSA and DMSS (Type V), and SAG-SEFA using only TSSA and DMCS (Type VI), are used. In Type I method, the groups are selected randomly to construct TNFC with different numbers of rules and each group performs crossover strategy independently. In Type II method, the traditional symbiotic evolution is used. In Type III method, the group-based symbiotic evolution (GSE) is adopted with fixed number of rules. In Type IV method, SAG-SEFA uses the proposed TSSA, DMSS, and DMCS to perform structure and parameter learning. In Type V method, SAG-SEFA uses only the proposed TSSA and DMSS; therefore, the each group performs crossover strategy independently. In Type VI method, SAG-SEFA uses only the proposed TSSA and DMCS; therefore, the groups are selected randomly to construct TNFC with different numbers of rules. In the six methods, the parameters are set according to [104]. In Type II and III methods, we determine the number of fuzzy rules by executing Type II and III methods with fixed string length for each specification of the number of fuzzy rules and then compute the average of the

CPU time. All the three are designed base on ISRL. The performance (accuracy and CPU time) compared of the six methods is shown in Table 5.12.

In Table 5.12, comparing Type II with Type III, the GSE outperform than SE because of the chromosomes that use to evaluate the solution locally can obtain better performance compared to systems of only one population be used to evaluate the solution. However, comparing Type I with Type III method, the Type I method needs few CPU time to balance the control system. The reason is that the TSSA can determine the suitable number of fuzzy rules automatically. However, in Type III method, the number of fuzzy rules is determined by trial-and-error testing. Therefore, the average of the generations of the Type III method is larger than Type I method. Comparing Type I with Type V method, it is observed that DMSS can reduce CPU time because of the chromosomes from suitable groups can be selected to construct TNFS models with different numbers of rules. Comparing Type I with Type VI method, it is observed that DMCS can reduce CPU time. This is because the chromosomes from suitable groups can be selected to cooperate for generating better solutions. As shown in Table 5.12, the SAG-SEFA (Type IV) performs better (CPU time) than other methods.

Table 5. 12: Performance comparison of six different methods in Example 1.

Method	CPU Time			
	Mean	Best	Worst	Std.
Type I	45.93	16.41	76.87	27.39
Type II	68.26	26.63	131.25	51.43
Type III	58.43	22.18	98.91	38.55
Type IV	**18.12**	**5.31**	**28.51**	**8.34**
Type V	27.37	12.83	36.55	15.17
Type VI	24.23	9.19	33.71	13.09

5.2 Tandem Pendulum Control System

Although the ISRL-SAEAs can obtain better performance than other methods in an inverted pendulum control system; however, in such classic setup of a pendulum balancing

problem, the task is too easy to find solutions quickly through random search. About this, in this example, a variety of extensions to a basic cart-pendulum balancing problem have been suggested. In [58]-[60], the author proposed several variations of an inverted pendulum control system. The most challenging extension of an inverted pendulum control system ([58]-[60]) is a tandem pendulum control system, where two pendulums of different length must be balanced synchronously. Therefore, a tandem pendulum control system is used to evaluate the proposed ISRL-SAEAs. As shown in Fig. 5.16, a tandem pendulum control system is the problem of learning how to balance two pendulums. There are four state variables in the system: θ_i, the angle of the *i*th pendulum; $\dot{\theta}_i$, the angular velocity of the *i*th pendulum. The only control action is *u*, which is the amount of force applied to cart to move it toward left or right. The system fails when the pendulum falls past a certain angle ($\pm 36^\circ$ is used here).

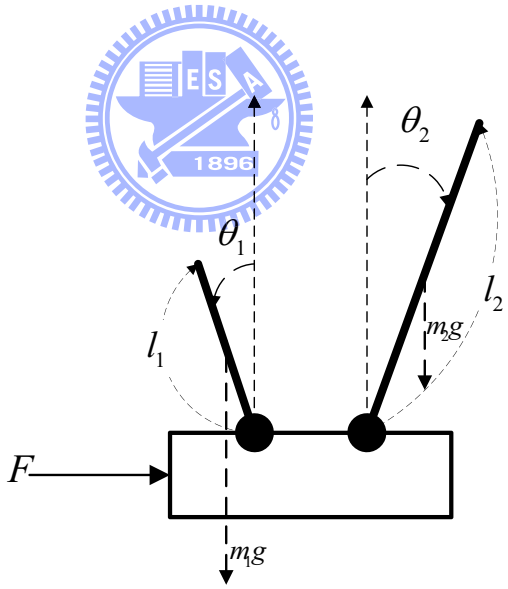


Figure 5. 16: The tandem pendulum control system.

The motion equations of the tandem pendulum control system ([58]-[60]) are described as follow:

$$J_i\ddot{\theta}_i = m_i g l_i \sin(\theta_i) - m_i l_i u \cos(\theta_i), \qquad i = 1,\ 2 \tag{5.12}$$

where for θ_i denotes the angle between *i*th pendulum and the vertical, J_i is the inertia moment

with respect to the pivot point, m_i is the mass of the ith pendulum, l_i is the distance between the center of mass and the pivot, g is the gravity acceleration, and u is the acceleration of the cart which is used as the control input.

By setting the potential energy of ith pendulum at the vertical to be 0, its energy consists of the kinetic energy of rotation with respect to the pivot and the potential energy is expressed as the following equation:

$$E_i(\theta_i, \dot{\theta}_i) = \frac{1}{2} J_i \dot{\theta}_i^2 + m_1 g l_1 (\cos(\theta_1) - 1) \tag{5.13}$$

A control strategy attempts to drive E_1 and E_2 to zero is obtained according to the Lyapunov function as show below:

$$V = \frac{1}{2} E_i^2, \ i = 1, \ 2. \tag{5.14}$$

Using the fallowing equation:

$$\dot{E}_i = -m_i g l_i u \dot{\theta}_i \cos(\theta_i). \tag{5.15}$$

We can obtain

$$\dot{V} = -Gu, \tag{5.16}$$

where

$$G = \sum_{i=1}^{2} m_i g l \dot{\theta}_i \cos(\theta_i). \tag{5.17}$$

The parameters used for the tandem pendulum control system are shown in Table 5.13.

Table 5. 13: The parameters for the tandem pendulum control system.

Parameters	Description	Value
θ	Angle of the pendulum	[-36, 36] deg.
u	Force applied to cart	[-10, 10] N
l_1	Half length of 1st pendulum	0.5m
l_2	Half length of 2nd pendulum	0.05m
m_1	mass of the 1st pendulum	0.1kg
m_2	mass of the 2nd pendulum	0.01kg

The purpose of this control task is to determine a sequence of forces applying to the cart to balance the pendulums upright, and maintains the cart as stationary as possible. Hence, we define a goal set comprising near-upright and near-stationary states as $G_1 = \left\{(\theta_i, \dot{\theta}_i): \left\|(\theta_i, \dot{\theta}_i)\right\| \le 0.001\right\}$. When the state of the tandem pendulum control system is in G_1, according to Eq. 5.13, the total mechanical energy E of the system is *0*. Let $q = \theta$, we define a Lyapunov function $V(q, \dot{q}) = \frac{1}{2}\sum_{i=1}^{2} E_i(q, \dot{q})^2$. The purpose of this control problem can be transformed to the problem of achieving $V(q, \dot{q}) = 0$. So we define another goal set $G_2 = \{(q, \dot{q}) : V(q, \dot{q}) = 0\}$.

According to Eq. 5.15, the derivative of E is proportional to the product of the speed of angular and input force. Hence in this paper, following [32], a control law for the learning agent based on Lyapunov analysis is proposed as follows:

$$P(q, \dot{q}) = \begin{cases} sgn(\dot{q})F, & \text{if } \mathrm{E}(q, \dot{q}) < 0 \\ 0, & \text{if } \mathrm{E}(q, \dot{q}) = 0 \end{cases} \tag{5.18}$$

where $sgn(\dot{q}) = \{1 \text{ if } x \ge 0, \text{ and -1 otherwise}\}$ and F is the output of the TNFC that is limited in [-10,10]. $P(q, \dot{q})$ guarantees the descent of the Lyapunov function; as a result, this control law satisfies the Theorem 4.1 defined in [32]. Denote the state of the environment at time step $t = (q_t, \dot{q}_t) = s_t$. Theorem 4.1 tells that the agent will bring the environment to G_1 within $\lceil L(s_0)/\Delta \rceil$ steps, and remain the environment in the set $\{s: L(s_{t+1}) \le L(s_t)\}$. In our simulation, since the descent step size can not be ensured. Theorem 4.1 can be reduced to the form that the agent will bring the environment to G_1 and remain the environment until it achieves G_2 eventually, if the controlling time step is long enough.

In the simulation of the tandem pendulum control system, the original successful region of the variables is $-36^\circ \le \theta_i \le 36^\circ$. The strict successful region of θ is described in Eq. 5.13. The constraints on the variables are $-36^\circ \le \theta_i \le 36^\circ$ and -10N$\le u \le$ 10N. A control strategy

is deemed successful if it can meet the control goal (θ and $\dot{\theta}$ decade to zero). The four input variables (θ_1, $\dot{\theta}_1$, θ_2, $\dot{\theta}_2$) and the output $u(t)$ are normalized between 0 and 1, and $u(t)$: [-10, 10]. The four normalized state variables are used as inputs to the TNFC. The values are floating-point numbers assigned to the SAEAs initially. The fitness function is defined according to Eq. 4.1-4.5.

In this example, the performance evaluation of the SAEAs consists of the HEA, SACG-SE, and SAG-SEFA. In the following sections, the performances of three methods are discussed.

5.2.1 Evaluating performance of the HEA

The initial parameters of the proposed ISRL-HEA in this example are determined by parameter exploration ([104). The parameters set for the ISRL-HEA are shown in Table 5.14.

Table 5. 14: The initial parameters of ISRL-HEA before training.

Parameters	Value	Parameters	Value
N_c	80	*Stable _TimeSteps*	5000
Crossover Rate	0.4	*Thres _TimeStep*	1000
Mutation Rate	0.25	*ERSTimes*	50
$[\sigma_{\min}, \sigma_{\max}]$	[0, 2]	A	10
$[m_{\min}, m_{\max}]$	[0, 2]	λ	0.01
$[w_{\min}, w_{\max}]$	[-20, 20]	η	7
$[R_{\min}, R_{\max}]$	[3, 12]	*Generations*	500
Thres _StableTimeSteps	500		

In this example, the coding of a rule in a chromosome is the form in Fig. 3.1 in Section 3.1. A total of thirty runs were performed. Each run started at the different initial state ($\dot{\theta}_i$ are set for 0, θ_i are set randomly within a predefined range). The learning curves of ISRL-HEA are shown in Fig. 5.17. In this figure, there are thirty runs each run represents that how soon the TNFC can meet the goal state. The fitness value is defined according to Eqs. 4.1-4.5. The higher fitness value by the end of each run represents that the sooner the plant meets the goal

set. When the ISRL-HEA is stopped, the best string from the population in the final generation is selected and applied on the testing phase of a tandem pendulum control system. After performing the MCGA, the final average number of rules is 6.

The testing results, which consist of the pendulums angle and the pendulums angular velocity (in degrees/seconds) are shown in Fig. 5.18. A total of thirty runs were executed in the testing phase. Each line in Fig. 5.18 represents a single run that starts form a different initial state. The results shown in this figure are the first 1,000 of 6,000 control time steps ($Thres\_TimeStep + Stable\_TimeSteps$). As shown in Fig. 5.18, the ISRL-HEA successfully controlled the tandem pendulum control system in all thirty runs (the pendulums angle, pendulums angular velocity decrease to 0).

