國 立 交 通 大 學

電控工程研究所

博士論文

合作式學習為基礎之混合型進化演算法在模糊類神經系統設計及多漏斗函數最佳化的應用

Cooperative Learning Based Hybrid Evolutionary Algorithms for Neural Fuzzy System Design and Optimization of Multi-funnel Functions

研究生：鄭逸章

指導教授：林昇甫 博士

中華民國一○一年六月

合作式學習為基礎之混合型進化演算法在模糊類神經系統設計及多漏斗函數最佳化的應用

Cooperative Learning Based Hybrid Evolutionary Algorithms for Neural Fuzzy System Design and Optimization of Multi-funnel Functions

研　究　生：鄭逸章　　　Student: Yi-Chang Cheng
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 Institute of Electrical Control Engineering
National Chiao Tung University
In Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy
in
Electrical and Control Engineering
June 2012
Hsinchu, Taiwan, Republic of China

中華民國一○一年六月

合作式學習為基礎之混合型進化演算法在模糊類神經系統設計及多漏斗函數最佳化的應用

研究生：鄭逸章　　　　　　　　　指導教授：林昇甫 博士

國立交通大學 電控工程研究所

摘　要

在這篇論文中，我們主要在探討的是進化型演算法的合作學習機制。本論文中所探討的三種進化型演算法包括：基因演算法、粒子群聚最佳化演算法、以及自適應共變異數矩陣演化策略。在基因演算法的改良上，我們提出了族群式的共生演化概念，使得基因演算法可以將解空間分割成數個子空間，且在每個子空間中分別得去探索最佳解。我們也在合作式的粒子群聚演算法中提出了一個可分割度的偵測方法，以便將不可分割之變數置入同一族群中演化。至於關於自適應共變異數矩陣演化策略的改良，本論文提出了一個基於均值移動的平行運算機制，使得我們可以平行地在解空間中提供多個自適應共變異數矩陣演化策略學習器來探索解空間中的不同區域。論文的內容包括了將進化型演算法套用在模糊類神經系統上之架構和參數學習、演算法上的改良、平行運算機制以及結合兩種演算法優點的混合型演算法的研究。

關鍵字：合作式學習，基因演算法，粒子群聚最佳化演算法，演化策略，自適應共變異數矩陣。

Cooperative Learning Based Hybrid Evolutionary Algorithms for Neural Fuzzy System Design and Optimization of Multi-funnel Functions

Student：Yi-Chang Cheng Advisor：Dr. Sheng-Fuu Lin

Institute of Electrical Control Engineering
National Chiao Tung University

Abstract

In this dissertation, we mainly focus on researching the cooperative behavior of evolutionary algorithms. Algorithms discussed in this dissertation include genetic algorithm (GA), particle swarm optimization (PSO) and evolution strategy with covariance matrix adaptation (CMA-ES). The modification of genetic algorithm (GA) is done by introducing the group-based symbiotic evolution (GSE) technique which enables genetic algorithm (GA) to partition the search space into smaller subspaces and explore each smaller subspace by a separate agent to alleviate the curse of dimensionality. We also propose a separability detection method based on covariance matrix adaption mechanism into the cooperative particle swarm optimization (CPSO) to locate non-separable variables into the same swarm. As to the research of evolution strategy with covariance matrix adaptation (CMA-ES), we introduce the mean shift procedure which allows us to apply multiple CMA-ES instances to explore different parts of the search space in parallel. The scope of this dissertation includes how to implement evolutionary algorithms on neural-fuzzy systems, the improvement of algorithms, parallel computing and the emergence of two algorithms

Keywords: cooperative learning, genetic algorithm, particle swarm optimization, evolution strategy, covariance matrix adaptation.

誌謝

經過六年的奮戰，博士生涯終告一段落，這六年，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，感謝父親鄭文昌先生、母親羅鳳美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。弟弟鄭彥章先生，感謝你的體諒；如此漫長的求學，父親母親都靠你們照顧，辛苦了。

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、林錫寬教授、張翔教授以及鍾鴻源教授，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。

研究室裡互相扶持幫忙的好友太多了，謝謝你們。特別感謝實驗室博士班學長永吉，同學啟曜及俊偉，有你們的並肩作戰，博士生涯才不致如此漫長。另外也特別感謝碩士班學弟煒清，你是一個很好的學習夥伴，感謝你的協助，此博士論文才能順利完成，辛苦了。

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

鄭逸章

一○一年　　六月　　五日

Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Evolutionary algorithms [1]-[6] are stochastic, population-based optimization learning algorithms that can be applied to a wide range of problems. Generally speaking, there is no clear rank between different evolutionary algorithms. We can only say that certain algorithm is more applicable than others to certain optimization problems. In this dissertation, we apply the modified version of genetic algorithm (GA) [7] and particle swarm optimization (PSO) [8]-[10] to high-dimensional, reinforcement learning tasks, and apply evolution strategy with covariance matrix adaptation (CMA-ES) [11]-[13] to complex, low-dimensional real-valued function optimization tasks.

1.1 Motivation

Evolutionary algorithms are discovered through simulating some social behavior, such as the bird flocking, the recombination or the mutation of genes. Normally, evolutionary algorithms maintain a population of potential solutions to some optimization problem, generating new solutions at each iteration by using a variety of corresponding operators. Their learning procedures take place in populations made of individuals with specific behaviors similar to certain biological phenomena. Individuals keep exploring the solution space and exploiting information between individuals while evolution proceeding. In general, by means of exploring and exploiting, evolutionary algorithms are less likely to be trapped at the local optimum.

Evolutionary algorithms are applicable to a wide range of problems, including training neural-fuzzy systems (NFS) [14]-[17], reinforcement learning control [18]-[22] and complex, multi-funnel [23]-[26] function optimization tasks. However, as with GA, PSO and CMA-ES, nearly every other kind of stochastic optimization algorithms suffer from the "curse of dimensionality," which simply put, implies that their performance deteriorates as the dimensionality of the search space increases. One way to overcome this difficulty is to partition the search space into lower dimensional subspaces, as long as the optimization algorithm can guarantee that it will be able to search every possible region of the search space. Van den Bergh and Engelbrecht suggested that the search space should be partitioned by splitting the solution vectors into smaller vectors and proposed a cooperative approach to particle swarm optimization (CPSO) [27]-[28]. Each of these smaller search spaces is then searched by a separate PSO instance; the fitness function is evaluated by combining solutions found by each swarm of the PSO instance. In this dissertation, we introduce the cooperative learning behavior to GA and proposed the groups-based symbiotic evolution (GSE) [29]. The proposed GSE is applied to training a NFS. It is different from traditional symbiotic evolution where each population in the GSE is divided to several groups and each group represents a set of chromosomes that belongs to one fuzzy rule. The fitness value of each fuzzy rule can be evaluated locally. However, separating the search space also arouses two issues.

The first issue is the possibility that the partitioning could lead to the introduction of pseudooptimum, which means that the combination of optima found by each learning instance may not be an actual optimum point to the original search space, may not even be a local optimum point. In [27], Van den Bergh and Engelbrecht proposed a variation of CPSO called the CPSO-H_K to alleviate the issue of pseudooptimum. The CPSO is one of the most significant improvements to the standard PSO. Algorithm CPSO-H_K is a hybrid from the standard PSO and the CPSO-SK model. It prevents the solution found so far from becoming a pseudooptimum by executing the CPSO-S_K algorithm for one iteration, followed by one iteration of the PSO algorithm. Computer simulations in [27] have shown that the CPSO-H_K indeed alleviates the issue of pseudooptimum. However, as with other cooperative learning algorithms [30, 31], the performance of the CPSO deteriorates when there exists dependence among parameters.

The second issue aroused by partitioning the search space is that performances of the cooperative learning algorithms deteriorate when correlated variables are placed into separate populations. In this dissertation, we call such variables "non-separable." A function *f* is said to be separable if

$$\arg\min_{(x_1,\cdots,x_n)} f(x_1,\cdots,x_n) = (\arg\min_{x_1} f(x_1,\cdots),\cdots,\arg\min_{x_n} f(\cdots,x_n)), \tag{1.1}$$

and it is followed by a fact that f can be optimized in a sequence of n independent 1-D optimization processes. In this dissertation, we propose a separability detection approach based on covariance matrix adaptation to find non-separable variables so that they can previously be placed into the same swarm to address the difficulty that the original CPSO encounters. This proposed variation on the original CPSO to detect the separability of the variables is called the SD-CPSO [106]. The SD-CPSO helps the CPSO self-organize the swarms composed of non-separable variables. In order to implement this idea, we have to determine the timing of switching between the PSO and the CPSO operation when dealing with a task. In this dissertation, we think this can be done by determining the separability between variables, and placing non-separable into the same swarm at each generation. If at certain moment, all variables are determined as non-separable, then the PSO operation is taken; otherwise, the CPSO operation is taken. The separability between variables is found by estimating the covariance matrix of the distribution of particles. The mechanism we adopt is the covariance matrix adaptation proposed from evolution strategy with covariance matrix adaption (CMA-ES) [11]-[13]. Conventionally, there exists a contradiction between the local search performance and the global exploration power of a learning algorithm [32]. For example, the GA and PSO are noted for their great global exploration power; whereas, due to the adaptivity of the local search, the CMA-ES owns an outstanding local search performance. In this dissertation, we apply the GA and PSO to high-dimensional, reinforcement learning tasks, and apply the CMA-ES to complex, mid-dimensional real-valued function optimization tasks.

1.2 Related Works

In this dissertation, we basically encode parameters on a NFS into individuals of GA or PSO to perform reinforcement learning control, and apply a parallel learning structured version of CMA-ES to complex, multi-funnel function optimization. As a result, we will discuss these two types of optimization tasks in this section. The discussion of reinforcement learning will be shown in section 1.2.1 and the discussion of multi-funnel function optimization will be shown in section 1.2.2.

1.2.1 Reinforcement Learning Tasks

In recent years, the application of NFS in control engineering has become a popular research topic [33]-[43]. In general, the way of tuning the parameters on a NFS can be divided into two categories: supervised learning [44] and reinforcement learning [18].

Supervised learning is a machine learning technique for updating its parameters from training data. The training data is composed of pairs of inputs, and desired outputs. The object of the supervised learning is to predict the output value of the NFS for any valid input data after its parameters have been trained by a number of training data. However, for many control tasks, training data are usually difficult or too costly, or even not accessible. As a result, reinforcement learning is more practicable than supervised learning in many occasions.

In reinforcement learning, the agent receives from its environment a reinforcement signal at each time step. This signal could be either a reward or a punishment. Meanwhile, the agent explores actions from the action set, and finds out which action yields the greatest reward. To solve reinforcement problem, temporal difference (TD) [19]-[21] is one of the most common method. In TD learning, learners don't have to wait until the end of a trial; instead, TD methods need wait only one time step. This is crucial for applications that have very long trials or tasks that are continuous and have no trials at all. Q-learning [22] is a powerful and easy-implementing TD-based approach. It is a reinforcement learning technique that works by updating a simple action-value iteration function. This function gives the measurement of taking a given action in a given state.

Besides TD methods, many evolutionary algorithms such as PSO, GA, evolutionary programming [45], and evolution strategies [46] are popular for solving reinforcement learning tasks. These learning procedures are based on populations made of individuals with specific behaviors similar to certain biological phenomena. Individuals keep exploring the solution space and exploiting information between individuals while evolution proceeding. In general, by means of exploring and exploiting, evolutionary algorithms are less likely to be trapped at the local optimum. Many researches on using evolutionary algorithm for solving reinforcement learning tasks have been proposed recently [47]-[50]. In [49], authors propose a swarm intelligence based reinforcement learning (SWIRL) method to train artificial neural networks (ANN). Authors apply ant colony optimization to select ANN topology and apply the PSO to adjust ANN connection weights. In [50], Lin and Hsu present a reinforcement hybrid learning algorithm (R-HELA) combining the compact GA (CGA) [51] and the modified variable-length GA (VGA) [52] on recurrent wavelet-based NFS. A counter is used to accumulate the time steps until the control task fails and the accumulated values are fed into individuals as fitness functions. Lin and Hsu's model is very effective; however, its fitness function only indicates how long can the controller work well instead of measuring how soon the system can meet the control goal, which is also very important in reinforcement learning. There is also a growing interest in combining the advantages of evolutionary algorithms and TD-based reinforcement learning [53]-[54]. In [53], a TD and GA based reinforcement learning (TDGAR) is proposed. Authors propose a neural structure composed of two feedforward networks for reinforcement learning, the critic network and the action network. The critic network predicts the external signal provides a more informative internal signal to the action network. The action network uses GA to determine the output of the learning system. The weight update rule for the hidden layer of the critic network is based on error backpropagation. In [54], an on-line clustering and Q-value based GA reinforcement learning for fuzzy system (CQGAF) is proposed. In one generation CQGAF learning, one individual is applied to the environment to estimate the fitness function, Q-value, and Q-values of other individuals are updated by eligibility trace. The GA operation is performed by the end of each trial and creates a new generation of individuals. In [55], authors proposed a recurrent wavelet-based NFS with a reinforcement group cooperation-based symbiotic evolution (R-GCSE) algorithm. In [55], a population is divided to several groups. The R-GCSE has a good ability of parameter learning by adopting the concept that each group formed by a set of chromosomes cooperates with other groups to generate better chromosomes.

Although the aforementioned reinforcement learning methods work well in many applications, there is an issue remains to be solved. No fitness function in these methods indicates how soon the learning agents can control the system's state into a set of goal states. Sure there is no need to define the fitness function that way if there is no guidance provided to the controller of how to maintain the system's state in a desired operating range. As a result, in this dissertation, we proposed a Q-value based particle swarm optimization (QPSO) [56] which adopts the concept of Lyapunov design [57] for constructing safe reinforcement learning agents, and a GA based learning method called two-strategy reinforcement evolutionary algorithm (TSR-EA) [29] to solve reinforcement learning tasks. In both algorithms proposed in this dissertation, we manipulate our fitness function so that it can indicate how soon the controller achieves its control goal.