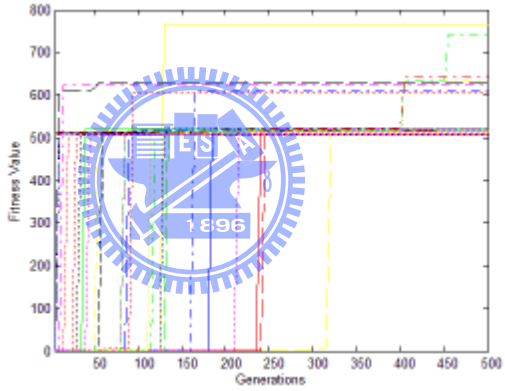
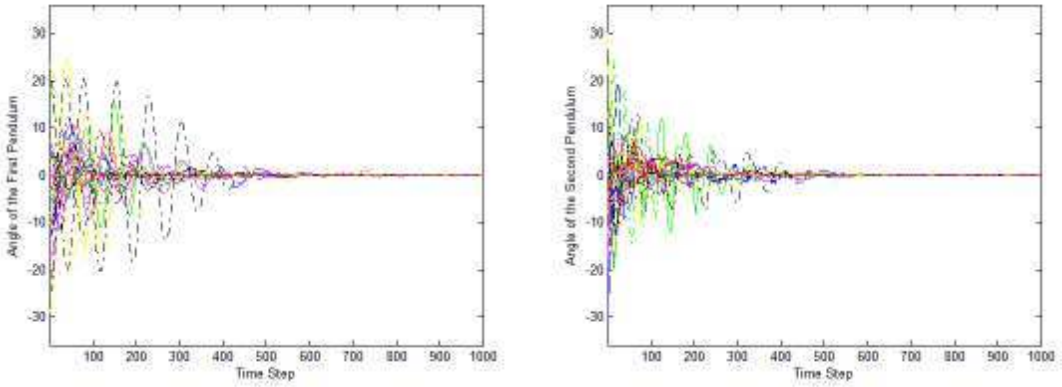
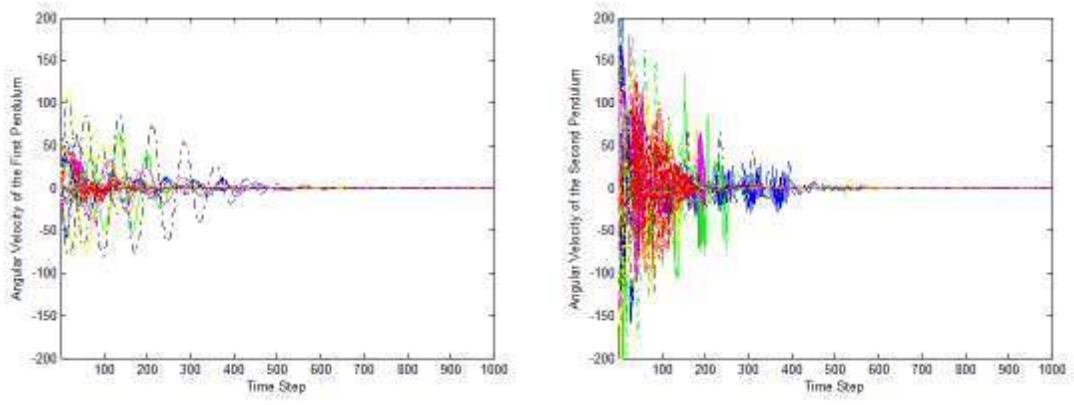


Figure 5. 17: The learning curves of the ISRL-HEA.



(a) (b)



(c) (d)

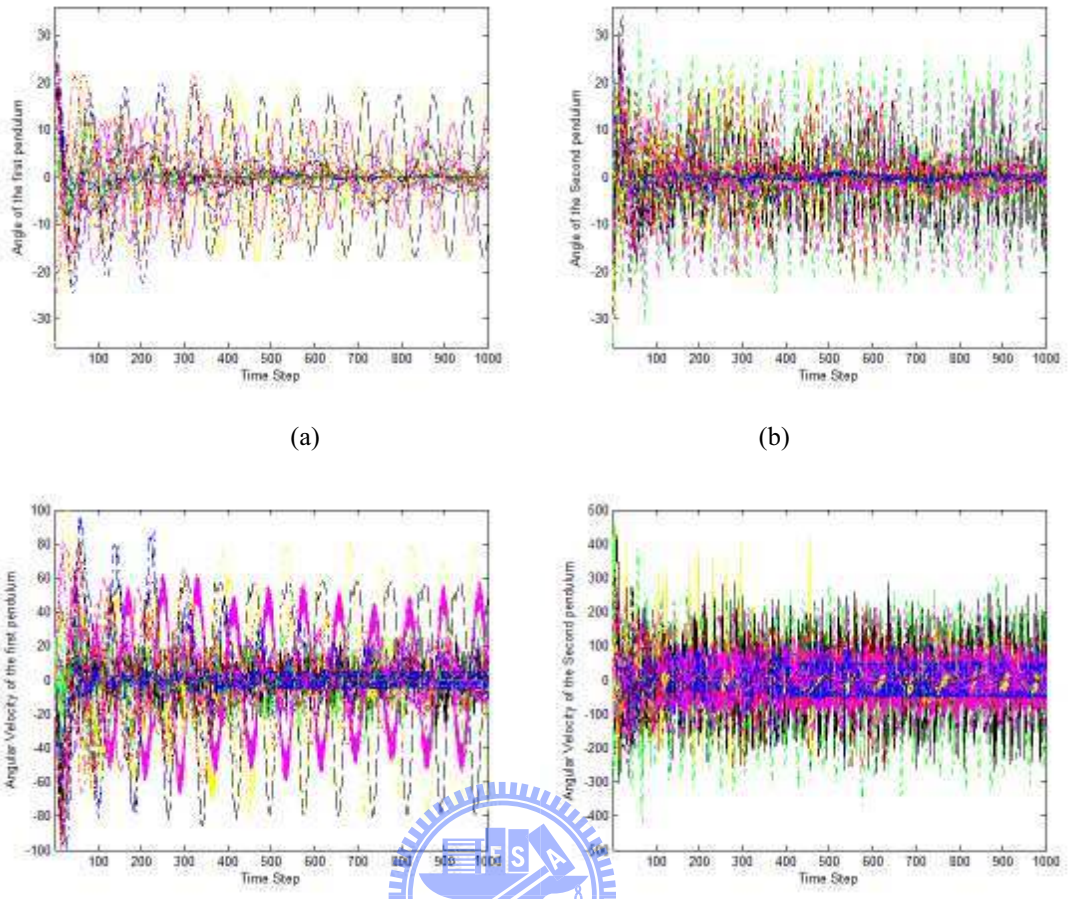
Figure 5. 18: Control results of the tandem pendulum control system using the ISRL-HEA. (a) Angle of the first pendulum. (b) Angle of the second pendulum. (c) Angular velocity of the first pendulum. (d) Angular velocity of the second pendulum.

As well as Section 5.1.1, the R-SE ([29]) and R-GA ([26]) were applied to this example to compare to the performance of the ISRL-HEA. The parameters are found using the method given in [104]. In R-SE ([29]) and (R-GA [26]), the reinforcement signal is designed based on time-step reinforcement architecture ([18]-[20]). The fitness function in R-SE and R-GA to train the TNFC is defined according to

$$Fitness\_Value = TIME\_STEP \tag{5.19}$$

where *TIME_STEP* represents how long the experiment is a "success" in one generation. In this example, Eq. 5.19 represents how long before the pendulum falls apart a certain angle ($\pm 36^{\circ}$ is used here). A control strategy is deemed successful if it can balance pendulums for 6,000 time steps.

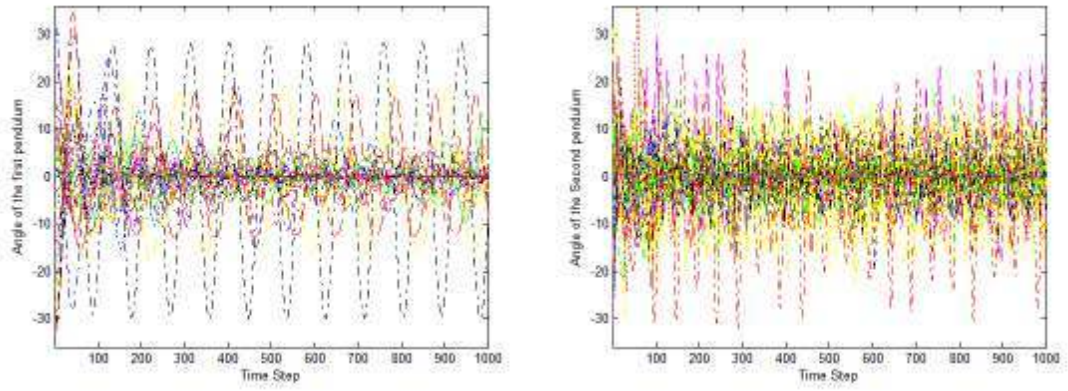
The simulation was carried out for 30 runs. The testing results of the R-SE and R-GA are shown in Figs. 5.19 and 5.20. The results shown in these figures are the first 1,000 of 6,000 control time steps. As shown in Figs. 5.19 and 5.20, not every line meets the control goal (θ_i and $\dot{\theta}_i$ decay to zero). It's obvious that the ISRL-HEA obtains better result when compared with [29] and [26], since θ_i and $\dot{\theta}_i$ of the ISRL swing in a smaller range near zero.



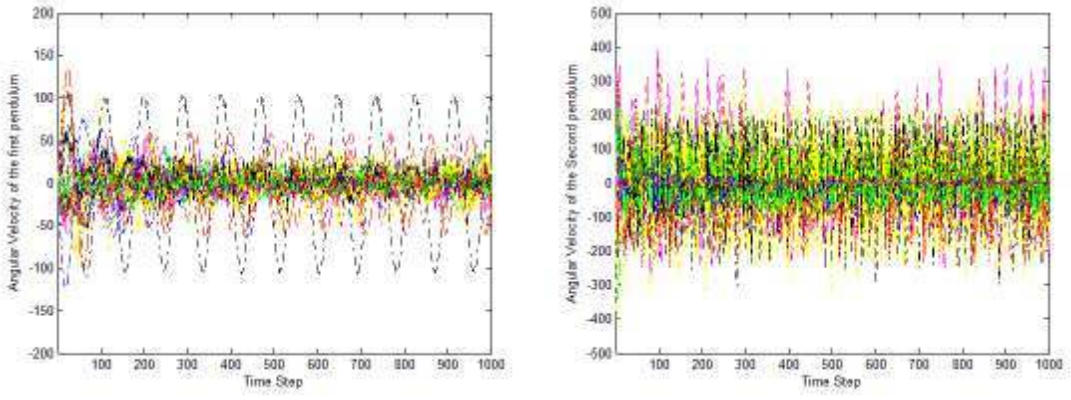
(a) (b)

(c) (d)

Figure 5. 19: Control results of the tandem pendulum control system using the R-SE. (a) Angle of the first pendulum. (b) Angle of the second pendulum. (c) Angular velocity of the first pendulum. (d) Angular velocity of the second pendulum.



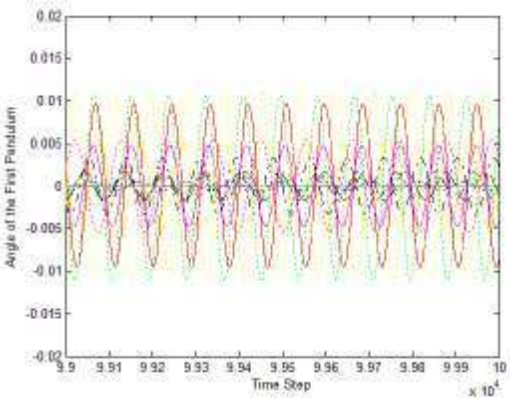
(a) (b)



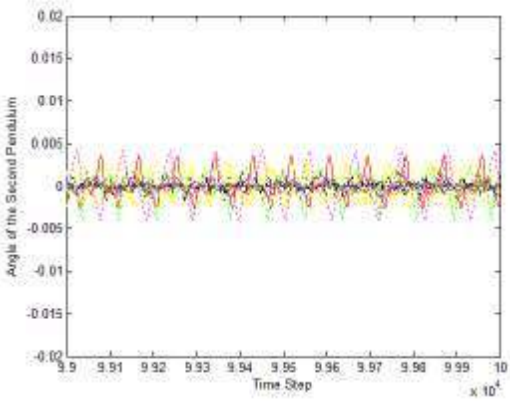
(c) (d)

Figure 5. 20: Control results of the tandem pendulum control system using the R-GA. (a) Angle of the first pendulum. (b) Angle of the second pendulum. (c) Angular velocity of the first pendulum. (d) Angular velocity of the second pendulum.

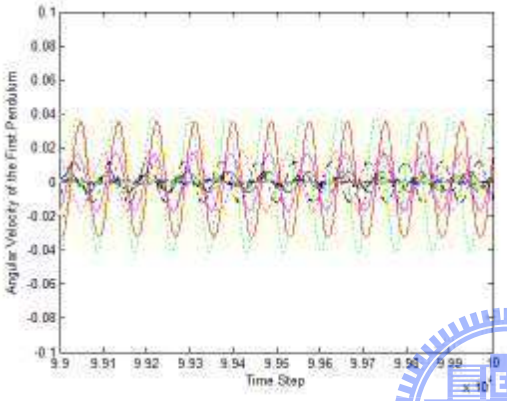
In the further simulation, we select the best-trained individual of the proposed ISRL-HEA, R-GA and R-SE in the training phase, and extend the control time steps to 100,000 in the testing phase. The simulation results, which consist of the pendulums angle and pendulums angular velocity, are shown in Fig. 5.21. Each line in Fig. 5.21 represents the result of the last 1000 time steps in a run that starts from the different initial state. As shown in Fig. 5.21 (e)-(l), not every line meets the control goal G_1 in the R-SE and R-GA. Moreover, the pendulums angle may swing outside the boundary at the last 1000 time steps. However, in the proposed ISRL-HEA, each line can meet the control goal G_1 and the pendulums are kept upright during the last 1000 time steps. The percentage that the R-SE and the R-GA controls the plant to G_1 are 53% (with 14 runs that the plant unreach G_1 and 4 out of 13 runs that the pendulum swings outside the boundary) and 50% (with 15 runs that the plant unreach G_1 and 5 out of 14 runs that the pendulum swings outside the boundary) respectively. However, in the ISRL-HEA, the percentage that the plant remains in G_1 during the last 1000 time steps is 100%. It's obvious that the ISRL allows the pendulums angle, the pendulums angular velocity and the cart velocity to swing a small range near zero and stabilize the control system.


Angle of the First Pendulum
Time Step
x 10^4

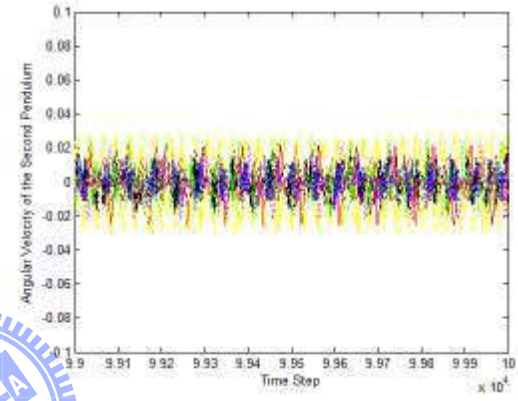
(a)


Angle of the Second Pendulum
Time Step
x 10^4

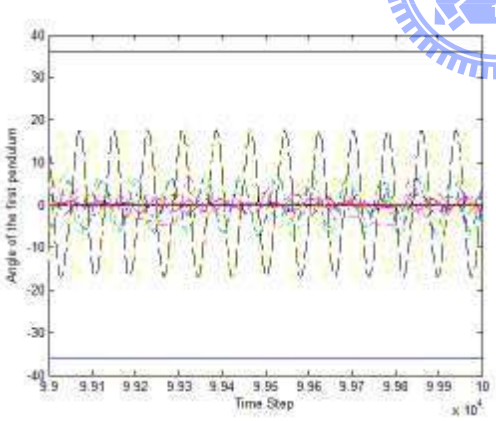
(b)


Angular Velocity of the First Pendulum
Time Step
x 10^4

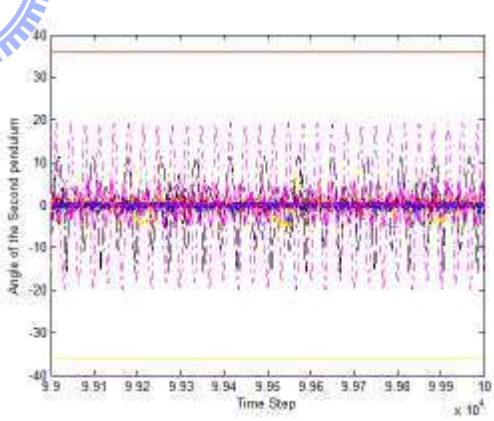
(c)


Angular Velocity of the Second Pendulum
Time Step
x 10^4

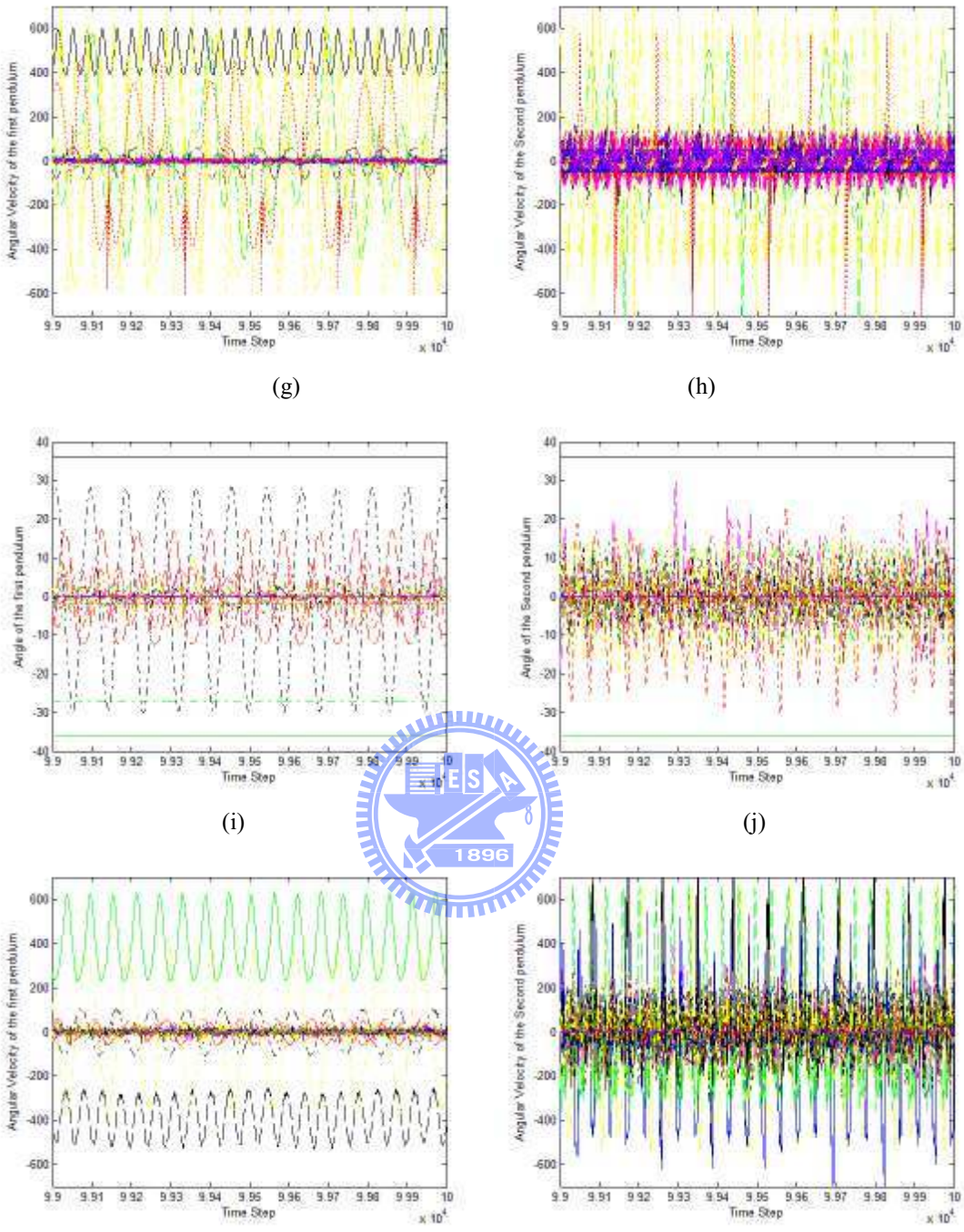
(d)


Angle of the first pendulum
Time Step
x 10^4

(e)


Angle of the Second pendulum
Time Step
x 10^4

(f)



(g) (h)

(i) (j)

(k) (l)

Figure 5. 21: Control results of the tandem pendulum control system. (a) Angle of the first pendulum of ISRL-HEA. (b) Angle of the second pendulum of ISRL-HEA. (c) Angular velocity of the first pendulum of ISRL-HEA. (d) Angular velocity of the second pendulum of ISRL-HEA. (e) Angle of the first pendulum of R-SE. (f) Angle of the second pendulum of R-SE. (g) Angular velocity of the first pendulum of R-SE. (h) Angular velocity of the second pendulum of R-SE. (i) Angle of the first pendulum of R-GA. (j) Angle of the second pendulum of R-GA. (k) Angular velocity of the first pendulum of R-GA. (l) Angular velocity of the second pendulum of R-GA.

The accuracy and CPU time comparison of ISRL-HEA, R-SE, and R-GA are shown in Table 5.15. The ISRL-HEA needs less CPU time than R-SE and R-GA. The reason is that the ISRL adopts a strict restriction in the earlier time steps and evaluates the control system by how soon the plant can meet the control goal. The individuals in ISRL-HEA with better performance mean it controls the plant to the goal set sooner.

The GENITOR ([57]), SANE ([96]), TDGAR ([20]), CQGAF ([43]), ERDSE ([44]), and enforce sub-population (ESP) ([40]) methods have been applied to the same control task and the simulation results are listed in Table 5.15. The accuracy of the controller meets the control goal and keep the pendulum in 100000 time steps and the CPU time are shown in Table 5.15. A total of thirty runs were executed. Each run started at the different initial state. The initial parameters of these methods ([57], [96], [20], [43], [44], and [40]) are determined according to [104]. In these methods, we determine the size of network structure by executing algorithms with fixed string length for each specification of the size of network structure and then compute the average of the generations. As shown in Table 5.15, the proposed ISRL-HEA is more feasible and effective when compared with other existing models ([26], [29], [57], [96], [20], [43], [44], and [40]).