1.2.2 Multi-funnel Function Optimization Functions

Normally, continuous function optimization problems are categorized into convex (unimodal) and non-convex (multimodal) functions. In this dissertation, we classified optimization problems into single-funnel and multi-funnel problems and we mainly focus on the optimization of multi-funnel functions. The difference between single- and multi-funnel functions can be illustrated by the following two figures, where Fig. 1.1 shows a visualization of a 2-D Rastrigin's function, from which we can see that in spite of the large amount of local minima, there exists a trend to the global minimum. Figure 1.2 shows a visualization of a 2-D double Rastrigin's function, from which we can see that there are two funnel-type global trends and a large amount of noisy local minima.

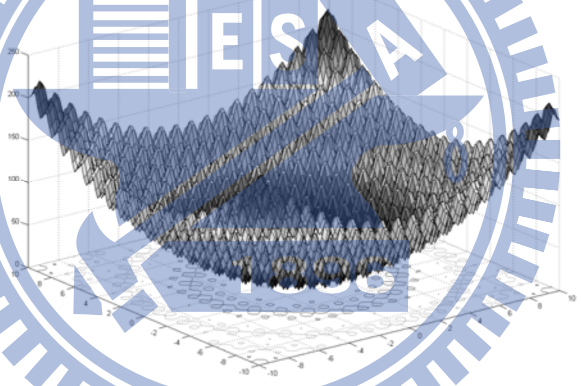


Figure 1.1: Visualization of a single-funnel, 2-D Rastrigin's function.

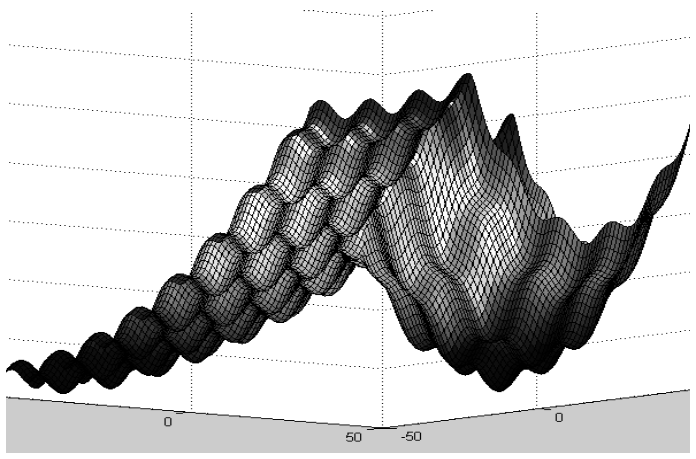


Figure 1.2: Visualization of a multi-funnel-funnel, 2-D double Rastrigin's function.

A function is said to be single-funnel, even though it is highly multi-modal, if its local optima is structured such that there exists a global trend toward the best solution [58]. However, there are several real-world applications do not have this simple structure. Many optimization problems are characterized by their local optima distributing in separate clusters within the search space and there is no underlying convex topology toward their global optima. Problems of this type are referred to as multi-funnel functions [23]. Prominent examples for such applications include potential energy surfaces of biomolecules [24] and protein aggregation and misfolding [25]. It has been suggested that the global topology of a problem may have a strong influence on the performance of optimization of multi-funnel functions [26]. To this end, we introduce the mean shift procedure [59] into the evolution strategy with covariance matrix adaptation (CMA-ES) which allows us to apply multiple CMA-ES instances to explore different parts of the search space in parallel.

The CMA-ES has been proven to be among the most successful optimization algorithms for optimization of non-convex functions. During exploring of the search space, the CMA-ES generates a population of samples from a multivariate Gaussian distribution. The mean and covariance matrix of the sampling distribution are continuously adapted in order to improve the search direction and the sampling distribution. Recent improvements include a local restart CMA-ES (LR-CMA-ES) [60], which greatly prevents CMA-ES from being trapped into local optima, and a CMA-ES with iteratively increasing population size (IPOP-CMA-ES) [61], achieving excellent performance on non-convex, high-dimensional optimization problems. However, Hansen and Kern [62] have pointed out that on multi-funnel functions, where local optima cannot be interpreted as perturbations to an underlying convex (unimodal) topology, performance can strongly be limited. This could be due to the fact that CMA-ES was originally proposed as a local search strategy, whereas the concept of multi-funnel functions is intrinsically based on global information. As a result, some researches [26, 63] combine the evolution strategy with global optimization schemes to increase its global exploration power. In [63], a particle swarm guided evolution strategy (PSGES) is proposed. Computer simulation results have shown that PSGES improves the original evolution strategy but is inferior to LR-CMA-ES and IPOP-CMA-ES. This could be due to a lack of local adaptivity mechanism. In [26], authors propose a particle swarm CMA-ES (PS-CMA-ES), which combines the local search performance of the CMA-ES with the global exploration power of the PSO. Computer simulation results in [26] have shown that no salient improvement over LR-CMA-ES and IPOP-CMA-ES on optimization of unimodal, basic multimodal functions, but improvement can be seen in optimization of high-dimensional multi-funnel functions. However, in spite of the great performance the PS-CMA-ES achieves, this methodology tends to be computationally expensive and the criterion of how frequent the PSO updates can be performed is not straightforward. In this dissertation, our objective is to improve performance on multi-funnel problems on one hand, and on-line determine the number of the CMA-ES instances on the other. Instead of directly combing the CMA-ES with certain "global" evolutionary algorithm, we introduce a computational module based on mean shift procedure into the CMA-ES. Mean shift procedure is a density estimation-based, non-parametric mode detection and clustering approach toward feature space analysis [59, 64, 65]. It determines the number of modes in a unknown probability density function (p.d.f.), and the density estimation is completed by kernel density estimator [66].

First, we apply kernel density estimation to the candidate solutions sampled by the CMA-ES. Then, we use the mean shift-based mode detection to determine the number of CMA-ES instances for exploring the search space simultaneously. In cases of more than one CMA-ES instances are applied, the proposed mean shift based evolution strategy with covariance matrix adaption (MS-CMA-ES) samples a population of candidate solutions from a mixture model of Gaussian distribution [67]. The covariance matrix of the mixture Gaussian sampling distribution is formed by the linear combination of the covariance matrixes of separate CMA-ES instances. Enforcing a mixture model provides a communication between different CMA-ES instances such that the CMA-ES instances with better search locations can sample more offspring, while the CMA-ES instances trapped in local optima can fade out. Another advantage of the proposed MS-CMA-ES is that there is no requirement of the criterion for the fusion (or division) of the CMA-ES instances, nor does the predefinition of the number of CMA-ES instances as a parameter. The bandwidth of the kernel density estimator can also be computed through kernel smoothing [68]. The only extra parameter besides the original parameters of the CMA-ES is the learning rate of mixture weightings for mixture Gaussian components, which reduces challenges of applying our methodology.

In this dissertation, we compare the proposed MS-CMA-ES with the standard CMA-ES [13], some of its improvements [60, 61], and some hybrid algorithms that combine the evolution strategy with the PSO [26, 63]. Computer simulation results will show that the MS-CMA-ES has better performance in optimizing multi-funnel functions.

1.3 Approach

In this dissertation, four major algorithms are proposed. The first two algorithms are called Q-value based particle swarm optimization (QPSO) and two-strategy reinforcement evolutionary algorithm (TSR-EA) respectively. These two algorithms are both proposed to solve reinforcement learning tasks. The advantages of the QPSO can be shown from that it provides an alternative for Q-learning to solve reinforcement learning problem in one hand, and it extends the applicability of the PSO into reinforcement environment on the other. It also provides a reliable initial learning performance due to the Lyapunov design of learning agents. But one drawback of the QPSO is that it requires additional priori knowledge. The main advantages of the TSR-EA can be summarized as follows: 1) the proposed TSR mechanism enables us to evaluate a learning trial for both how long can the controller work under operating range instead of measuring how soon the system meet the control goal; 2) the

GSE is proposed to evaluate the fuzzy rule locally. However, in the TSR-EA, divide parameters corresponding to different fuzzy rule into separate groups is very straightforward. However, in the optimization task, if the correlation between parameters is unknown, placing uncorrelated parameters into a same group would be a challenge.

The third algorithm proposed in this dissertation is a separability approach to cooperative particle swarm optimization (SD-CPSO), and it is mainly proposed to help placing uncorrelated variables into a same swarm. The proposed separability detection approach is based on the CMA-ES.

The fourth algorithm proposed in this dissertation is the mean shift-based evolution strategy with covariance matrix adaptation (MS-CMA-ES). The introduced mean shift procedure provides functions of mode detection and clustering which allows us to apply multiple CMA-ES instances to explore different parts of the search space in parallel. The global exploration power of the standard CMA-ES is enhanced by the concept that each instance forms a separate CMA-ES agent to explore different parts of the search space. We evaluate the performance of the MS-CMA-ES on the optimization of multi-funnel functions and the new MS-CMA-ES algorithm shows superior performance on it.

1.4 Organization of Dissertation

The dissertation is arranged as follows.

Chapter 1 introduces the motivation, related work, approach, and organization of the dissertation.

Chapter 2 provides the fundamental information used in the dissertation. The foundation includes neural fuzzy network, genetic algorithm, standard and cooperative PSO, mean shift procedure and CMA-ES.

In Chapter 3, the proposed QPSO, TSR-EA, SD-CPSO and MS-CMA-ES are described.

In Chapter 4, two types of computer simulations, reinforcement learning control tasks and multi-funnel optimization functions, are performed to verify the performance of the proposed algorithms. We apply QPSO and TSR-EA to two reinforcement learning control tasks, cart-pole balancing system and two-pole inverted pendulum control. The SD-CPSO and MS-CMA-ES are applied to real-valued function optimization tasks.

In Chapter 5, the conclusions, contribution, and future works of the dissertation 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, genetic algorithm, particle swarm optimization, and evolution strategy with covariance matrix adaptation) are introduced in this chapter. The concept of neuro-fuzzy controller is discussed in the first section. The concept of genetic algorithm (GA) is introduced in Section 2.2. In Section 2.3, the concept of particle swarm optimization (PSO) and some of its improvements are discussed. The final section focuses on some background knowledge related to the proposed mean shift-based evolution strategy with covariance matrix adaptation (MS-CMA-ES), such as kernel density estimation, mean shift procedure and standard CMA-ES

.

2.1 Neural Fuzzy System

In general, there are three typical types of neural-fuzzy system (NFS) and they are the TSK-type [34], Mamdani-type [16], and singleton-type. According to [69] and [70], the authors have shown that the TSK-type NFS can offer better network size and learning accuracy than the Mamdani-type and singleton-type NFS. Thus, in this dissertation, only the TSK-type NFS is introduced and such NFS is applied to reinforcement learning tasks.

A TSK-type NFS employs different implication and aggregation methods from a standard Mamdani fuzzy model. Instead of using fuzzy sets, the conclusion part of a rule is a linear combination of the crisp inputs.

$$\begin{gathered} IF\ x_1\ is\ A_{1j}\,(m_{1j}\,,\ \sigma_{1j}\,)and\ x_2\ is\ A_{2j}(m_{2j}\,,\ \sigma_{2j}\,)\ldots and\ x_n\ is\ A_{nj}\,(m_{nj}\,,\ \sigma_{nj}\,) \\ THEN\ y'=w_{0j}+w_{1j}x_1+\ldots+w_{nj}x_{n.} \end{gathered} \tag{2.1}$$

The structure of a TSK-type NFS is shown in Fig. 2.1. It is a five-layer network structure. In a TSK-type NFS, the firing strength of a fuzzy rule is calculated by performing the following "AND" operation on the truth values of each variable to its corresponding fuzzy sets. The functions of the nodes in each layer are described as follows:

Layer 1 (input node): Each node in this layer is called an input linguistic node, which corresponding one linguistic variable. These nodes only pass the input signal to the next layer.

$$u_i^{(1)} = x_i, \tag{2.2}$$

where $u_i^{(1)}$ denotes the *i*th node's input in the 1st layer and x_i denotes *i*th input dimension.

Layer 2 (membership function node): each node in this layer acts as a Gaussian membership function, and its output value specifies the degree to which the given input value belongs to a fuzzy set. Thus, the membership value in layer 2 can be calculated by:

$$u_{ij}^{(2)} = \exp\left(-\frac{\left[u_i^{(1)} - m_{ij}\right]^2}{\sigma_{ij}^2} \right), \tag{2.3}$$

where $u_i^{(1)} = x_i$ and $u_{ij}^{(2)}$ are the outputs of 1st and 2nd layers ; m_{ij} and σ_{ij} are the center and the width of the Gaussian membership function of the *j*th term of the *i*th input variable x_i respectively. In this paper, the reason of adopting the Gaussian membership function is that it can be a universal approximator of any nonlinear functions on a compact set [69].

Layer 3 (rule node): The output in this layer are used to perform precondition matching of fuzzy rules. In the TSK-type NFS, the firing strength of a fuzzy rule is calculated by performing the following "AND" operation:

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} . \tag{2.4}$$

Layer 4 (consequent node): each node in this layer calculates the consequent value. Each consequent value (linear combination of the crisp inputs) is weighted by the firing strength of the fuzzy rule and it can be written by:

$$u_j^{(4)} = u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i), \tag{2.5}$$

where the summation is the consequent part and w_{ij} is its corresponding parameters.