Table 5. 15: Performance comparison of various existing models.

Method	CPU time				Accuracy
	Mean	Best	Worst	Std.	
GENITOR ([57])	412.49	91.85	572.54	109.69	56%
SANE ([96])	225.73	63.87	276.54	60.23	57%
R-GA ([26])	218.34	49.56	251.68	52.97	52%
R-SE ([29])	192.67	47.49	241.67	47.38	53%
TDGAR ([20])	231.45	56.37	258.74	54.05	56%
ESP ([40])	187.96	39.54	238.95	46.32	54%
ERDSE ([44])	123.73	26.18	214.51	37.89	55%
CQGAF ([43])	105.52	24.67	203.18	34.42	53%
ISRL-HEA	**87.23**	**23.54**	**113.18**	**28.92**	**100%**

5.2.2 Evaluating performance of the SACG-SE

The initial parameters of the proposed ISRL-SACG-SE in this example are determined by parameter exploration ([104]). The parameters set for the ISRL-SACG-SE are shown in Table 5.16.

Table 5. 16: The initial parameters of ISRL-SACG-SE before training.

Parameters	Value	Parameters	Value
N_c	30	$Stable\_TimeSteps$	5000
Crossover Rate	0.35	$Thres\_TimeStep$	1000
Mutation Rate	0.25	$TSSATimes$	50
$[\sigma_{\min}, \sigma_{\max}]$	[0, 2]	A	10
$[m_{\min}, m_{\max}]$	[0, 2]	λ	0.01
$[w_{\min}, w_{\max}]$	[-20, 20]	η	7
$[R_{\min}, R_{\max}]$	[3, 12]	*Generations*	500
P_{size}	20	$Thres\_StableTimeSteps$	500

A total of thirty runs were performed. Each run started at the different initial state ($\dot{\theta}_i$ are set for 0, θ_i are set randomly according to the predefined ranges). After performing the TSSA, the final average number of rules is 6. The learning curves of the SACG-SE after thirty runs are shown in Fig. 5.22. In this figure, there are thirty runs each run represents that how soon the TNFC can meet the goal state. When SACG-SE is stopped, the best combination of strings from the groups in the final generation is selected and tested on the tandem pendulum control system.

The simulation was carried out for thirty runs. The simulation results, which consist of the pendulums angle and angular velocity of pendulums, are shown in Fig. 5.23. Each line in Fig. 5.23 represents each run with a different initial state. The results shown in this figure are the first 1,000 of 6,000 control time steps ($Thres\_TimeStep + Stable\_TimeSteps$). As shown in Fig. 5.23, the ISRL-SACG-SE successfully controlled the tandem pendulum control system in all thirty runs (the pendulums angle and pendulums angular velocity decrease to 0).

As well as ISRL-HEA, we select the best-trained individual of the proposed ISRL-SACG-SE in the training phase, and extend the control time steps to 100,000 in the testing phase. The simulation results, which consist of the pendulums angle and the pendulums angular velocity, are shown in Fig. 5.24. Each line in Fig. 5.24 represents the result of the last 1000 time steps in a run that starts from the different initial state. As shown in Fig. 5.24, each line can meet the control goal G_1 and the pendulums are kept upright during the last 1000 time steps. The percentage that the plant remains in G_1 during the last 1000 time steps is 100%. It's obvious that the ISRL allows the pendulums angle, the pendulums angular velocity and the cart velocity to swing a small range near zero and stabilize the control system.

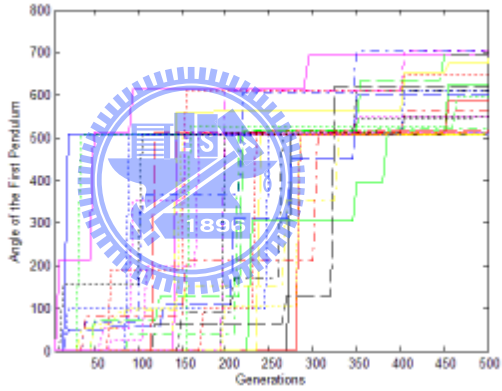
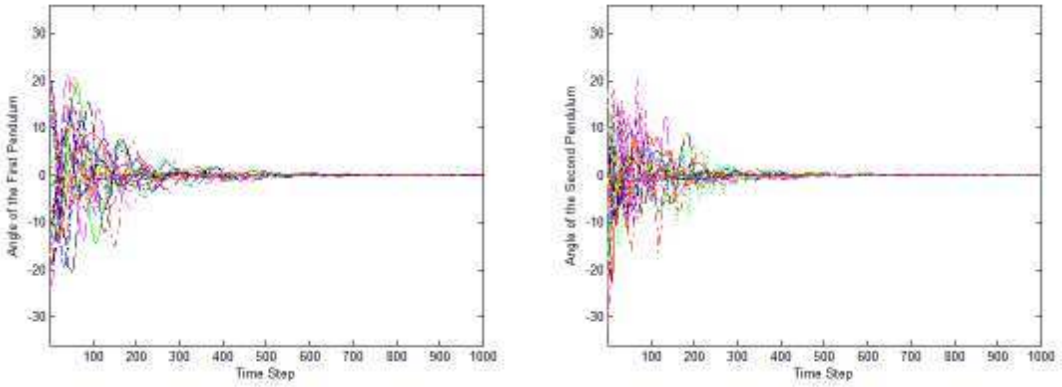
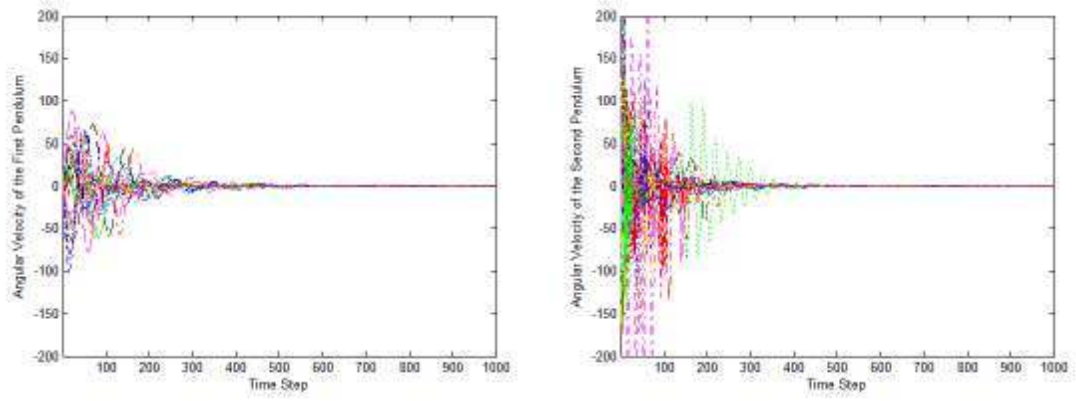


Figure 5. 22: The learning curve of the SACG-SE.

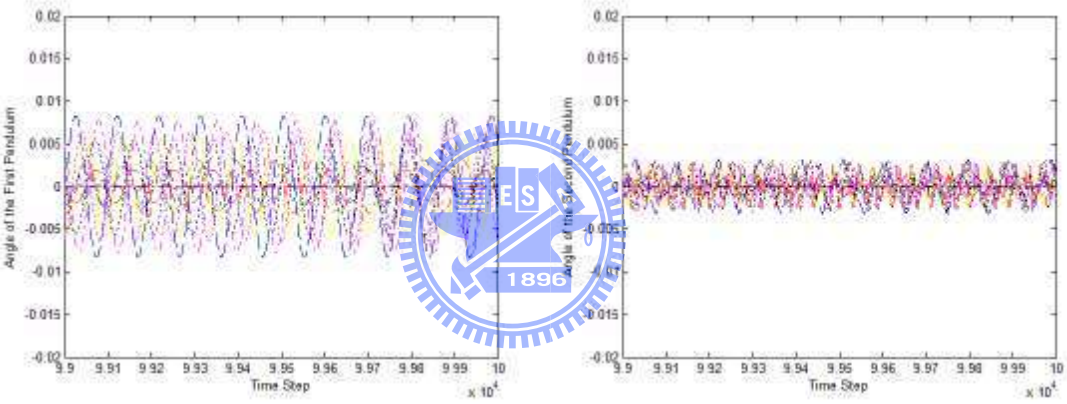


(a) (b)

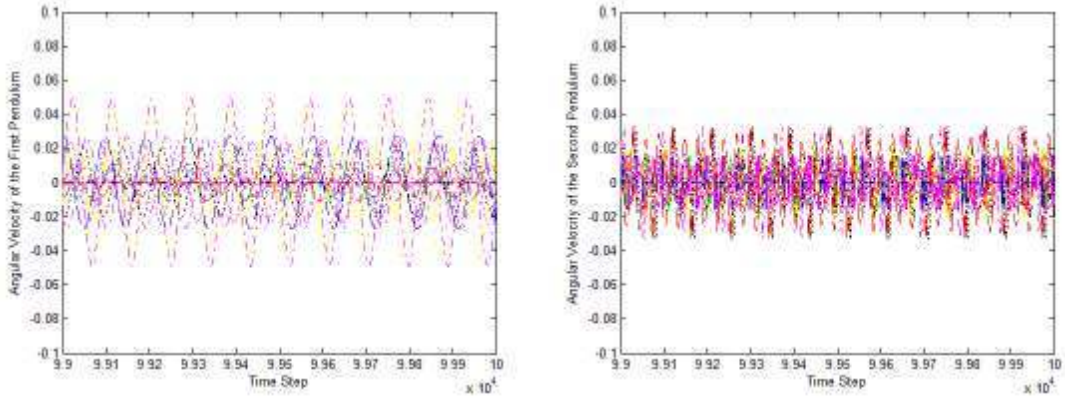


(c) (d)

Figure 5. 23: Control results of the tandem pendulum control system using the ISRL-SACG-SE (first 1000 time). (a) Angle of the first pendulum. (b) Angle of the second pendulum. (c) Angular velocity of the first pendulum. (d) Angular velocity of the second pendulum.



(a) (b)



(c) (d)

Figure 5. 24: Control results of the tandem pendulum control system using the ISRL-SACG-SE (last 1000 time). (a) Angle of the first pendulum. (b) Angle of the second pendulum. (c) Angular velocity of the first pendulum. (d) Angular velocity of the second pendulum.

In this example, in order to demonstrate the effectiveness and efficiency of the proposed ISRL-SACG-SE, the R-SE ([29]) and R-GA ([26]) are used to compare with ISRL-SACG-SE. As shown in Fig. 5.21 (e)-(l), the accuracy of the TNFC with the R-SE and R-GA the pendulum does not swing outside the boundary after 6,000 time steps are 53% and 50%. However, in the ISRL-SACG-SE, the accuracy of the TNFC success meets the control goal and keeps the pendulums in 100,000 time steps is 100%. As shown in Figs. 5.21 and 5.24, the ISRL-SACG-SE can perform better than the R-SE and R-GA.

The accuracy and CPU time comparison of the ISRL-SACG-SE, R-SE, and R-GA are shown in Table 5.17. As shown in Table 5.17, when compared with the traditional reinforcement signal design, the proposed ISRL can reduce the CPU time and always control the plant to the goal set. Moreover, the SACG-SE not only can determine the fuzzy rules automatically without trail and error testing but also let the chromosomes that perform well to cooperate for generating better solutions.

The GENITOR ([57]), SANE ([96]), TDGAR ([20]), CQGAF ([43]), ERDSE ([44]), and enforce sub-population (ESP) ([40]) methods have been applied to the same control problem and the simulation results are listed in Table 5.17. The accuracy and CPU time are shown in Table 5.17. A total of thirty runs were performed. Each run started at a different initial state. The initial parameters of these methods ([57], [69], [20], [43], [44], and [40]) are determined according to [104]. The control time steps for testing are extended to 100,000 time steps. As shown in Table 5.17, the proposed ISRL-SACG-SE is more feasible and effective when compared with other existing models ([26], [29], [57], [96], [20], [43], [44], and [40]).

Table 5. 17: Performance comparison of various existing models in Example 2.

Method	CPU time				Accuracy
	Mean	Best	Worst	Std.	
GENITOR [57]	412.49	91.85	572.54	109.69	56%
SANE [96]	225.73	63.87	276.54	60.23	57%
R-GA [26]	218.34	49.56	251.68	52.97	52%
R-SE [29]	192.67	47.49	241.67	47.38	53%
TDGAR [20]	231.45	56.37	258.74	54.05	56%
ESP [40]	187.96	39.54	238.95	46.32	54%
ERDSE [44]	123.73	26.18	214.51	37.89	55%
CQGAF [43]	105.52	24.67	203.18	34.42	53%
ISRL-SACG-SE	**65.74**	**18.50**	**84.43**	**23.18**	**100%**

5.2.3 Evaluating performance of the SAG-SEFA

The initial parameters of the proposed ISRL-SAG-SEFA in this example are determined by parameter exploration ([104]). The parameters set for the ISRL-SAG-SEFA are shown in Table 5.18.

Table 5. 18: The initial parameters of ISRL-SAG-SEFA before training.

Parameters	Value	Parameters	Value
P_{size}	18	$Stable\_TimeSteps$	5000
N_C	20	$Thres\_TimeStep$	1000
$Selection\_Times$	250	$TSSAimes$	30
$NormalTimes$	20	A	10
$SearchingTimes$	25	λ	0.01
$ExploringTimes$	30	η	7
Crossover Rate	0.5	$Generations$	500
Mutation Rate	0.2	$Minimum\_Suppor$	$TransactionNum/2$
$ThreadFitnessvalue$	550	$Thres\_StableTimeSteps$	500

The coding of a rule in a chromosome is the form given in Fig. 3.10. The values are floating-point numbers initially assigned using the ISRL-SAG-SEFA. A total of thirty runs were performed in this simulation. Each run started at a different initial state ($\dot{\theta}_i$ are set for 0,

θ_i are set randomly according to the predefined ranges). After performing the TSSA, the final average number of rules is 5.

The learning curves of the ISRL-SAG-SEFA after thirty runs are shown in Fig. 5.25. In this figure, there are thirty runs each run represents that how soon the TNFC can meet the goal state. When ISRL-SAG-SEFA is stopped, the best combination of strings from the groups in the final generation is selected and tested on the tandem pendulum control system. The successful results, which consist of the pendulums angle and angular velocity of the pendulums, are shown in Fig. 5.26. Each line in Fig. 5.26 represents each run with a different initial state. The results shown in this figure are the first 1,000 of 6,000 control time steps ($Thres\_TimeStep + Stable\_TimeSteps$). As shown in Fig. 5.26, the ISRL-SAG-SEFA successfully controlled the tandem pendulum control system in all thirty runs (the pendulums angle and pendulums angular velocity decrease to 0).

As well as ISRL-HEA and ISRL-SACG-SE, we select the best-trained individual of the proposed ISRL-SAG-SEFA in the training phase, and extend the control time steps to 100,000 in the testing phase. The simulation results, which consist of the pendulums angle, the pendulums angular velocity, and the cart velocity, are shown in Fig. 5.27. Each line in Fig. 5.27 represents the result of the last 1000 time steps in a run that starts from the different initial state. As shown in Fig. 5.27, each line can meet the control goal G_1 and the pendulums are kept upright during the last 1000 time steps. However, in the ISRL-SAG-SEFA, the percentage that the plant remains in G_1 during the last 1000 time steps is 100%. It's obvious that the ISRL allows the pendulums angle, the pendulums angular velocity to swing a small range near zero and stabilize the control system.

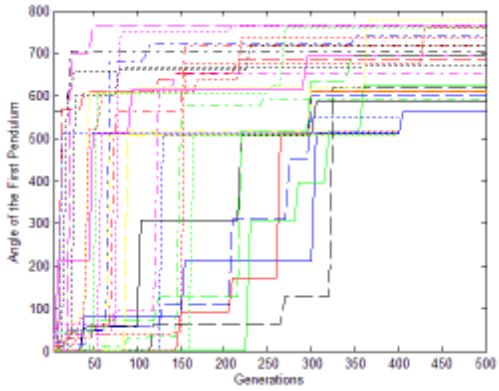


Figure 5. 25: The learning curve of the SAG-SEFA.

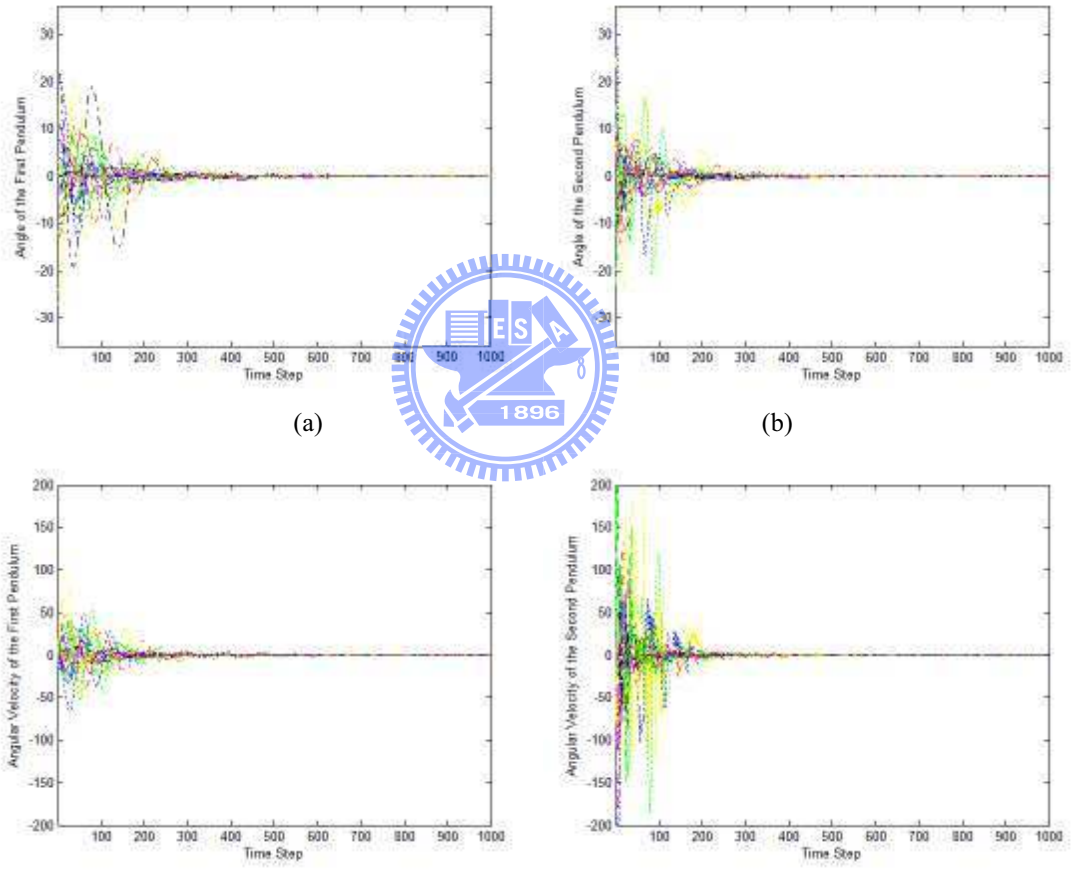


Figure 5. 26: Control results of the tandem pendulum control system using the ISRL-SAG-SEFA (first 1000 time). (a) Angle of the first pendulum. (b) Angle of the second pendulum. (c) Angular velocity of the first pendulum. (d) Angular velocity of the second pendulum.

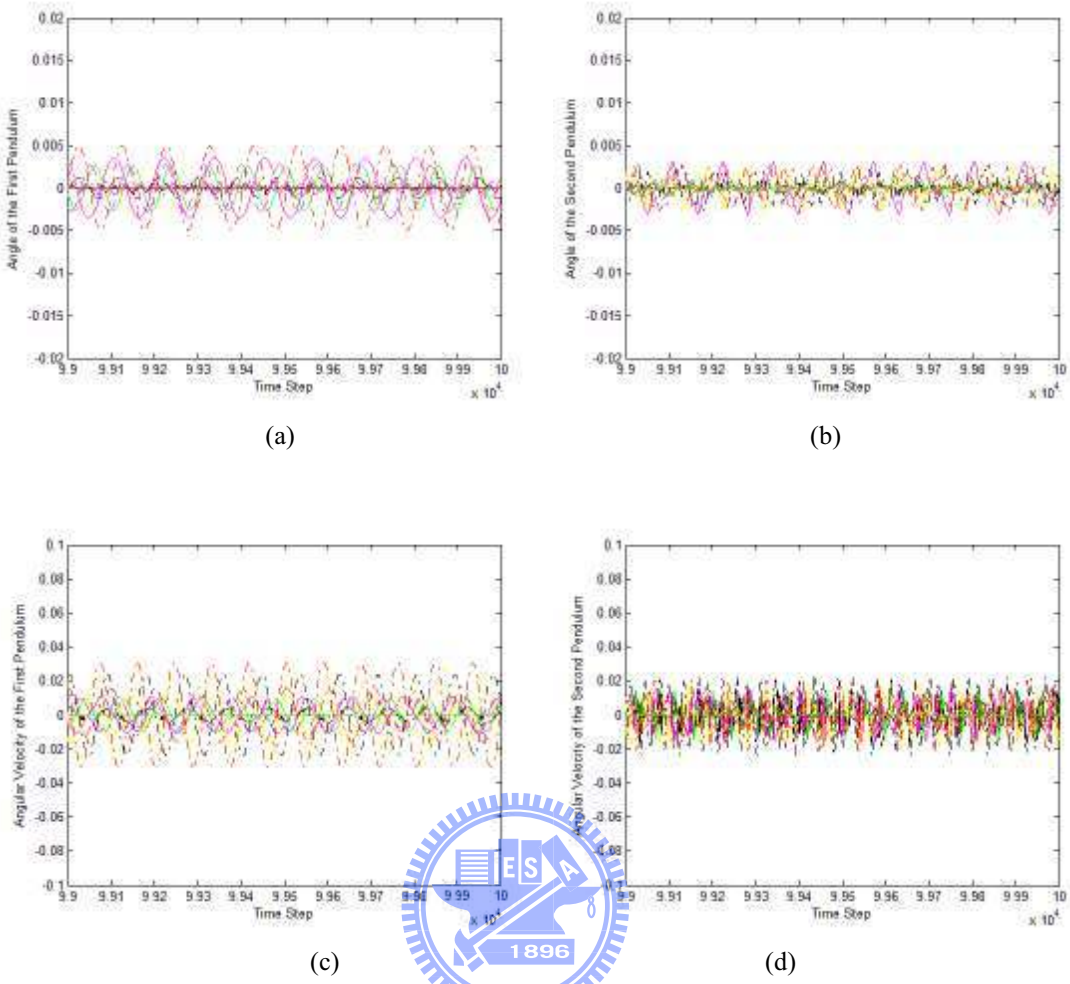


Figure 5. 27: Control results of the tandem pendulum control system using the ISRL-SAG-SEFA (last 1000 time). (a) Angle of the first pendulum. (b) Angle of the second pendulum. (c) Angular velocity of the first pendulum. (d) Angular velocity of the second pendulum.