Layer 5 (output node): The node in this layer computes output signal. The output node integrates with links connected to it and acts as a defuzzifier with:

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{M} u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i)}{\sum_{j=1}^{R} u_j^{(3)}}, \tag{2.6}$$

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

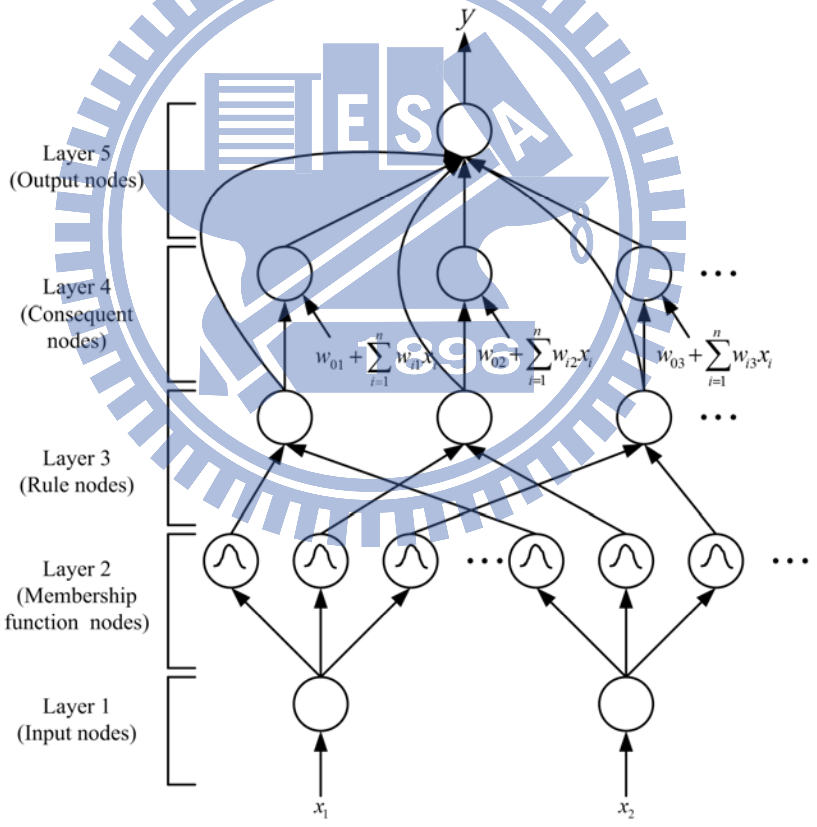


Figure 2.1: Structure of TSK-type NFS.

2.2 Genetic Algorithm

Genetic algorithms (GAs) 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. 2.2, 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 [71]-[73] 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 [73] (see Fig.2.3). In Fig.2.3, 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.

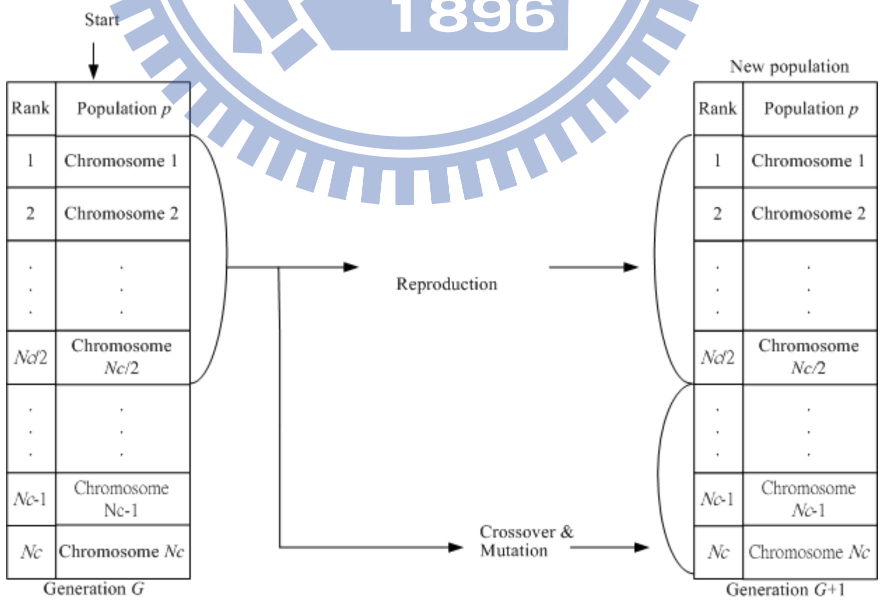


Figure 2.2: Flowchart of the genetic algorithm.

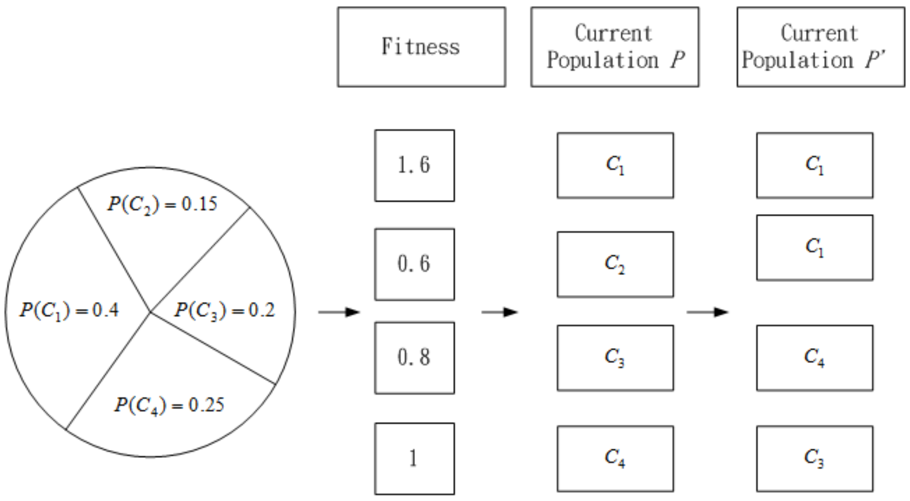


Figure 2.3: The roulette wheel selection.

In crossover step [74]-[78], 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 2.4 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.

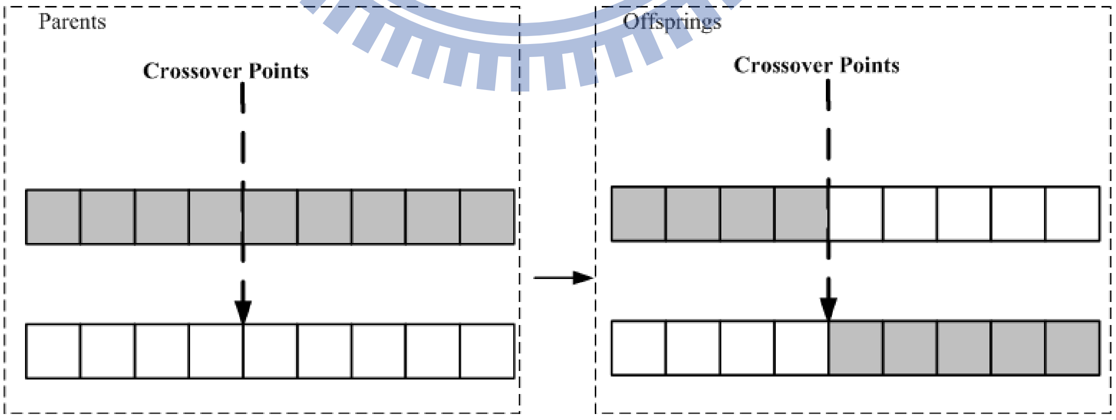


Figure 2.4: Crossover operator.

In mutation step [79]-[85], 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 2.5 illustrates how the mutation works. When an offspring is mutated, one of its genes selected randomly is changed to a new value.

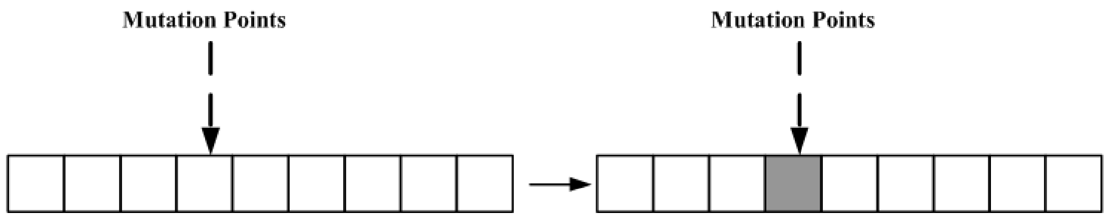


Figure 2.5: Mutation operator.

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.

2.3 Particle Swarm Optimization

In this section, we will introduce the PSO. The standard PSO is introduced in section 2.3.1 and the CPSO is introduced in section 2.3.2.

2.3.1 Standard Particle Swarm Optimization

PSO is first introduced by Kennedy and Eberhart in 1995 [8]. It's one of the most powerful methods for solving global optimization problems. The algorithm searches an optimal point in a multi-dimensional space by adjusting the trajectories of its particles. The individual particle updates its position and velocity based on its previous best performance and previous best performance of other particles which denote y and $\hat{y}$ respectively. A simple demonstration of how PSO learning proceeds can be shown in Fig. 2.7 as follows:

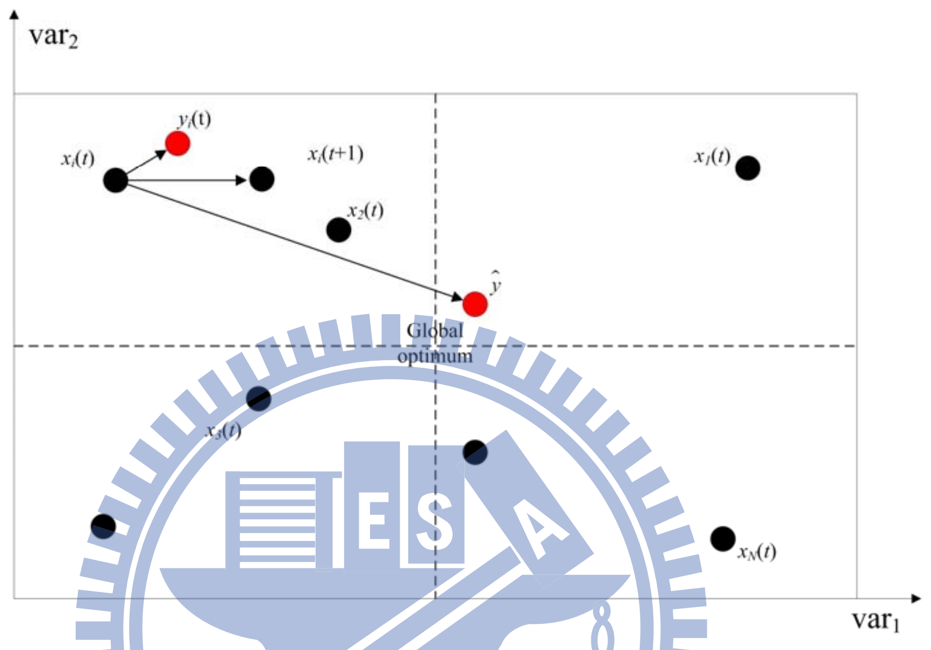


Figure 2.6: Diagram of the PSO learning mechanism.

The position $x_{i,d}$ and velocity $v_{i,d}$ of the d-th dimension of i-th particle are updated as follows:

$$\begin{aligned}&v_{i,d}(t+1) = v_{i,d}(t) + c_1 \cdot rand_1 \cdot (y_{i,d}(t) - x_{i,d}(t)) + c_2 \cdot rand_2 \cdot (\hat{y}_d(t) - x_{i,d}(t)),\\ &x_i(t+1) = x_i(t) + v_i(t+1),\end{aligned} \tag{2.7}$$

where y_i represents the previous best position yielding the best performance of the i-th particle; c_1 and c_2 denote the acceleration constants describing the weighting of each particle been pulled toward y and $\hat{y}$ respectively; $rand_1$ and $rand_2$ are two random numbers in the range [0, 1].

Let s denote the swarm size and $f()$ denote the fitness function evaluating the performance yielded by a particle. After Eq. (2.7) is executed, the personal best position y of each particle is updated as follows:

$$y_i(t+1)=\begin{cases} x_i(t+1), \text{ if } f(x_i(t+1)) \geq f(y_i(t)), \\ y_i(t+1), \text{ if } f(x_i(t+1)) < f(y_i(t)), \end{cases} \tag{2.8}$$

and the global best position is found by:

$$\hat{y}(t+1)=\arg\min_{y_i} f(y_i(t+1)), \quad 1 \leq i \leq s . \tag{2.9}$$

In 2002, Clerc [12] confirms the convergence of PSO by using a constriction factor which greatly enhances the applicability of PSO. The implementation of the constriction factor is shown in Eq. (2.10)-(2.12):

$$\begin{aligned} v_{i,d}(t+1) &= \chi[v_{i,d}(t)+c_1 \cdot rand_1 \cdot (y_{i,d}(t)-x_{i,d}(t))+c_2 \cdot rand_2 \cdot (\hat{y}_d(t)-x_{i,d}(t))], \\ x_i(t+1) &= x_i(t)+v_i(t+1), \end{aligned} \tag{2.10}$$

where

$$\chi=\frac{2}{\left|2-\phi-\sqrt{\phi^2-4\phi}\right|}, \tag{2.11}$$

and

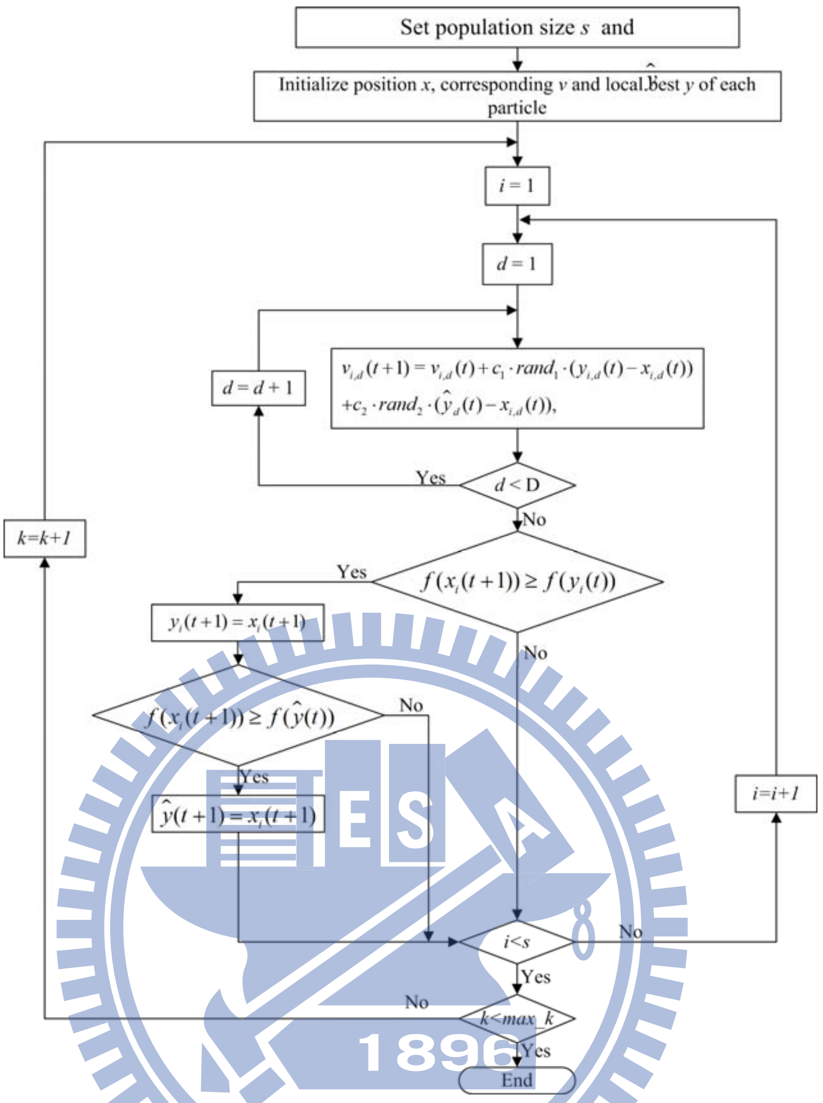
$$\phi=c_1+c_2, \ \phi>4 . \tag{2.12}$$

The flowchart of the PSO is shown in Fig. 2.7.