In this example, in order to demonstrate the effectiveness and efficiency of the proposed ISRL-SAG-SEFA, the R-SE ([29]) and R-GA ([26]) are used to compare with ISRL-SAG-SEFA. As shown in Fig. 5.21 (e)-(l), the accuracy of the TNFC with the R-SE and R-GA the pendulum does not swing outside the boundary after 6,000 time steps are 53% and 50%. However, in the ISRL-SAG-SEFA, the accuracy of the TNFC success meets the control goal and keeps the pendulums in 100,000 time steps is 100%. As shown in Fig. 5.21 and 5.27, the ISRL-SAG-SEFA can perform better than the R-SE and R-GA.

The GENITOR ([57]), SANE ([96]), TDGAR ([20]), CQGAF ([43]), ERDSE ([44]), and ESP ([40]) have been applied to the same control problem. Their simulation results are listed

in Table 5.19. The accuracy and CPU time are shown in Table 5.19. The initial parameters of these methods ([57], [69], [20], [43], [44], and [40]) are determined according to [104]. The control time steps for testing are extended to 100,000 time steps. As shown in Table 5.19, the proposed ISRL-SAG-SEFA is more feasible and effective when compared with other existing models (([26], [29], [57], [96], [20], [43], [44], and [40]).

Table 5. 19: Performance comparison of various existing models in Example 2.

Method	CPU time				Accuracy
	Mean	Best	Worst	Std.	
GENITOR ([57])	412.49	91.85	572.54	109.69	56%
SANE ([96])	225.73	63.87	276.54	60.23	57%
R-GA ([26])	218.34	49.56	251.68	52.97	52%
R-SE ([29])	192.67	47.49	241.67	47.38	53%
TDGAR ([20])	231.45	56.37	258.74	54.05	56%
ESP ([40])	187.96	39.54	238.95	46.32	54%
ERDSE ([44])	123.73	26.18	214.51	37.89	55%
CQGAF ([43])	105.52	24.67	203.18	34.42	53%
ISRL-SAG-SEFA	**41.91**	**11.36**	**62.54**	**17.32**	**100%**

Chapter 6
Conclusion

The goal of this dissertation is to provide a stable and robust way for applying evolutionary algorithm to neuro-fuzzy controllers. In order for a proposed method to be stable and robust, it must not only perform the stability analysis but also decide the number of fuzzy rules automatically. This dissertation proposes a complete approach to achieve this goal. Therefore, improved safe reinforcement learning based self adaptive evolutionary algorithms (ISRL-SAEAs) for neuro-fuzzy controller is proposed. The ISRL-SAEAs consist of two parts: improved safe reinforcement learning (ISRL) and self adaptive evolutionary algorithms (SAEAs). This chapter summarizes the contributions of this dissertation of these two parts (ISRL and SAEAs), and then makes a conclusion and discusses the future research.

6. 1 Contributions

The proposed ISRL-SAEAs can be divided into two parts: improved safe reinforcement learning (ISRL) and self adaptive evolutionary algorithms (SAEAs). Each of these two parts has different contributions. The contributions are discussed as follows.

1. Self Adaptive Evolutionary Algorithms (SAEAs)

   The goals of SAEAs are to reduce the number of parameters in traditional evolution method (ie. initial network size), consider both of cooperation and specialization, and determine the suitable groups for performing selection and crossover steps. The SAEAs consist of three method: the hybrid evolutionary algorithm (HEA), self adaptive group cooperation based symbiotic evolution (SAGC-SE), and self adaptive group based

symbiotic evolution using FP-growth algorithm (SAG-SEFA). The contributions of these three methods are discussed as follows.

(1) Hybrid Evolutionary Algorithm (HEA)

The HEA has structure-and-parameter learning ability. That is, it can determine the average optima number of fuzzy rules and tune the free parameters in the TNFC. The proposed learning method also processes variable length of the chromosomes in a population. Computer simulations have shown that the proposed HEA has a better performance than the other methods. The well performance of each component of the proposed HEA has been demonstrated.

(2) Self Adaptive Group Cooperation based Symbiotic Evolution (SAGC-SE)

The SAGC-SE can determine number of fuzzy rules automatically by using two-step self-adaptive algorithm (TSSA), evaluate the fuzzy rule locally by using group-based symbiotic evolution, and let groups to cooperate with each other to generate the better chromosomes by using an elites-base compensation crossover strategy (ECCS). The advantages of the proposed SAGC-SE are summarized as follows: 1) the SAGC-SE can determine the average optima number of fuzzy rules; 2) the SAGC-SE uses group-based population to evaluate each fuzzy rule locally; 3) the SAGC-SE uses the ECCS to let the better solutions from different groups to cooperate for generating better solutions in the next generation.

Computer simulations have shown that the SAGC-SE that considers both of cooperation and specialization has a better performance than the HEA. The well performance of each component of the SAGC-SE has also been demonstrated.

(3) Self Adaptive Group based Symbiotic Evolution using FP-growth Algorithm (SAG-SEFA)

The proposed SAG-SEFA has structure and parameter learning ability. That is, it can determine the suitable number of fuzzy rules and efficiently tune the

parameters in the TNFC. The goal of the SAG-SEFA is to determine the suitable groups to perform the selection and crossover steps. The advantages of the proposed SAG-SEFA are summarized as follows: 1) the SAG-SEFA uses the group-based population so that each group represents only one fuzzy rule; 2) the TSSA is used to determine the suitable number of rules; 3) the SAG-SEFA uses group-based population to evaluate the fuzzy rule locally; 4) the DMSS and DMCS are used to select the suitable groups to perform the selection step and crossover step; 5) it performs better and converges more quickly than some genetic methods.

Computer simulations have shown that the proposed SAG-SEFA that considers how to select the suitable groups for performing selection and crossover steps has a better performance than the SAGC-SE. The well performance of each component of the proposed SAG-SEFA has also been demonstrated.

2. Improved Safe Reinforcement Learning (ISRL)

To solve the problem of how soon the system can enter the desired state and consider the stability analysis, the ISRL is proposed. In the ISRL, the reinforcement signal is measured by two different strategies (judgment and evaluation strategies). The judgment strategy determines the reinforcement signal when the plant fails entering a predefined goal set, and the evaluation strategy applies under the condition that the plant enters the goal set. The key to the ISRL is using the Lyapunov-based manipulations on control laws to drive the plant to reach and remain in a predefined desired set of states with probability 1. Then, the time step for the plant entering the desired set of states can indicate the concept of how soon the system becomes stable. It will be observed that the advantage of the proposed ISRL is that it can meet global optimization capability.

As shown in simulation results, the proposed ISRL can not only work with different conditions of the system but also still controlled in the letter time steps. Moreover, the learning results are smoother than the traditional reinforcement signal design.

6. 2 Future Research

The goal of this dissertation is to provide a stable and robust way for applying evolutionary algorithm with neuro-fuzzy controller to control problems. This dissertation has developed the proposed ISRL-SAEAs to achieve this goal. Although the ISRL-SAEAs can obtain better performance than other methods, there still has a limitation of the ISRL-SAEAs. The initial parameters are determined by practical experimentation or trial-and-error tests. There is not a systematic method to determine the initial parameters. In the future work, how to find a well-defined method to define such parameters is an important work. Furthermore, in the simulation, the noise toleration can be further considered in the future. Therefore, the more robust method may be obtained.

Moreover, in the experiments in Chapters 5, a controller is successful if it can achieve goal sets during evolution. That is, the simulation environment used for training is also used for testing. However, in real-world applications, the environment used to test may be different with the environment used to train. The reason is that the environment used to test is time consuming. About this problem, for letting the evolutionary algorithms can be used in real-world applications, a controller transfer is needed to transfer the simulation environment to the environment used to test. How to design a controller transfer of the ISRL-SAEAs is also a future work of this dissertation.

Bibliography

[1] C. T. Lin and C. S. G. Lee, *Neuro-fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System*, NJ:Prentice-Hall, 1996.

[2] G. G. Towell and J. W. Shavlik, "Extracting refined rules from knowledge-based neural networks," *Mach. Learn.*, vol. 13, pp. 71-101, 1993.

[3] C. J. Lin and C. T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Trans. Fuzzy systs.*, vol. 5, no. 4, pp. 477-496, 1997.

[4] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1414-1427, 1992.

[5] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. 15, no. 1, pp. 116-132, 1985.

[6] C. F. Juang and C. T. Lin, "An on-line self-constructing neuro-fuzzy inference network and its applications," *IEEE Trans. Fuzzy Systs.*, vol. 6, no.1, pp. 12-31, 1998.

[7] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, and Cybern.*, vol. 23, no. 3, pp. 665-685, 1993.

[8] F. J. Lin, C. H. Lin, and P. H. Shen, "Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive," *IEEE Trans. Fuzzy Systs.*, vol. 9, no. 5, pp. 751-759, 2001.

[9] H. Takagi, N. Suzuki, T. Koda, and Y. Kojima, "Neural networks designed on approximate reasoning architecture and their application," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 752-759, 1992.

[10] E. Mizutani and J. S. R. Jang, "Coactive neuro-fuzzy modeling," *in Proc. IEEE Int. Conf.*

Neural Networks, Perth, WA , USA, vol. 2, pp. 760-765, November 27-December 1, 1995.

[11] C. J. Lin and C. C. Chin, "Prediction and identification using wavelet-based recurrent fuzzy neural networks," *IEEE Trans. Syst., Man, Cybern., Part B,* vol. 34, no. 5, pp. 2144-2154, 2004.

[12] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990.

[13] C. F. Juang and C. T. Lin, "A recurrent self-organizing neuro-fuzzy inference network," *IEEE Trans. Neural Networks*, vol. 10, no. 4, pp.828-845, 1999.

[14] P. A. Mastorocostas and J. B. Theocharis, "A recurrent fuzzy-neural model for dynamic system identification," *IEEE Trans. Syst., Man, Cybern.*, *Part B,* vol. 32, no. 2, pp. 176-190, 2002.

[15] X. Xu and H. G. He, "Residual-gradient-based neural reinforcement learning for the optimal control of an acrobat," in *Proc. IEEE Int. Symposium on Intelligent Control*, Vancouver, Canada, pp. 758-763, October 27-30, 2002.

[16] O. G.rigore, "Reinforcement learning neural network used in control of nonlinear systems," in *Proc. IEEE Int. Conf. Industrial Technology*, vol. 1, Goa, India, pp. 662-665, January 19-22, 2000.

[17] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuron like adaptive elements that can solve difficult learning control problem," *IEEE Trans. Syst., Man, Cybern.,* vol. 13, no 5, pp. 834-847, 1983.

[18] C. J. Lin, "A GA-based neural network with supervised and reinforcement learning," *Journal of the Chinese Institute of Electrical Engineering*, vol. 9, no. 1, pp. 11-25, 2002.