2.3.2 Cooperative Particle Swarm Optimization

The CPSO [9] is one of the most significant improvements to the original PSO. Van den Bergh presented a family of CPSOs, including CPSO-S, CPSO-S$_K$, CPSO-H, CPSO-H$_K$. Algorithm CPSO-H$_K$ is the hybrid from PSO and CPSO-S$_K$ and it is proposed to address the issue of "pseudominima."

The concept of CPSO-S is that instead of trying to find an optimal n-dimensional vector, the vector is split into n parts so that each of n swarms optimizes a 1-D vector. The CPSO-S$_K$ is a family of CPSO-S, where a vector is split into K parts rather than n, where $K \leq n$. K also represents the number of swarms. Each of the K swarms acts as a PSO optimizer. The main



Set population size s and
Initialize position x, corresponding v and local best $\hat{y}$ of each particle
$i = 1$
$d = 1$
$v_{i,d}(t+1) = v_{i,d}(t) + c_1 \cdot rand_1 \cdot (y_{i,d}(t) - x_{i,d}(t))$ $+c_2 \cdot rand_2 \cdot (\hat{y}_d(t) - x_{i,d}(t)),$
$d = d + 1$
Yes
$d < D$
No
$f(x_i(t+1)) \geq f(y_i(t))$
Yes
$y_i(t+1) = x_i(t+1)$
No
$f(x_i(t+1)) \geq f(\hat{y}(t))$
No
Yes
$\hat{y}(t+1) = x_i(t+1)$
$i=i+1$
$i<s$
No
Yes
$k=k+1$
No
$k<max\_k$
Yes
End

Figure 2.7: Flowchart of the PSO.

difference between the PSO and the CPSO is that the fitness of a single particle of the CPSO has to be evaluated through global best particles of the other swarms. Let P_j denote the j-th swarm and $P_j \cdot x_i$ represents the i-th particle in the swarm j. The concept of the CPSO can be illustrated as follows:

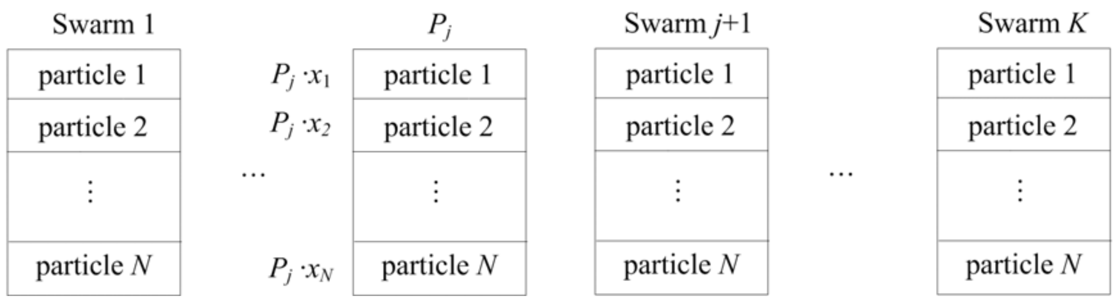


Figure 2.8: Schematic diagram of the CPSO.

The fitness of $P_j \cdot x_i$ is defined as:

$$f(P_j \bullet x_i) = f(P_1 \bullet \hat{y}, \ldots, P_{j-1} \bullet \hat{y}, P_j . x_i, \ldots, P_K \bullet \hat{y}). \tag{2.13}$$

The CPSO applies cooperative behavior to improve the PSO on find the global optimum in a high-dimensional space. This is achieved by employing multiple swarms to explore the subspaces of the search space separately to reduce the curse of dimensionality. However, there is no absolute criterion stating that the CPSO is superior to the PSO since independent changes made by different swarms on correlated variables will deteriorate its performance. In addition, in one generation of an n-dim CPSO-S operation, the computational cost is n times larger than that of a PSO operation.

2.4 Evolution Strategy with Covariance Matrix Adaptation

In this section, we introduce some background knowledge related to the proposed MS-CMA-ES. The standard CMA-ES is introduced in section 2.4.1, kernel density estimation is introduced in section 2.4.2 and mean shift procedure is introduced in section 2.4.3.

2.4.1 Standard CMA-ES

In the standard CMA-ES, a population of new search points is generated by sampling a multivariate normal distribution N with mean $m \in \mathbb{R}^n$ and covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$. The equation of sampling new search points, for each generation number $g = 0,1,2,\ldots$, reads

$$x_i^{(g+1)} \sim m^{(g)} + \sigma^{(g)} N(0, \mathbf{C}^{(g)}) \quad \text{for } i = 1, \cdots, \lambda, \tag{2.14}$$

where $\sim$ denotes the same distribution on the left and right hand side; $\sigma^{(g)}$ denotes the overall standard deviation, step-size, at generation g and λ is the sample size. The new mean $m^{(g+1)}$ of the search distribution is a weighted average of the μ selected points from λ samples $x_1^{(g+1)}, x_2^{(g+1)}, \ldots, \ x_\lambda^{(g+1)}$:

$$m^{(g+1)} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{(g+1)} , \tag{2.15}$$

with

$$\sum_{i=1}^{\mu} w_i = 1, \quad w_1 \geq w_2 \geq \cdots \geq w_1 > 0 , \tag{2.16}$$

where w_i are positive weights, and $x_{i:\lambda}^{(g+1)}$ denotes the i-th rank individual out of λ samples. The index i:λ denotes the i-th rank individual and

$$f(x_{1:\lambda}^{(g+1)}) \leq f(x_{2:\lambda}^{(g+1)}) \leq \cdots \leq f(x_{\lambda:\lambda}^{(g+1)}), \tag{2.17}$$

where $f(\cdot)$ is the objective function to be minimized. The adaptation of new covariance matrix $\mathbf{C}^{(g+1)}$ is formed by a combination of rank-μ and rank-one update [13]

$$\mathbf{C}^{(g+1)} = (1 - c_{\text{cov}})\mathbf{C}^{(g)} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}} \underbrace{p_c^{(g+1)} \left(p_c^{(g+1)}\right)^T}_{\text{rank-one update}} + c_{\text{cov}}(1 - \frac{1}{\mu_{\text{cov}}}) \times \underbrace{\sum_{i=1}^{\mu} w_i y_{i:\lambda}^{(g+1)} \left(y_{i:\lambda}^{(g+1)}\right)^T}_{\text{rank-}\mu\text{ update}} , \tag{2.18}$$

where $\mu_{\text{cov}} \geq 1$ is the weighting between rank-μ update and rank-one update; $c_{\text{cov}} \in [0,1]$ is the learning rate for the covariance matrix update, and

$$y_{i:\lambda}^{(g+1)} = (x_{i:\lambda}^{(g+1)} - m^{(g)}) / \sigma^{(g)} \tag{2.19}$$

is a modified formula used to compute the estimated covariance matrix for the selected samples. The evolution path $p_c^{(g+1)}$ for rank-one update is described as follows:

$$p_c^{(g+1)} = (1 - c_c) p_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{\text{eff}}} \, \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}} , \tag{2.20}$$

where $c_c \leq 1$ denotes the backward time horizon and

$$\mu_{\text{eff}} = \left(\sum_{i=1}^{\mu} w_i^2 \right)^{-1} \tag{2.21}$$

denotes the variance effective selection mass. The new step-size $\sigma^{(g+1)}$ is updated according to

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\left\| p_\sigma^{(g+1)} \right\|}{E\left\| N(0, \mathbf{I}) \right\|} - 1 \right) \right), \tag{2.22}$$

with

$$p_\sigma^{(g+1)} = (1-c_\sigma)p_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\,\mathbf{C}^{(g)^{-1/2}}\,\frac{m^{(g+1)}-m^{(g)}}{\sigma^{(g)}}, \tag{2.23}$$

where c_σ is the backward time horizon of evolution path, similar to c_c; d_σ is a damping parameter and $p_\sigma^{(g+1)}$ is the conjugate evolution path for step-size $\sigma^{(g+1)}$. The expectation of the Euclidean norm of a $N(0, \mathbf{I})$ reads

$$E\|N(0,\mathbf{I})\| = \sqrt{2}\Gamma(\frac{n+1}{2})/\Gamma(\frac{n}{2}) \approx \sqrt{n} + O(1/n)\,. \tag{2.24}$$

where Γ() denotes the gamma function and *O*() represents high-order terms.

2.4.2 Kernel Density Estimation

In parametric model estimation analysis, we need to suppose the distribution of data points coincides with certain model. Empirical evidence have shown that there tends to exist large differences between parametric estimation-based models and real-world physical models. Based on above defects, Rosenblatt and Parzen proposed a non-parametric way called kernel density estimator [66] to estimate the unknown p.d.f. of a random variable. The kernel density estimator does not require prior knowledge of how data distribute; instead, it analyzes the characteristic of the distribution of data. Hence, it is highly valuable in both statistical theory and application.

In the proposed MS-CMA-ES, sampled search points in the search space are considered as data in the feature space. It is very intuitive since the location of search points tends to be the phenomenal feature in function optimization problems. The rationale behind density estimation-based clustering approach is that the feature space can be regarded as the empirical p.d.f. Due to the fact that search points are sampled from normal distribution with adjusted mean and adapted covariance matrix, and are further selected according to their fitness, dense regions in the search space correspond to local maxima of the p.d.f.; in other words, the modes of the unknown density. Consider n points x_i, i = 1,…, n, in the d-dimensional space $\mathbb{R}^d$, the multivariate kernel density estimator with kernel $K(x)$ and a symmetric positive definite matrix bandwidth matrix **H**, computed in the point x is given by

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i), \tag{2.25}$$

with

$$K_h(x) = h^{-1}K(\frac{x}{h}), \tag{2.26}$$

where $K_h(x)$ is a d-variate kernel function satisfying

$$\int_{-\infty}^{\infty} K_h(x)dx = 1. \tag{2.27}$$

Normally speaking, kernel functions are symmetric, unimodal probability density functions. Uniform, normal and Epanechnikov kernel are the most common seen. It has been proven that, in certain routine conditions, kernel density estimator approximates the real density functions gradually with increasing sampling size [86]. Although the choice of different kernel functions have different effects on the results, but the effect appears small compared with the effect caused by the bandwidth, so researches focus more on the selection of bandwidth [87]. Theoretically, the selection of bandwidth is based on the mean integrated square error (MISE) between kernel density estimation and the real density function. However, the computation of MISE is too complicated. In practice, how selection of bandwidth affects the performance is analyzed by computing an asymptotic mean integrated error (AMISE) from a large number of samples. Recently, many literatures use plug-in method and cross-validation method to determine the optimal bandwidth, so that the selection of bandwidth no longer depends on the prior guess of true density function [86, 87]. In addition to the aforementioned fixed bandwidth mechanism, the variable bandwidth mechanism, bandwidth varies with different sample position, is also widely adopted in practice [88, 89]. Because it is very difficult for the fixed bandwidth mechanism to properly address multimodal density functions, especially in cases when density of each peak varies greatly. However, the analysis is relatively more complicated when compared with fixed bandwidth mechanism. In practice, the utilization of variable bandwidth mechanism is mostly based on rule of thumb [86]. If the variable bandwidth mechanism is adopted, the kernel density estimator Eq. (2.25) becomes

$$\hat{f}_{\mathbf{H}}(x) = \frac{1}{n}\sum_{i=1}^{n} K_{\mathbf{H}}(x - x_i)\,, \tag{2.28}$$

where

$$K_{\mathbf{H}}(x) = |\mathbf{H}|^{-1/2} K(\mathbf{H}^{-1/2}x)\,, \tag{2.29}$$

H is the symmetric, positive definite bandwidth matrix.

2.4.3 Mean Shift Procedure

Mean shift procedure is a very versatile tool for feature space analysis and it is applicable to many field of tasks [90-92]. In the previous research [65], authors successfully extend this algorithm to computer vision applications, and have attracted huge attention. Mean shift procedure is an iterative algorithm based on kernel density estimation, which continually updates the mean shift vectors of data points according to the gradient of kernel function. Although the mean shift algorithm is very simple in form, but in practice there is a high efficiency and stability. The most classic application is the mean shift-based clustering algorithm. If we can have a good estimation of bandwidth, mean shift-based clustering algorithm would be a nice alternative relative to algorithms that the number of clusters needs to be pre-set, such as *K*-means algorithm. In the proposed MS-CMA-ES, search points in the search space are considered as data in the feature space. It is very intuitive since location of search points tends to be the phenomenal feature in function optimization problems. Due to the fact that, in the MS-CMA-ES, sampled search points are further selected according to their fitness, dense regions in the search space correspond to local maxima of the p.d.f.; in other words, the modes of the unknown density.

Consider the density estimation kernel $K_h(x)$ introduced earlier this section. If the profile notation [21] is employed, the kernel $K_h(x)$ can also be written as

$$K_h(x) = c_{k,d}k_h(\|x\|^2)\,, \tag{2.30}$$

where $k_h(x)$ is a radially symmetric kernel defined as the profile of the $K_h(x)$, and $c_{k,d}$ is the normalization constant which makes $K_h(x)$ integrate to one. If we define

$$g_h(x) = -k_h'(x)\,, \tag{2.31}$$

the d-variate kernel $G_h(x)$ can also be written as

$$G_h(x) = c_{g,d}\, g_h(\|x\|^2)\,, \tag{2.32}$$

and similarly, $c_{g,d}$ denotes the normalization constant. The density estimation kernel $K_h(x)$ is also called the shadow kernel of $G_h(x)$ [65]. Consider n points x_i, i=1,…, n, in the d-dimensional space $\mathbb{R}^d$, the mean shift vector at x is given by

$$m(x) = \frac{\sum_{i=1}^{n} x_i G_h(x - x_i)}{\sum_{i=1}^{n} G_h(x - x_i)} - x\,. \tag{2.33}$$

Intrinsically, mean shift procedure can be viewed as a mode seeking method [59], which determines the modes of p.d.f. estimated by kernel $K_h(x)$. Denote $\{y_j\}_{j=1,2,\ldots}$ the sequence of successive search locations of kernel G_h, from Eq. (2.33) it has the form

$$y_{j+1} = \frac{\sum_{i=1}^{n} x_i G_h(y_j - x_i)}{\sum_{i=1}^{n} G_h(y_j - x_i)} \qquad j = 1, 2, \cdots \tag{2.34}$$

and y_1 is the initial search location. The corresponding sequence of density estimates computed with kernel K_h is given by

$$\left\{\hat{f}_{h,K}(j)\right\} = \hat{f}_h(y_j) \qquad j = 1, 2, \cdots. \tag{2.35}$$

In the previous research [59], authors have proven that once search location $\underline{y_j}$ gets sufficiently close to a mode of estimated density function $\hat{f}_{h,K}$, it converges to it, and the set of all locations converge to the same mode is defined as the *basin of attraction* of that mode. The general steps of applying mean shift procedure is listed as follows:

Step 1: Uniformly generate appropriate number of initial search points.

Step 2: Sequentially or parallelly run the mean shift procedure until the search points converge.

Step 3: Each convergence point defines a mode and each initial location converges to that mode defines the basin of attraction of that mode.

CHAPTER 3
EVOLUTIONARY ALGORITHMS

In this chapter, the proposed four algorithms are discussed. In section 3.1, a Q-valued based particle swarm optimization and the concept of using Lyapunov design principles for constructing safe reinforcement learning agents are introduced. In section 3.2, the proposed two-strategy reinforcement (TSR) learning mechanism and the group-based symbiotic evolution (GSE) which enables the learning agent to evaluate the fuzzy rule locally are introduced. In section 3.3, a separability detection approach to cooperative particle swarm optimization (SD-CPSO) for placing correlated variables into the same swarm is discussed. In section, 3.4, the proposed mean shift based evolutionary strategy with covariance matrix adaptation (MS-CMA-ES) is introduced. We cannot directly apply mean shift clustering to the sampled points generated by CMA-ES because the adopted mean shift clustering requires independent identity distribution of samples to perform density estimations. Several previous works such as importance sampling [93,94] and bandwidth estimation [86,87] are also discussed in this section.

3.1 Q-value based PSO

Thorough learning algorithm of QPSO is described in this section. The architecture is shown in Fig. 3.1. The whole learning process can be roughly divided into two parts: the Q-value and PSO operation part. The learning strategy for Q-values of particles is detailed in section 3.1.1 while the PSO operation and the flowchart of QPSO are described in section 3.1.2.

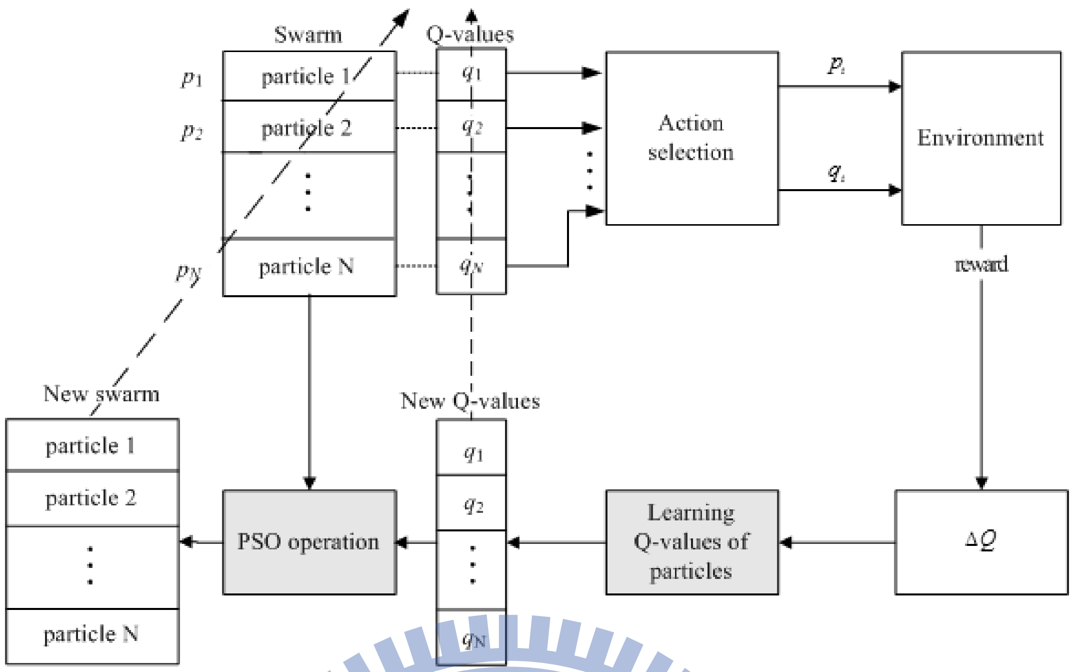


Figure 3.1: Architecture of QPSO.

3.1.1 Learning Q-values of Particles

In QPSO learning, if there are *s* particles in the swarm, *s* trials are taken in one generation. The agent applies in each trial an action to the environment by selecting a particle based on its Q-value. Every time a particle is selected, the Q-value of the selected particle is updated based on the system's reward. If the i-th particle is selected, its Q-value q_i is updated as

$$q_i(t) = q_i(t) + \alpha[-\frac{1}{t} + \gamma Q^*(x(t+1)) - q_i(t)], \tag{3.1}$$

for i=1…s, where

$$\begin{aligned} Q^*(x(t+1)) &= \max_{a' \in A(x(t+1))} Q(x(t+1),\ a') \\ &= \max_{i=1\ldots s} Q(x(t+1),\ p_i) \\ &= \max_{i=1\ldots s} q_i(t) = q_{i^*}(t). \end{aligned} \tag{3.2}$$

That is

$$\begin{aligned} q_i(t) &= q_i(t) + \alpha[-\frac{1}{t} + \gamma q_{i^*}(t) - q_i(t)] \\ &= q_i(t) + \alpha\delta_i(t), \end{aligned} \tag{3.3}$$

for i=1,$\cdots$,s, where $\delta_i(t)$ is regarded as TD error.

The new Q-values of all particles calculated from Eq. (3.3) are subsequently adopted as the fitness values for PSO evolution.

3.2.2 Q-value based PSO

The PSO operation used in QPSO consists of two major steps: swarm initialization and Q-valued base PSO evolution. Details of these two steps are described step-by step as follows.

- Swarm initialization:

  The particle swarm is composed of particles encoded by the parameters on a NFS. Each particle is encoded by the mean, deviation of Gaussian membership functions and the weightings for output action strength. The number of fuzzy rules determines the length of each particle. After the number of rules is set, the initial particles are generated according to the following equations:

$$\begin{aligned}&\text{Mean: } \mathrm{x}_i[n] = random[m_{\min}, m_{\max}],\\ &\text{where } n = 1,\ 3,\ \cdots,\ 2NR\text{-}1;\ i = 1,\ 2,\ \cdots,\ s.\end{aligned} \tag{3.4}$$

$$\begin{aligned}&\text{Deviation: } \mathrm{x}_i[n] = random[\sigma_{\min}, \sigma_{\max}],\\ &\text{where } n = 2,\ 4,\ \cdots,\ 2NR.\end{aligned} \tag{3.5}$$

$$\begin{aligned}&\text{Weight: } x_i[n] = random[w_{\min}, w_{\max}],\\ &\text{where } n = 2NR+1,\ 2NR+2,\ \cdots,\ D.\end{aligned} \tag{3.6}$$

p_i represents the *i-th* particle in the swarm; N represents the input dimension; R represents the number of fuzzy rules; D represents the size of each particle, usually D equals to $(N+1)R$ in most of cases where the dimension of output variable is 1; $[m_{\min}, m_{\max}]$, $[\sigma_{\min}, \sigma_{\max}]$ and $[w_{\min}, w_{\max}]$ are the predefined ranges. The above equations result in the coding scheme between a neural fuzzy system and a particle shown in Fig. 3.2.

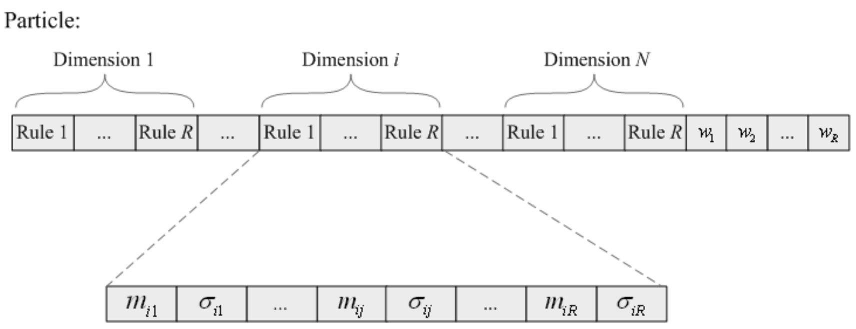


Figure 3.2: Coding scheme between a particle and a TSK-type NFS in QPSO.

- Q-value based PSO evolution:

  The Q-values derived in Eq. (3.3) are used as the fitness values for PSO evolution. The Q-value of each particle determines the performance of a particle for controlling the system. In the proposed QPSO, the Q-value of each particle indicates how soon a particle can guide the system's state to reach the set of goal states. The learning processes proceed to new generation until a predefined stop criterion is met. The block diagram of whole learning process in QPSO is shown in Fig. 3.3.

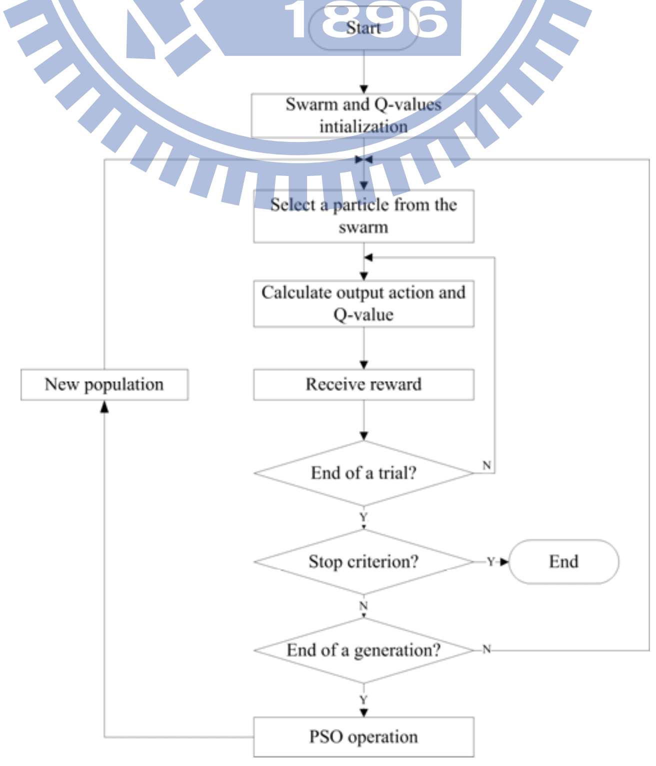


Figure 3.3: Block diagram of QPSO.

3.2 Two-strategy Reinforcement Evolutionary Algorithm

The proposed two-strategy reinforcement evolutionary algorithm (TSR-EA) is introduced in this section. Two major modifications are proposed in this algorithm: a two-strategy reinforcement signal design and the group-based symbiotic evolution (GSE). Details of these two operations are described as follows:

3.2.1 Two-strategy Reinforcement Signal Design

The TSR-EA is constructed on a TSK-type NFS model. The NFS model acts as a control network to determine a proper action according to the current input vector (environment state). The feedback signal is the reinforcement fitness value that functions as a performance measurement. The reinforcement learning architecture adopted in the TSR-EA is the time-step reinforcement architecture [95]-[97]. In this architecture, the only available feedback is a reinforcement signal that notifies the model only when a failure occurs. This architecture is straightforward and easy to implement. However, its fitness function can only indicates how long can the controller work well instead of measuring how soon the system can enter the desired state, which is also very important. Most reinforcement learning algorithms offer no guarantee on stabilizing a system around a certain operating point, or keeping the state of a system within a certain range. In this dissertation, the proposed QPSO described in section 3.1 can meet the aforementioned goals by adopting the concept of safe reinforcement learning agents based on Lyapunov design principles proposed Perkins and Barto [57]. Using the concept proposed in [57], the QPSO can guide the state of a system to reach and remain in a desired set of goal states by constraining the action choices of the agents. Actions constrained by Lyapunov design principles cause the system to descend on an appropriate Lyapunov function. The feedback reinforcement signal of in the QPSO is the time step that indicates how soon the system enters the desired set of goal states. The QPSO provides not only reliable initial learning performance but also accurate learning result. However, in order to apply Lyapunov design principles, we have to identify the Lyapunov function of a control plant in advance, which refers to the requirement of more information about the state of the control plant. For some real-world applications, some states are difficult or expensive to obtain. As a result, in the TSR-EA method, we proposed the TSR design so that our method can enjoy the convenience brought by the standard reinforcement learning architecture on one hand, and the accurate learning performance on the other. The TSR learning signal design for determining the fitness value of each learning trial is described as follows.