[19] X. W. Yan, Z.D. Deng, and Z.Q. Sun, "Competitive Takagi-Sugeno fuzzy reinforcement learning," in *Proc. IEEE Int. Conf. Control Applications*, Mexico City, Mexico, pp. 878-883, September 5-7, 2001.

[20] C. J. Lin and Y. C. Hsu, "Reinforcement hybrid evolutionary learning for recurrent

wavelet-based neuro-fuzzy systems," *IEEE Trans. on Fuzzy Systems*, vol. 15, no. 4, pp. 729-745, 2007.

[21] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 724-740, 1992.

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

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

[24] L. J. Fogel, "Evolutionary programming in perspective: The top-down view," in *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks II, and C. Goldberg, Eds. Piscataway, NJ: IEEE Press, 1994.

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

[26] C. L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," *in Proc. Int. Conf. Genetic Algorithms*, San Diego, CA, USA, pp. 450-457, July 1991.

[27] B. Carse, T. C. Fogarty, and A. Munro, "Evolving fuzzy rule based controllers using genetic algorithms," *Fuzzy Sets and Systems*, vol. 80, no. 3, pp. 273-293, 1996.

[28] C. T. Lin and C. P. Jou, "GA-based fuzzy reinforcement learning for control of a magnetic bearing system," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 30, no. 2, pp. 276-289, 2000.

[29] C. F. Juang, J. Y. Lin and C. T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans. Syst., Man, Cybern., Part B,* vol. 30, no. 2, pp. 290-302, 2000.

[30] S. Bandyopadhyay, C. A. Murthy, and S. K. Pal, "VGA-classifier: design and applications," *IEEE Trans. Syst., Man, and Cybern., Part B,* vol. 30, no. 6, pp. 890-895, 2000.

[31] D. Y. Wang, H. C. Chuang, Y. J. Xu, and C. J. Lin, "A novel evolution learning for recurrent wavelet-based neuro-fuzzy networks," in *Proc. IEEE Int. Conf. Fuzzy Systems*, Reno, NV, USA, pp. 1092-1097, May 25-25, 2005.

[32] T. J. Perkins, and A. G. Barto, "Lyapunov design for safe reinforcement learning," *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 803-832, 2003.

[33] C. K. Ting, C. N. Lee, H. C. Chang, and J. S. Wu, "Wireless heterogeneous transmitter placement using multiobjective variable-length genetic algorithm," *IEEE Trans. Syst., Man, Cybern., Part B,* vol. 39, no. 4, pp. 945-958, 2009.

[34] E. Saeidpour, V. S. Parizy, M. Abedi, and H. Rastegar, "Complete, integrated and simultaneously design for STATCOM fuzzy controller with variable length genetic algorithm for voltage profile improvement," in *Proc. Int. Conf. Harmonics and Quality of Power*, Wollongong, NSW, Australia, pp. 1-7, September 28-October 1, 2008.

[35] Y. C. Hsu and S. F. Lin, "Reinforcement self-adaptive evolutionary algorithm for fuzzy systems design," *in Proc. IEEE Int. Conf. Industrial Technology*, Chengdu, China, pp. 1-6, April 21-24, 2008.

[36] I. Saha, U. Maulik, and S. Bandyopadhyay, "A new differential evolution based fuzzy clustering for automatic cluster evolution," in *Proc. IEEE Int. Conf. Advance Computing*, Patiala, India, pp. 706-711, March 6-7, 2009.

[37] K. S. Tang, "Genetic algorithms in modeling and optimization," Ph. D. Dissertation, Dep. Electronic Engineering, City Univ. Hong Kong, Hong Kong, 1996.

[38] M. Ma and L. B. Zhang, "Optimizing a fuzzy neural network with a hierarchical genetic algorithm," *in Proc. IEEE Int. Conf. Machine Learning and Cybernetics*, Hong Kong, China, vol. 5, pp. 2812-2815, August 19-22, 2007.

[39] R. Kumar, K. Izui, Y. Masataka, and S. Nishiwaki, "Multilevel redundancy allocation optimization using hierarchical genetic algorithm," *IEEE Trans. Reliability,* vol. 57, no. 4, pp. 650-661, 2008.

[40] F. J. Gomez, “Robust non-linear control through neuroevolution,” Ph. D. Disseration, Dep. Computer Sciences, Univ. Texas of Austin, USA, 2003.

[41] F. Gomez and J. Schmidhuber, “Co-evolving recurrent neurons learn deep memory POMDPs,” *in Proc. Conf. Genetic and Evolutionary Computation*, Washington, DC, USA, pp. 491-498, June 25-29, 2005.

[42] J. Li and Z. J. Miao, “Entertainment oriented intelligent virtual environment with agent and neural networks,” in *Proc. IEEE Int. Workshop on Haptic, Audio and Visual Environments and Games*, Ottawa, Canada, pp. 90-95, October12-14, 2007.

[43] C. F. Juang, “Combination of online clustering and Q-value based GA for reinforcement fuzzy system design,” *IEEE Trans. Fuzzy Systs.,* vol. 13, no. 3, pp. 289-302, 2005.

[44] C. J. Lin and Y. J. Xu, “Efficient reinforcement learning through dynamical symbiotic evolution for TSK-type fuzzy controller design,” *International Journal of General Systems,* vol. 34, no.5, pp. 559-578, 2005.

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

[46] U. Fayyad, “Data mining and knowledge discovery in databases: implications for scientific database,” *in Proc. Int. Conf. Scientific and Statistical Database Management*, Olympia, Greece, pp. 2-11, August 11-13, 1997.

[47] M. Chen and Z. W. Yao, “Classification techniques of neural networks using improved genetic algorithms,” in *Proc. Int. Conf. Genetic and Evolutionary Computing*, Hubei, China, pp. 115-119, September 25-26, 2008.

[48] Q. L. Guo and M. Zhang, “A novel approach for fault diagnosis of steam turbine based on neural network and genetic algorithm,” in *Proc. IEEE Int. Conf. Neural Networks*, Hong Kong, China, pp. 25-29, June 1-8, 2008.

[49] R. Agrawal and R. Srikant, “Fast algorithm for mining association rules,” *in Proc. Int .Conf. Very Large Data Base*s, Santiago, Chile, pp. 487-499, September 12-15, 1994.

[50] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *in Proc. Int. Conf. Management of Data,* Dallas, Texas, USA, pp.1-12, May 16-18, 2000.

[51] T. P. Hong, C. S. Kuo, and S. C. Chi, "A data mining algorithm for transaction data with quantitative values," *Intelligent Data Analysis*, vol. 3, no. 5, pp. 363-376, 1999.

[52] Y. T. Wu, Y. J. An, J. Geller, and Y. T. Wu, "A data mining based genetic algorithm," in *Proc. IEEE Int. Workshop on Software Technologies for Future Embedded and Ubiquitous Systems and Collaborative Computing*, *Integration*, *and Assurance*, Gyeongju, Korea, pp. 6, April 27-28, 2006.

[53] K. G. Srinivasa, M. Jagadish, K. R. Venugopal, and L. M. Patnaik, "Data mining based query processing using rough sets and genetic algorithms," in *Proc. IEEE Symposium on Computational Intelligence and Data Mining*, Honolulu, USA, pp. 275-282, March 1-April 5 , 2007.

[54] S. P. Dai and P. Zhang, "A data mining algorithm in distance learning," in *Proc. IEEE Int. Conf. Computer Supported Cooperative Work in Design*, Xi'an, China, pp.1014-1017, April 16-18, 2008.

[55] C. J. Lin and Y. J. Xu, "The design of TSK-type fuzzy controllers using a new hybrid learning approach," *International Journal of Adaptive Control and Signal Processing*, vol. 20, no. 1, pp. 1-25, 2006.

[56] K. C. Cheok and N. K. Loh, "A ball-balancing demonstration of optimal and disturbance-accommodating control," *IEEE Contr. Syst. Mag.*, vol. 7, no. 1, pp. 54-57, 1987.

[57] D. Whitley, S. Dominic, R. Das, and C. W. Anderson, "Genetic reinforcement learning for neuro control problems," *Mach. Learn.*, vol. 13, pp. 259-284, 1993.

[58] A. Wieland, "Evolving neural network controllers for unstable systems," *in Proc. IEEE Int. Conf. Neural Networks*, Seattle, WA , USA, vol. 2, pp. 667-673, July 8-14, 1991.

[59] X. Xin and M. Kaneda, "Analysis of the energy-based control for swinging up two

pendulums," *IEEE Trans. Automatic Control,* vol. 50, no. 5, pp. 679-684, 2005.

[60] X. Xin and M. Kaneda, "Analysis of the energy based control for swinging up two pendulums," *in Proc. IEEE Int. Conf. Decision and Control*, Maui, Hawaii, USA, vol. 5, pp. 4688-4693, December 9-12, 2003.

[61] R. A. Howard, *Dynamic Programming and Markov Processes*, Cambridge, MA: MIT Press, 1960.

[62] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Cambridge, MA: MIT Press, 1998.

[63] C. J. C. H. Watkins, "Learning from delayed rewards," Ph. D. Dissertation, Univ. Cambridge, England, 1989.

[64] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279-292, 1992.

[65] G. A. Rummery and M. Niranjan, On-line Q-learning using connectionist systems, Technical Report CUED/F-INFENG/TR-166, Dep. Eng., Univ. Cambridge, 1994.

[66] L. J. Lin, "Reinforcement learning for robots using neural networks," Ph. D. Dissertation, Dep. Computer Sciences, Univ. Carnegie Mellon, Pittsburg, 1993.

[67] R. H. Crites and A. G. Barto, "Improving elevator performance using reinforcement learning," *Advances in Neural Information Processing System*, vol. 8, pp. 1017-1023, 1996.

[68] R. S. Sutton, "Generalization in reinforcement learning: successful examples using sparse coarse coding," *Advances in Neural Information Processing System*, vol. 8, pp. 1038-1044, 1996.

[69] H. K. Khalil, *Nonlinear systems*, NJ: Prentice-Hall, 2002.

[70] N. Chaiyaratana and A. M. S. Zalzala, "Recent developments in evolutionary and genetic algorithms: theory and applications," *in Proc. IEEE Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications*, Glasgow, United Kingdom, pp.

270-277, September 2-4, 1997.

[71] D. Wicker, M. M. Rizki, and L. A. Tamburino, "The multi-tiered tournament selection for evolutionary neural network synthesis," *in Proc. IEEE Int. Symposium on Combinations of Evolutionary Computation and Neural Networks*, San Antonio, USA, pp. 207-215, May 11-13, 2000.

[72] Y. P. Zou, Z. K. Mi, and M. H. Xu, "Dynamic load balancing based on roulette wheel selection," *in Proc. IEEE Int. Conf. Communications, Circuits and Systems*, Guilin, China, vol. 3, pp.1732-1734, June 25-28, 2006.

[73] P. M. Godley, D. E. Cairns, J. Cowie, and J. McCall, "Fitness directed intervention crossover approaches applied to bio-scheduling problems," *in Proc. IEEE Int. Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, Sun Valley, USA, pp. 120-127, September 15-17, 2008.

[74] S. Su and D. H. Zhan, "New genetic algorithm for the fixed charge transportation problem," *in Proc. IEEE Int. World Congress on Intelligent Control and Automation*, Dalian, China, vol. 2, pp. 7039-7043, June 21-23, 2002.