The proposed two strategies are judgment and evaluation. The judgment strategy measures the fitness value of a learning trial that fails to maintain the system's state in a desired operating range, whereas the evaluation strategy measures the fitness value of a learning trial that works the system well in the original successful range, but fails under a stricter successful range is applied. At first, for each different control task, a corresponding operating range *Original_Range* is predefined. Then, we shrink the original successful operating range as the control time step increases, as defined in Eq. (3.7).

$$Strict\_Range = Original\_Range \times \delta, \text{ where}$$
$$\delta = \begin{cases} 1, \text{ if } t \le A, \\ \left(Thres\_TimeStep + A - t \big/ Thres\_TimeStep \right), \text{ if } A < t \le Thres\_TimeStep, \\ \left(A \big/ Thres\_TimeStep \right), \ otherwise, \end{cases} \tag{3.7}$$

where A is a parameter that simply prevents the modified range from becoming zero. This equation provides guidance to the controller to meet the control goal sooner. The *Original_Range* and *Strict_Range* are both considered as stopping criteria. If a learning trial fails because the system state falls beyond the *Original_Range*, this learning trial is then considered as failing under a "looser" constraint. Hence, a smaller fitness value is obtained from this learning trial. On the contrary, if a learning trial fails for the system state deviating from the *Strict_Range*, this learning trial is then considered as failing under a “stricter” constraint, and a relatively larger fitness value is obtained from this learning trial. The determining fitness values in both strategies are detailed as follows:

•**Strategy 1.** Judgment strategy:

If the system fails at time step t deviating from the original successful operating range, then

$$Fitness\_Value = \frac{1}{Thres\_TimeStep}\frac{t\text{-}1}{Thres\_TimeStep}, \tag{3.8}$$

where $Thres\_TimeStep$ is a predefined parameter. A learning trial is deemed unsuccessful if it is unable to meet the control goal before *Thres_TimeStep*.

•**Strategy 2.** Evaluation strategy

Under the condition that the controller successfully maintains the system’s state in the original successful operating range, the fitness value is calculated by the following two cases. Case 1 represents the system works well under the original successful operating range but falling beyond the range defined in Eq. (3.7). Case 2 represents the controller successfully controlling the system.

Case 1. If the system enters the set of goal states at time step t_1 but falls beyond the strict successful range defined in Eq. (3.7) at time step t_2, then

$$Fitness\_Value = \frac{1}{t_1}(t_2 - t_1)\ . \tag{3.9}$$

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

$$Fitness\_Value = \ Stable\_TimeSteps + (\frac{Stable\_TimeSteps}{t_1})\ . \tag{3.10}$$

The reinforcement fitness value evaluates how soon the plant can meet the desired set of goal states and how long the controller maintains the plant within it. The advantage of the proposed TSR-EA method is that it provides a relatively accurate learning performance compared with standard time-step reinforcement architecture.

3.2.2 Group-based Symbiotic Evolution

In this section, the idea of GSE is introduced. Unlike traditional GA that uses each individual in a population as a full solution to a problem, GSE assumes that each individual in a population represents only a partial solution to a problem. In a standard evolution algorithm, a single individual is responsible for the overall performance, with a fitness value assigned to that individual according to its performance. In the GSE, in order to calculate the fitness of an individual (a partial solution), we have to combine the current individual with other "global best" individuals of other groups to form a context vector first. A context vector stands for a complete solution and can be used to evaluate the fitness value. This idea is adopted from the CPSO introduced earlier. Let x_j denote the *j*-th chromosome and $P_j \bullet x_i$ represents the *i*-th chromosome in the group *j*. Then the fitness of $P_j \bullet x_i$ is defined as:

$$f(P_j \bullet x_i) = f(P_1 \bullet \hat{y}, \ldots, P_{j-1} \bullet \hat{y}, P_j x_i, \ldots, P_K \bullet \hat{y}). \tag{3.11}$$

As shown in [96-100], 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, the symbiotic evolution finds solutions in different, unconverted populations. With the fitness assignment performed by GSE, and the local property of a fuzzy rule, GSE and the fuzzy system design can complement each other.

The structure of the GSE is shown in Fig. 3.4, where *Ncs* is the number of complete solutions the GSE will select individuals to form in one generation, *R* denotes the number of fuzzy rules in a NFS.

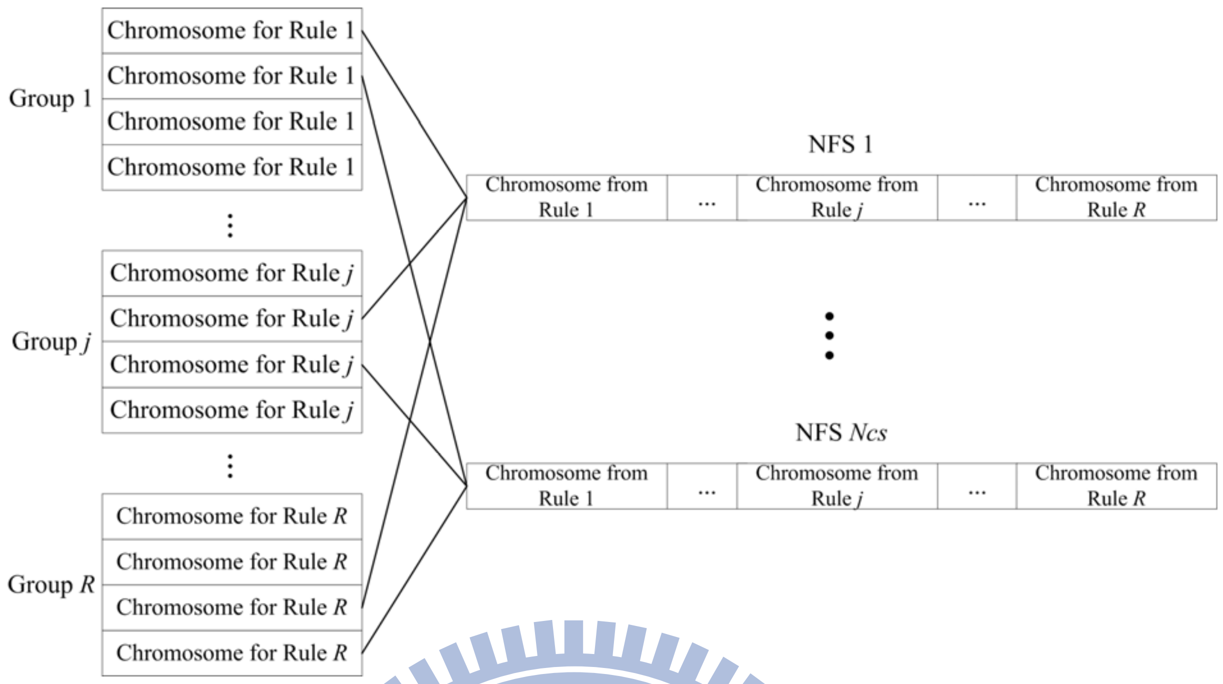


Figure 3.4: Structure of a chromosome in the GSE.

The coding structure of a chromosome is shown in Fig. 3.5, which describes that where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation, respectively, and w_j is the weight of the *j*th rule node and *n* denotes the input dimension.

m_{1j}	σ_{1j}	…	m_{ij}	σ_{ij}	…	m_{nj}	σ_{nj}	w_j

Figure 3.5: Coding structure of a chromosome in the TSR-EA.

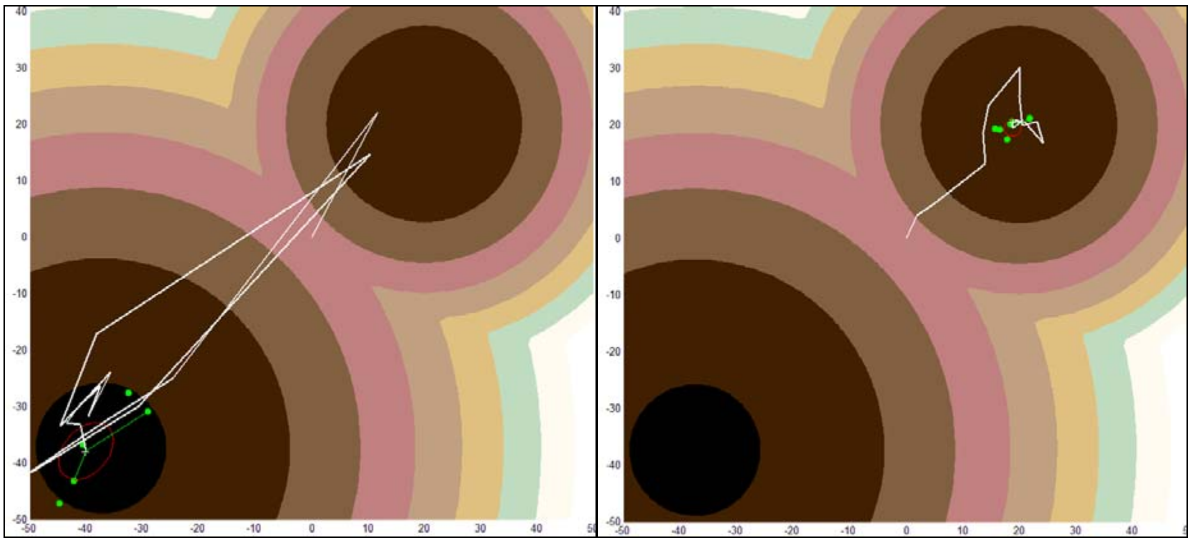
3.3 Mean Shift-Based Evolution Strategy with Covariance Matrix Adaptation

Evolution strategy with covariance matrix adaptation (CMA-ES) is very effective in optimization of unimodal functions, but inferior to other algorithms that emphasize the global search ability, such as particle swarm optimization (PSO) or differential evolution (DE), in optimization of multi-funnel functions. Enhancing the global search ability of CMA-ES has becoming urgent goals of many scholars within the field. In this dissertation, we propose a mean shift based CMA-ES (MS-CMA-ES). The framework of proposed method is constructed on CMA-ES. In the traditional CMA-ES, new search points are sampled from normal distribution; however, in the MS-CMA-ES, new search points are sampled from mixture normal model. The introduced mean shift procedure is a clustering method, which allows us to apply multiple CMA-ES instances to explore multiple search directions in parallel according to the its clustering result. In the MS-CMA-ES, the mean shift procedure is also used to compute the mean vector of the mixture normal distribution. During the mean shift procedure, each search point is "shifted" toward their corresponding local optima area of the p.d.f. until all search points converge. The converge points represents new mean vectors of the mixture normal model; in other words, the initial sampling locations of the MS-CMA-ES.

In this dissertation, we mainly focus on studying how to apply mean shift based clustering approach in optimization of complex objective functions, detecting their modes, and try to preserve the advantage of CMA-ES that converge rapidly in optimization of single-funnel functions. In the chapter we will detail the architecture of our method and its learning process. In section 3.3.1, we will describe our motivation, which is followed by a block diagram learning process. The detail of each block will be described in section 3.3.2 through 3.3.4.

3.3.1 Motivation

Before introducing the proposed MS-CMA-ES, we observe a drawback that CMA-ES may encounter as shown in Fig. 3.6:



(a) (b)

Figure 3.6: Computer simulation result of CMA-ES with initial search location at (0, 0).

Figure 3.6 is an computer simulation result of an optimization problem with two local optimal solution. The upper right region is a suboptimal region with steeper gradient toward it and the lower left region is the global optimal region with a smoother trend. As shown in Fig. 3.6, white lines represents the locus of average of search points and darker background color stands for higher fitness value. The initial search location is at (0,0) which is in the middle of two local optimal solutions. The ideal case is that the locus wanders between two local optimal regions then converge to the lower left global optimal region as shown in Fig. 3.6(a). However, we found by simulation that most of times the locus only temporary wanders and converge to the upper right region as shown in Fig. 3.6(b). The search direction of CMA-ES cannot continually expand to two optimal regions and determine the real optimal solution according to their converge points.

In this dissertation we think this drawback is due to the fact that the sampling distribution is limited to normal distribution. In statistical learning [66], simple normal model is not enough to deal with complex problems, an advanced alternative is the mixture model. Mixture normal model can effectively approximate the p.d.f. of multimodal functions, and reveal their important characteristics: number and location of the modes. In this dissertation, we think the aforementioned drawback can be relieved if the distribution of search points are sampled by a mixture normal model.

Recently there are researches attempt to turn CMA-ES search pattern into multiple region search style; for example, authors [26] incorporates particle swarm optimization to enhance the global search ability of CMA-ES. In this dissertation, we adopt the different concept by altering the sampling model. After sampling the search points, all samples are clustering by mean shift procedure. A new cluster represents a new CMA-ES instance. By perspective of mixture model, a new cluster stands for a new component of the mixture. Observed from the computer simulation result, the proposed mechanism can alleviate the deficiency that CMA-ES cannot search multiple directions in parallel. The block diagram of the proposed MS-CMA-ES is shown in the following figure and the detail of each block will be introduced in the subsequent sections.

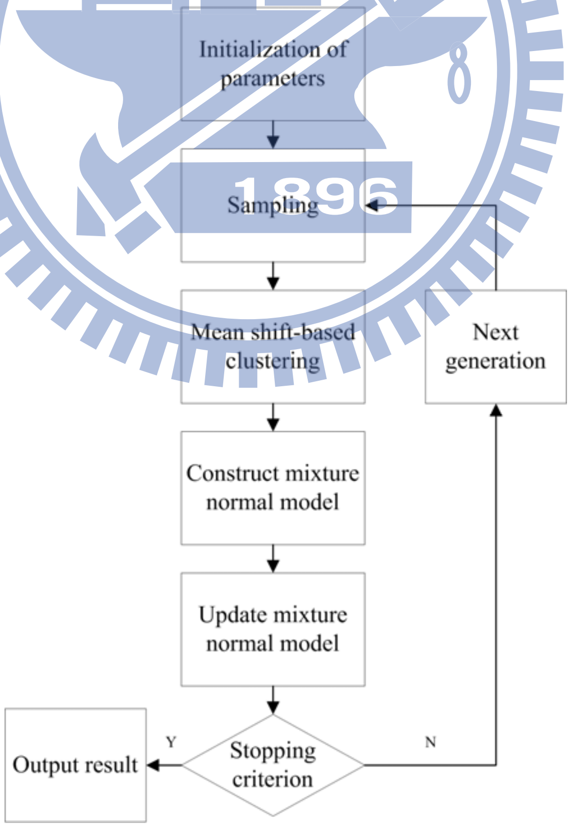

Initialization of parameters
Sampling
Mean shift-based clustering
Next generation
Construct mixture normal model
Update mixture normal model
Output result
Y
Stopping criterion
N

Figure 3.7: The block diagram of the MS-CMA-ES.

3.3.2 Sampling from a Mixture Model

In the proposed MS-CMA-ES, the sampling of new search points is given by

$$x_i^{(g+1)} \sim N_{\text{mix}}(m^{(g)}, \sigma^{(g)}, \mathbf{C}^{(g)}, \alpha^{(g)}) \quad \text{for } i = 1, \cdots, \lambda, \tag{3.12}$$

where N_{mix} denotes a mixture Gaussian distribution with its p.d.f. $p_M(\bullet)$ reads

$$p_M(x) = \sum_{k=1}^{K^{(g)}} \alpha^{(g)}(k) G\left(x; m^{(g)}(k), \left(\sigma^{(g)}(k)\right)^2 \mathbf{C}^{(g)}(k)\right). \tag{3.13}$$

G(•; θ) denotes a multivariate Gaussian function parameterized by θ; k denotes the index of component of mixture model; $K^{(g)}$ is the total number of components at generation g; λ denotes the total number of samples $\lambda = \lambda_1^{(g)} + \lambda_2^{(g)} + \cdots + \lambda_{K^{(g)}}^{(g)}$; $m^{(g)}$ is the set of mean of search points $m^{(g)} \equiv \{m^{(g)}(1), \ldots, m^{(g)}(K^{(g)})\}$; $\mathbf{C}^{(g)}$ is the set of covariance matrix $\mathbf{C}^{(g)} \equiv \{\mathbf{C}^{(g)}(1), \ldots, \mathbf{C}^{(g)}(K^{(g)})\}$; $\sigma^{(g)}$ is the set of search step size $\sigma^{(g)} \equiv \{\sigma^{(g)}(1), \ldots, \sigma^{(g)}(K^{(g)})\}$; $\alpha^{(g)}$ is the mixture weighting $\alpha^{(g)} \equiv \{\alpha^{(g)}(1), \ldots, \alpha^{(g)}(K^{(g)})\}$.

3.3.3 Mean Shift-based Clustering

The total number of components $K^{(g)}$ is variable at each generation and is determined by the result of clustering. In this dissertation, we don't directly apply clustering method to the sampled points because the adopted mean shift based clustering method requires independent identity distribution of samples to perform density estimations. In our method, search points are sampled from a Gaussian mixture model. Due to the absence of independent identity distribution of samples, we introduce the importance sampling method [93, 94] to find a more reliable density estimator. Let us denote f the unknown p.d.f. of the search space and $\hat{f}_{\mathbf{H}}$ the kernel density estimation with bandwidth matrix $\mathbf{H}$. The kernel density estimation of f after introducing importance sampling method is given by

$$\hat{f}_{\mathbf{H}} = \sum_{i=1}^{\lambda} \omega_i K_{\mathbf{H}}(x - x_i), \quad \text{for } i = 1, \cdots, \lambda, \tag{3.14}$$

which is almost identical with traditional kernel density estimation besides the importance weighting ω_i

$$\omega_i = \frac{w_i / p_M(x_i)}{\sum_{m=1}^{\lambda} w_m / p_M(x_m)}, \tag{3.15}$$

where w_i is the fitness weighting of x_i; $p_M(\bullet)$ is the p.d.f. of mixture normal distribution shown in Eq. (3.1w). The sequence of mean shift-based clustering of each search point after introducing importance sampling is given by

$$\begin{aligned} x_i^{(t+1)} &= \left[\sum_{m=1}^{\lambda} \omega(x_m) \left| \mathbf{H}_m \right|^{-1/2} \exp\left(-\frac{1}{2} D^2(x_i^{(t)}, x_m; \mathbf{H}_m) \right) \mathbf{H}_m^{-1} \right]^{-1} \\ &\quad \times \left[\sum_{m=1}^{\lambda} \omega(x_m) \left| \mathbf{H}_m \right|^{-1/2} \exp\left(-\frac{1}{2} D^2(x_i^{(t)}, x_m; \mathbf{H}_m) \right) \mathbf{H}_m^{-1} x_m \right], \\ &\text{for } i = 1, \cdots, \lambda, \end{aligned} \tag{3.16}$$

where

$$D^2(x_i^{(t)}, x_m; \mathbf{H}_m) = (x_i^{(t)} - x_m)^T \mathbf{H}_m^{-1} (x_i^{(t)} - x_m), \tag{3.17}$$

and D is the Mahalanobis distance of $x_i^{(t)}$ to x_m and $\mathbf{H}_m$ is the positive definite bandwidth matrix for x_m. $\mathbf{H}_m$ is another important parameter needs to be determined. In general, when doing kernel density estimation, literatures process adequate, at least 50 to 100, amount of samples. In such cases, the selection of bandwidth matrix can be achieved based on the analysis of asymptotic mean integrated square error (AMISE) [87]. The proposed MS-CMA-ES is mainly constructed on traditional CMA-ES that only generates few samples at each generation; therefore, we cannot cite AMISE based bandwidth selection methods which have richer research results. In this dissertation, the selection of bandwidth matrix is according to Theorem 3.1, the analysis of MISE [87], which is more applicant to cases with small amount of samples.

First, we derive equations of optimal bandwidth matrix for samples within a same cluster. In the following derivation, we ignore the term index and the superscript of variables $m^{(g)}(k)$, $\sigma^{(g)}(k)$, and $\mathbf{C}^{(g)}(k)$ since we only consider samples within a same cluster at a certain generation:

Theorem 3.1 [87] *Consider $K_{\mathbf{H}}$ a kernel function parameterized by bandwidth matrix **H**, and the true distribution of samples is N(m, **Σ**). The optimal MISE bandwidth for density estimation is given by*

$$\begin{aligned} \boldsymbol{H}^* &= \arg\min_{\boldsymbol{H}} MISE\{\hat{f}(g;\boldsymbol{H})\} \\ &= \arg\min_{\boldsymbol{H}} \left\{ n^{-1}(4\pi)^{-\frac{d}{2}}|\boldsymbol{H}|^{-\frac{1}{2}} + (1+n^{-1})(4\pi)^{-\frac{d}{2}}|\boldsymbol{H}+\boldsymbol{\Sigma}|^{-\frac{1}{2}} \right. \\ &\quad \left. -2(2\pi)^{-\frac{d}{2}}|\boldsymbol{H}+2\boldsymbol{\Sigma}|^{-\frac{1}{2}} \right\}, \end{aligned} \tag{3.18}$$

*where d is the dimension of samples, n is the number of samples, m and **Σ** are the mean and the covariance matrix of the normal distribution respectively.*

In this dissertation, we let **Σ** be the covariance matrix adapted by CMA-ES

$$\mathbf{\Sigma} = \sigma^2\mathbf{C}; \tag{3.19}$$

in other words, we assume that the true distribution of samples is similar to the normal distribution adapted by CMA-ES. The covariance matrix **C** stands for a favorable shape of distributing samples for finding local optimums and we expect it to be a good approximation to the true distribution of samples. The global step size σ stands for the bandwidth of kernel density estimation and it is self-adaptive in CMA-ES algorithm. From experimental observations, the smaller the bandwidth is, the more number of modes will be estimated and the larger bandwidths correspond to smoother estimation results.

The dimension of **H** is d^2; in other words, there are d^2 parameters need to be optimized at each generation according to Eq. (3.18), which is very computationally expensive. In this dissertation, we propose a method to prevent the d^2 optimization task at each generation according to the following theorem [87]:

Theorem 3.2 [87] *Consider $K_{\boldsymbol{H}}$ a kernel function parameterized by bandwidth matrix **H**, and the true distribution of samples is N(m, **Σ**). The optimal AMISE bandwidth for density estimation satisfies*

$$\mathbf{H}^*_{AMISE} = h\mathbf{\Sigma}. \tag{3.20}$$

Based on theorem 3.2, we limit **H** to the following equation

$$\mathbf{H} = h\mathbf{\Sigma} = h\sigma^2\mathbf{C}, \tag{3.21}$$

where h denotes the global width of the bandwidth matrix. According to the eigen-decomposition theorem

$$\mathbf{\Sigma} = \sigma^2\mathbf{C} = \sigma^2\mathbf{B}\mathbf{D}^2\mathbf{B}^T, \tag{3.22}$$

and the following fact

$$|\mathbf{H}| = h^d\left|\sigma^2\mathbf{D}\right|,\ |\mathbf{H}+\mathbf{\Sigma}| = (h+1)^d\left|\sigma^2\mathbf{D}\right|,\cdots,|\mathbf{H}+k\mathbf{\Sigma}| = (h+1)^k\left|\sigma^2\mathbf{D}\right|, \tag{3.23}$$

the search of optimal bandwidth matrix can be simplified to a 1-dim optimization problem relevant only to n and d

$$\begin{aligned} h^* &= \arg\min_{h>0}\left\{ n^{-1}(4\pi)^{-\frac{d}{2}} h^{-\frac{d}{2}}\left|\sigma^2\mathbf{D}^2\right|^{-\frac{1}{2}} + (1+n^{-1})(4\pi)^{-\frac{d}{2}}(h+1)^{-\frac{d}{2}}\left|\sigma^2\mathbf{D}^2\right|^{-\frac{1}{2}} \right. \\ &\quad \left. -2(2\pi)^{-\frac{d}{2}}(h+2)^{-\frac{d}{2}}\left|\sigma^2\mathbf{D}^2\right|^{-\frac{1}{2}} \right\} \\ &= \arg\min_{h>0}\left\{ n^{-1}h^{-\frac{d}{2}} + (1+n^{-1})(h+1)^{-\frac{d}{2}} - 2^{\frac{d+2}{2}}(h+2)^{-\frac{d}{2}} \right\}. \end{aligned} \tag{3.24}$$

In this dissertation, we use steepest descent method to compute optimal solutions of Eq. (3.24) as a database indexed for n = 1, 2,…, 50 and d = 1, 2,…,50.

After deriving equations of optimal bandwidth matrix for samples within the same cluster, the bandwidth matrix for each sample can be assigned according to its cluster index to complete the mean shift-based clustering method. The proposed kernel density estimation method density estimation utilizes the variable bandwidth selection, which is necessary considering that search points are sampled by a mixture model distribution. The following figure shows an example of applying kernel density estimation with variable bandwidth mechanism to complete the mean shift-based clustering. Search points shown in Fig. 3.8(a) are sampled by a 2-component mixture probability distribution. Pink contour represents high fitness value while blue contour represents the opposite. Clustering result of search points shown in Fig. 3.8(b), three clusters are determined and marked by three different colors. Figure 3.8(b) shows the result of density estimation with samples generated by a 2-component mixture probability distribution converge to three modes. After mean shift-based clustering, each cluster forms a separate component of a mixture probability model. The updating of parameters of the mixture probability model will be introduced at the next section.

3.3.4 Updating of Mixture Probability Model

In this section, we derive parameters of sampling new search points in the MS-CMA-ES. As described in section 3.3.1, new search points at generation g+1, $\left\{x_i^{(g+1)}\right\}$, are sampled by a mixture normal model parameterized by $m^{(g)}$, $\mathbf{C}^{(g)}$, $\sigma^{(g)}$ and $\alpha^{(g)}$. After sampling, the classification of each search points and the number of clusters $K^{(g+1)}$; in other words, the number of components of mixture probability model are determined by the mean shift based clustering method. Before deriving equations of updating $m^{(g)}$, $\mathbf{C}^{(g)}$, $\sigma^{(g)}$ and $\alpha^{(g)}$, we introduce two operators

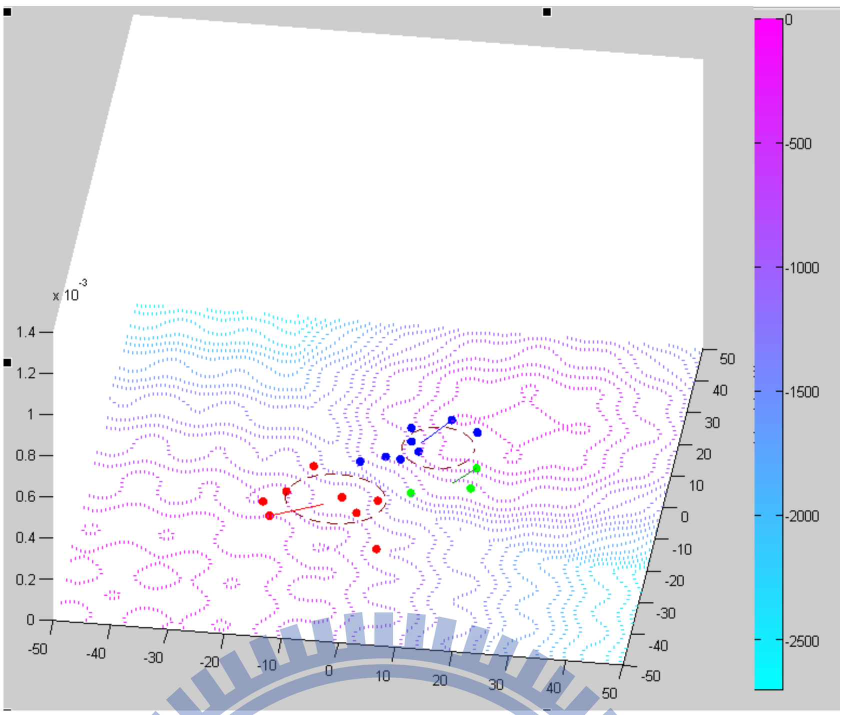
$$z(k,i)=\begin{cases}0, \text{ if } x_i \notin \text{cluster } k, \\ 1, \text{ if } x_i \in \text{cluster } k,\end{cases} \tag{3.25}$$

and $\kappa_i \in \{1,\ldots,K\}$ represents the cluster index of sample x_i. The updating rule of $m^{(g)}$ of the k-th cluster reads