[75] W. Y. Wang, T. T. Lee, C. C. Hsu, and Y. H. Li, "GA-based learning of BMF fuzzy-neural network," *in Proc. IEEE Int. Conf. Fuzzy Systems*, Honolulu, USA, pp. 1234-1239, May 12-17, 2002.

[76] G. Lin and X. Yao, "Analysing crossover operators by search step size," *in Proc. IEEE Int. Conf. Evolutionary Computation*, Indianapolis, USA, pp. 107-110, April 13-16, 1997.

[77] C. P. Chen, S. P. Koh, I. B. Aris, F. Y. C. Albert, and S. K. Tiong, "Path optimization using genetic algorithm in laser scanning system," *in Proc. IEEE Int. Symposium on Information Technology*, Kuala Lumpur, Malaysia, vol. 3, pp. 1-5, August 26-28, 2008.

[78] C. J. Hsu, C. Y. Huang, and T. Y. Chen, "A modified genetic algorithm for parameter estimation of software reliability growth models," *in Proc. IEEE Int. Symposium on*

Software Reliability Engineering, Seattle, WA, USA, pp. 281-282, November 10-14, 2008.

[79] P. Luo, J. F. Teng, J. H. Guo, and Q. Li, "An improved genetic algorithm and its performance analysis," *in Proc. IEEE Int. Conf. Info-tech and Info-net*, Beijing, China, vol. 4, pp. 329-333, October 29-November 1, 2001.

[80] G. W. Greenwood, "Adapting mutations in genetic algorithms using gene flow principles," *in Proc. IEEE Congress on Evolutionary Computation*, Canberra, Australia vol. 2, pp.1392-1397, December 8-12, 2003.

[81] H. J. Lee, Y. S. Ma, and Y. R. Kwon, "Empirical evaluation of orthogonality of class mutation operators," *in Proc. IEEE Int. Conf. Software Engineering*, Busan, Korea, pp. 512-518, November 30-Decmber 3, 2004.

[82] N. S. Chaudhari, A. Purohit, and A. Tiwari, "A multiclass classifier using Genetic Programming," *in Proc. IEEE Int. Conf. Control, Automation, Robotics and Vision*, Hanoi, Vietnam, pp. 1884-1887, December 17-20, 2008.

[83] N. Gomez Bias, L. F. Mingo, and J. Castellanos, "Networks of evolutionary processors with a self-organizing learning," *in Proc. IEEE Int. Conf. Computer Systems and Applications*, Doha, Qatar, pp. 917-918, March 31-April 4, 2008.

[84] S. Abedi and R. Tafazolli, "Genetically modified multiuser detection for code division multiple access systems," *IEEE Journal on Selected Areas*, pp. 1884-1887, 2008.

[85] P. J. Darwen, "Co-evolutionary learning by automatic modularization with speciation," Ph. D. Dissertation, Univ. South Wales, 1996.

[86] J. B. Pollack, A. D. Blair, and M. Land, "Coevolution of a backgammon player," *in Proc. Int. Conf. Artificial Life: Synthesis and Simulation of Living Systems*, Nara, Japan, pp. 92-98, May 1996.

[87] G. Miller and D. Cliff, Co-evolution of pursuit and evasion i: Biological and game-theoretic foundations, Technical Report CSRP311, Dep. Cognitive and Computing

Sciences, Univ. Sussex, Brighton, UK, 1994.

[88] D. Grady, "The vision thing: mainly in the brain," *Discover*, vol. 14, pp.57-66, 1993.

[89] J. H. Holland, and J. S. Reitman, Cognitive systems based on adaptive algorithms, In Waterman, D. A., and Hayes-Roth, F., editors, *Pattern-Directed Inference Systems*. New York: Academic Press, 1978.

[90] R. Eriksson and B. Olsson, "Cooperative coevolution in inventory control optimization," in *Proc. Int. Conf. Artificial Neural Networks and Genetic Algorithms,* Norwich, UK, pp. 583-587, April 1997.

[91] J. Horn, D. E. Goldberg, and K. Deb, "Implicit niching in a learning classifier system: nature's way," *Evolutionary Computation*, vol. 2, no. 1, pp. 37-66, 1994.

[92] J. Paredis, "Coevolutionary computation," *Artificial Life*, vol. 2, no. 4, pp. 355-375, 1995.

[93] B. A. Whitehead and T. D. Choate, "Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction," *IEEE Trans. Neural Networks*, vol. 7, no. 4, pp. 869-880, 1996.

[94] M. A. Potter and K. A. De Jong, "Evolving neural networks with collaborative species," in *Proc. Conf. Summer Computer Simulation*, Ottawa, Ontario, Canada , pp. 1-7, July 24-26, 1995.

[95] D. E. Moriarty, "Symbiotic evolution of neural networks in sequential decision tasks," Ph. D. Dissertation, Dep. of Computer Sciences, Univ. Texas at Austin, USA, Technical Report UT-AI97-257, 1997.

[96] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Mach. Learn.*, vol. 22, pp. 11-32, 1996.

[97] R. E. Smith, S. Forrest, and A. S. Perelson, "Searching for diverse, cooperative populations with genetic algorithms," *Evol. Comput.*, vol. 1, no. 2, pp. 127-149, 1993.

[98] Z. Michalewicz, *Genetic Algorithms+Data Structures=Evolution Programs*, New York:

Springer-Verlag, 1999.

[99] R. Tanese, “Distributed genetic algorithm,” *in Proc. Int. Conf. Genetic Algorithms*, San Mateo, USA, pp. 434-439, December 1989.

[100] J. Arabas, Z. Michalewicz, and J. Mulawka, “GAVaPS—A genetic algorithm with varying population size,” *in Proc. IEEE Int. Conf. Evolutionary Computation*, Orlando, USA, pp. 73-78, June 27-29, 1994.

[101] G. R. Harik, F. G. Lobo and D. E. Goldberg, **“**The compact genetic algorithm,” *IEEE Trans. Evolutionary Computation*, vol. 3, no. 4, pp. 287-297, 1999.

[102] K. Y. Lee, X. M. Bai, and Y. M. Park, “Optimization method for reactive power planning by using a modified simple genetic algorithm,” *IEEE Trans. Power Systems,* vol. 10, no. 4, pp. 1843-1850, 1995.

[103] K. A. De Jong, “Analysis of the behavior of a class of genetic adaptive systems,” Ph. D. Dissertation, Dep. Computer and Communication Sciences, Univ. Michigan, Ann Arbor, MI, 1975.

[104] J. J. Grefenstette, “Optimization of control parameters for genetic algorithms,” *IEEE Trans*. *Syst*., *Man*, *Cybern*., vo1. 6, no. 1, pp. 122-128, 1986.

Vita

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 and the M.S. degree in computer science and information engineering from Chaoyang University of Technology, Taiwan, R.O.C. He proposed the Ph. D. oral exam at department of electrical and control engineering from the National Chiao Tung University, Taiwan, R.O.C. in June, 2009. He is a member of the Phi Tau Phi. He is also a member of the Taiwanese Association for Artificial Intelligence (TAAI). His research interests include neural networks, fuzzy systems, and genetic algorithms.

Publication List

Accepted Journal Papers:

1. Yung-Chi Hsu and Sheng-Fuu Lin, "Reinforcement Group Cooperation based Symbiotic Evolution for Recurrent Wavelet-Based Neuro-Fuzzy Systems," *Neurocomputing*, vol. 72, no. 10-12, pp. 2418-2432, 2009. (SCI)

2. Yung-Chi Hsu and Sheng-Fuu Lin, "Self-Organization Hybrid Evolution Learning Algorithm for Recurrent Wavelet-based Neuro-Fuzzy Identifier Design," accepted to appear in *Journal of Intelligent and Fuzzy Systems*, 2009. (SCI)

Submitted Journal Papers:

1. Yi-Chang Cheng, Sheng-Fuu Lin, and Yung-Chi Hsu, "Two-Strategy Reinforcement Hybrid Evolutionary Learning for Recurrent Wavelet-Based Neuro-Fuzzy Systems," revised in *International Journal of Adaptive Control and Signal Processing*. (SCI)

2. Yung-Chi Hsu, Sheng-Fuu Lin, and Yi-Chang Cheng, "Two-Strategy Reinforcement Evolutionary Algorithm using Data-mining based Crossover Strategy with TSK-type Fuzzy Controllers," revised in *International Journal of Innovative Computing, Information and Control*. (SCI)

3. Yung-Chi Hsu and Sheng-Fuu Lin, "Reinforcement Evolutionary Learning using Data Mining Algorithm with TSK-type Fuzzy Controllers," submitted to *Applied Soft Computing*. (SCI)

4. Yung-Chi Hsu and Sheng-Fuu Lin, "Data-Mining based Evolutionary Learning Algorithm with TSK-type Neuro-Fuzzy Systems," submitted to *International Journal of Control, Automation, and Systems*. (SCI)

5. Yung-Chi Hsu, Sheng-Fuu Lin, and Yi-Chang Cheng, "Multi Groups Cooperation based Symbiotic Evolution for TSK-type Neuro-Fuzzy Systems Design," submitted to *Expert Systems with Applications*. (SCI)

6. Yung-Chi Hsu, Sheng-Fuu Lin, and Yi-Chang Cheng, "Reinforcement Self

Adaptive Multi Groups based Symbiotic Evolutionary Algorithm for TSK-type Fuzzy Model Design," submitted to *IEICE Trans. Inf. & Syst.*. (SCI)

7. Yung-Chi Hsu and Sheng-Fuu Lin, "Reinforcement Self Adaptive Evolutionary Learning using FP-growth Algorithm with TSK-type Neuro-Fuzzy Systems," submitted to *Soft Computing*. (SCI)

8. Yung-Chi Hsu, Sheng-Fuu Lin, Yi-Chang Cheng, and Jyun-Wei Chang, "A Self-Organization Mining Based Hybrid Evolution Learning for TSK-type Fuzzy Model Design," submitted to *International Journal of General Systems*. (SCI)

9. Yung-Chi Hsu, Sheng-Fuu Lin, Yi-Chang Cheng, and Jyun-Wei Chang, "Two-Strategy Reinforcement Group Cooperation based Symbiotic Evolution for TSK-type Fuzzy Controller Design," submitted to *Journal of Multiple-valued Logic and Soft Computing*. (SCI)

Conference Papers:

1. Sheng-Fuu Lin, Yi-Chang Cheng, Jyun-Wei Chang, and Yung-Chi Hsu, "Reinforcement Learning for Data Mining Based Evolution Algorithm for TSK-type Neural Fuzzy Systems Design," *Int. Conf. Information Systems Analysis and Synthesis*, Orlando, Florida, USA, July 10-13, 2009.

2. Yung-Chi Hsu and Sheng-Fuu Lin," Reinforcement Hybrid Evolutionary Learning for TSK-type Neuro-Fuzzy Controller Design," *17th IFAC World Congress (IFAC WC 2008)*, Seoul, Korea, July 6-11, 2008.

3. Yung-Chi Hsu and Sheng-Fuu Lin, "Reinforcement Self-Adaptive Evolutionary Algorithm for Fuzzy Systems Design," *IEEE Int. Conf. on Industrial Technology (IEEE ICIT2008)*, Sichuan University, Chengdu, China, April 21-24, 2008.

4. Yung-Chi Hsu and Sheng-Fuu Lin, "TSK-Type Fuzzy Systems with a Novel Symbiotic Evolution for Face Detection System," *The 11th Conference on Artificial Intelligence and Applications*, Kaohsiung, Taiwan, December15-16, 2006.