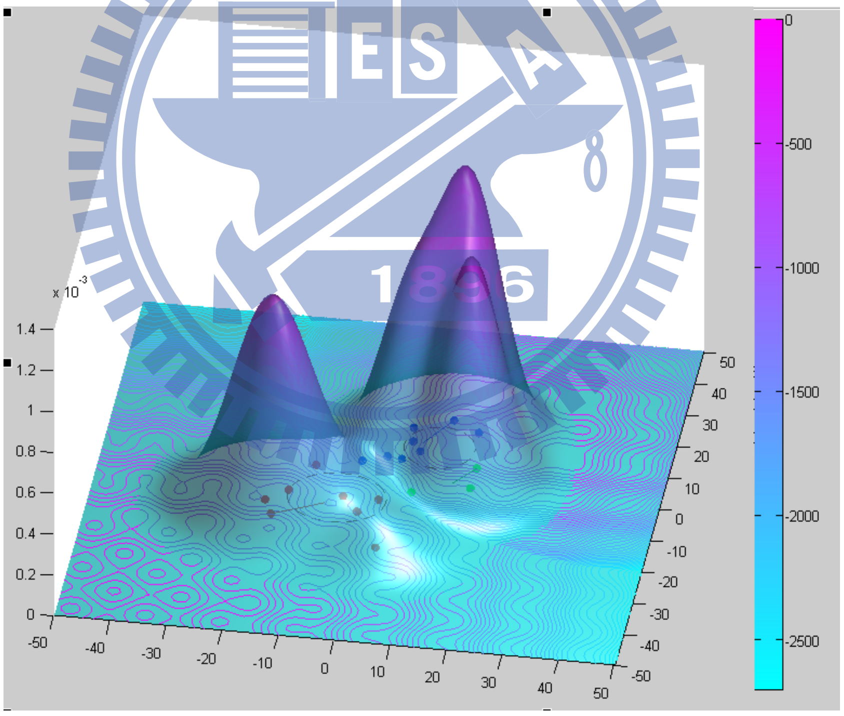
$$m^{(g+1)}(k)=\sum_{i=1}^{\lambda} w_i z(k,i) x_i^{(g+1)}, \text{ for } k=1,\cdots,K^{(g+1)}. \tag{3.26}$$

The updating rule of $\mathbf{C}^{(g)}$ of the k-th cluster reads

$$\mathbf{C}^{(g+1)}(k)=(1-c)\tilde{\mathbf{C}}^{(g)}(k)+c\sum_{i=1}^{\lambda} z(k,i) w_i y_i^{(g+1)}\left(y_i^{(g+1)}\right)^T, \tag{3.27}$$



(a)



(b)

Figure 3.8: Example of kernel density estimation with variable bandwidth selection.

The updating rule of $\sigma^{(g)}$ of the k-th cluster reads

$$\sigma^{(g+1)}(k) = \tilde{\sigma}^{(g)}(k) \exp\left[\frac{c_\sigma}{d_\sigma} \left(\frac{\left\| p_\sigma^{(g+1)}(k) \right\|}{E\left\| N(0, \mathbf{I}) \right\|} - 1 \right) \right], \tag{3.30}$$

where

$$p_\sigma^{(g+1)}(k) = (1-c_\sigma)\tilde{p}_\sigma^{(g)}(k) + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}} \sum_{i=1}^{\lambda} z(k,i) w_i \left(\mathbf{C}^{(g)}(\kappa_i)\right)^{-\frac{1}{2}} \frac{x_i^{(g+1)} - m^{(g)}(\kappa_i)}{\sigma^{(g)}(\kappa_i)}, \tag{3.31}$$

$$\tilde{p}_\sigma^{(g)}(k) = \frac{\sum_{i=1}^{\lambda} z(k,i) p_\sigma^{(g)}(\kappa_i)}{\sum_{i=1}^{\lambda} z(k,i)}, \text{ for } k = 1, \cdots, K^{(g+1)}, \tag{3.32}$$

and

$$\tilde{\sigma}^{(g)}(k) = \frac{\sum_{i=1}^{\lambda} z(k,i) \sigma^{(g)}(\kappa_i)}{\sum_{i=1}^{\lambda} z(k,i)}. \tag{3.33}$$

Compared to the traditional CMA-ES, the updating of $m^{(g)}$, $\mathbf{C}^{(g)}$ and $\sigma^{(g)}$ for the MS-CMA-ES are performed in each cluster. Equations are slightly different besides the two introduced operator that indicates the cluster index of a search sample. The MS-CMA-ES also introduces a set mixture weightings $\alpha^{(g)}$ as new variables, and its updating rule is given by

$$\alpha^{(g+1)}(k) = (1-c_\alpha)\tilde{\alpha}^{(g)}(k) + c_\alpha \hat{\alpha}^{(g)}(k), \tag{3.34}$$

where

$$\tilde{\alpha}^{(g)}(k) = \frac{1}{\lambda} \sum_{i=1}^{\lambda} z(k,i), \text{ for } k = 1, \cdots, K^{(g+1)}. \tag{3.35}$$

$\hat{\alpha}^{(g)}(k)$ represents the objective updating value and c_α denotes the updating step size. In this dissertation, we set the objective updating values as the density estimation values of the modes, which were obtained from the mean shift-based clustering result:

$$\hat{\alpha}^{(g)}(k) = f_{\text{KDE}}(\text{mode}_k^{(g)}), \tag{3.36}$$

where $f_{\text{KDE}}(\text{mode}_k^{(g)})$ denotes the kernel density estimation of the mode of the k-th cluster. Macroscopically, Eq. (3.34)-(3.36) are performing selection among components. The above equations perform a series of comparison and elimination in the hierarchical structure formed by the mixture model to explore the local search ability at different search locations, which is also the key to the global optimization.

3.4 A Separability Detection Approach to Cooperative Particle Swarm Optimizer

In this section we introduce an approach to help the CPSO self-organize the swarms composed of non-separable variables. Consider a particular optimization task illustrated in Fig. 3.9, from which we can see a 2-dim function with a bar-shaped local optimal region and a global optimum lies in it. The task is to find its global optimum by particle swarm optimizer. At first, particles are uniformly distributed in the search space. At this moment, we expect particles to be divided into two swarms, performing separate 1-dim PSO operation on each dimension to speed up the process of particles gathering around the optimal region.

If by any chance particles gather around the optimal region as we expected, as shown in Fig. 3.10. At this point of time, we prefer particles performing 2-dim PSO operation on the whole search space to reduce the computational cost, which, in this case, represents the number of function evaluations.

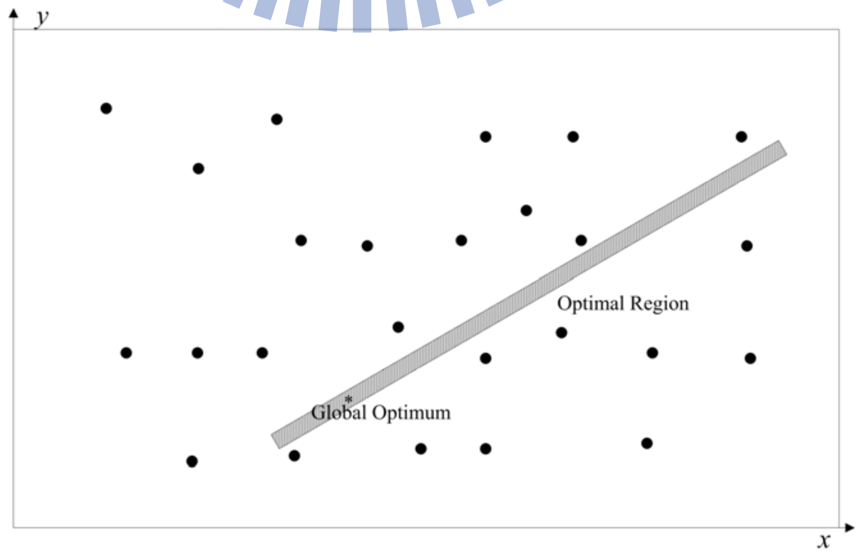


Figure 3.9: Case with particles uniformly distributed in the search space to find the global optimum lies in a bar-shaped local optimal region.

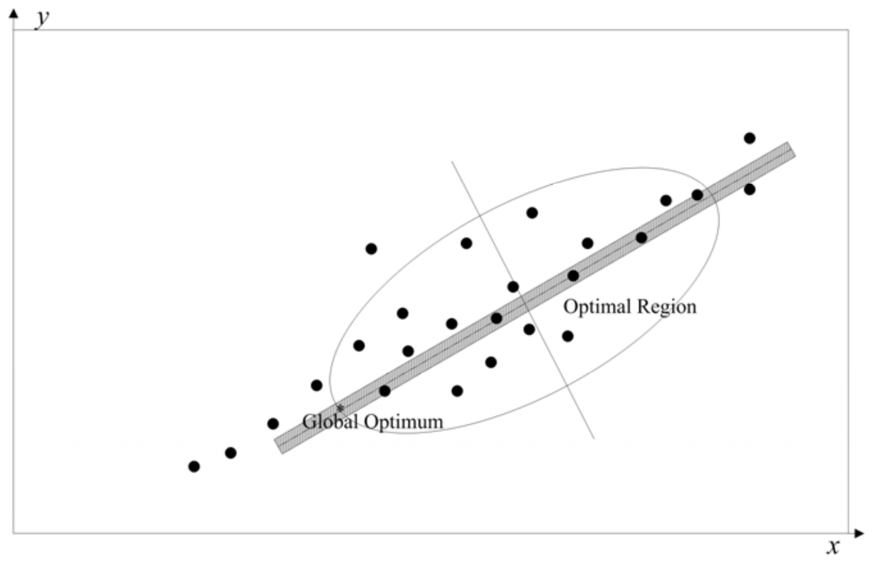


Figure 3.10: Case with particles gather around the bar-shaped optimal region to find the global optimum.

In order to implement the idea illustrated above, we have to determine the timing of switching between the PSO and the CPSO operation when dealing with a task. In this paper, we think this can be done by determining the separability between variables, and placing non-separable into the same swarm at each generation. If at certain moment, all variables are determined as non-separable, then the PSO operation is taken; otherwise, the CPSO operation is taken.

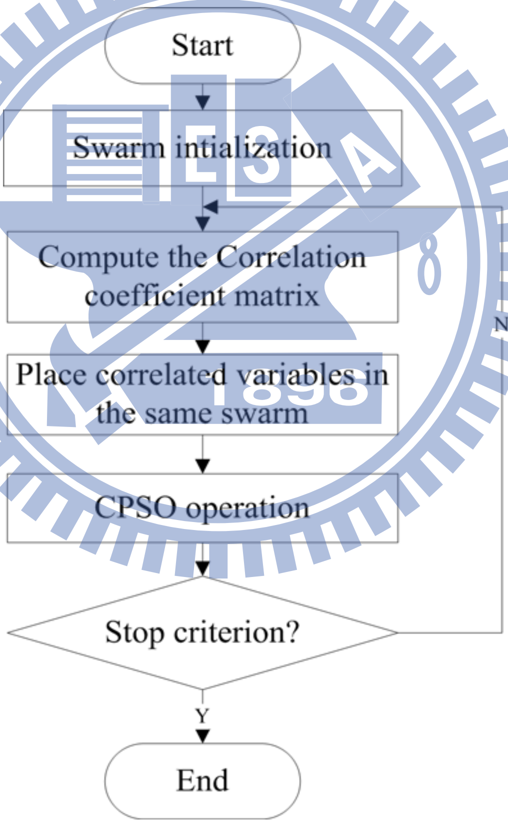
The separability between variables is found by estimating the covariance matrix of the distribution of particles. Instead of computing the sample covariance matrix of the distribution of particles directly, we adopt the CMA mechanism to estimate the covariance matrix of the distribution of particles. The adaptation of new covariance matrix $\mathbf{C}^{(g+1)}$ is formed by a combination of rank-μ and rank-one update. Detailed adaptation equations can be seen from Eq. (2.14)-(2.24). Consider the estimated covariance matrix has the form shown as follows,

$$\mathbf{C} = \begin{bmatrix} c_1^2 & c_{12} & \cdots & c_{1n} \\ \vdots & c_2^2 & \cdots & \vdots \\ \vdots & \vdots & & \vdots \\ c_{1n} & c_{2n} & \cdots & c_n^2 \end{bmatrix}, \tag{3.37}$$

where n is the number of dimensions, c_{jk} represents the weighted covariance between variables *j* and *k*. The separability between dimensions can be obtained from correlation coefficient matrix with its element defined as follows:

$$\rho_{jk} = c_{jk} / c_j c_k \, , \tag{3.38}$$

We define a parameter ρ_{thres} to determine whether dimension *j* and *k* are separable. If $\rho_{jk} < \rho_{\text{thres}}$ then we say variable *j* and *k* are separable. Conventionally, if $|\rho|>0.8$, it implies that there exists a very strong linear relationship between these two variables; $0.8>|\rho|>0.6$ implies strong relationship, and $0.6>|\rho|>0.4$ implies moderate relationship. In this dissertation, we avoid setting ρ_{thres} less than 0.6. The block diagram of the SD-CPSO can be found in Fig. 3.11.



Start
Swarm intialization
Compute the Correlation coefficient matrix
Place correlated variables in the same swarm
CPSO operation
Stop criterion?
N
Y
End

Figure 3.11: Block diagram of SD-CPSO.

CHAPTER 4

SIMULATIONS

To verify the performance of four algorithms proposed in this dissertation, three optimization tasks and performance contrasts with some other models are presented. The optimization tasks can be categorized into reinforcement learning control task and multi-funnel function optimization task. We apply the QPSO and TSR-EA to high-dimensional, reinforcement learning control tasks, and apply the MS-CMA-ES and SD-CPSO to complex, low-dimensional multi-funnel function optimization task. The optimization tasks used to compare the performance of the proposed four algorithms with other existing models are described as follows.

In Section 4.1, the cart-pole balance control [101] and the control of a double-link inverted pendulum system [102] are adopted to evaluate the performance of the proposed QPSO and TSR-EA. These problem are often used as examples of inherently unstable and dynamic systems to demonstrate both modern and classical control techniques or the reinforcement learning schemes.

In Section 4.2, we will compare the performance of the MS-CMA-ES and SD-CPSO with other existing models through real-valued function optimization tasks [103]. In section 4.2.1, we introduce a simple computer simulation that illustrates the improvement of the MS-CMA-ES over standard CMA-ES on global search ability. In section 4.2.2, the test environment and the comparison results are presented.

4.1 Reinforcement Learning Tasks

Two computer simulations are discussed in this section. The first simulation is the cart-pole balance control and the second simulation is the control of a double-link inverted pendulum system.

Example 1: Control of a cart-pole balancing system

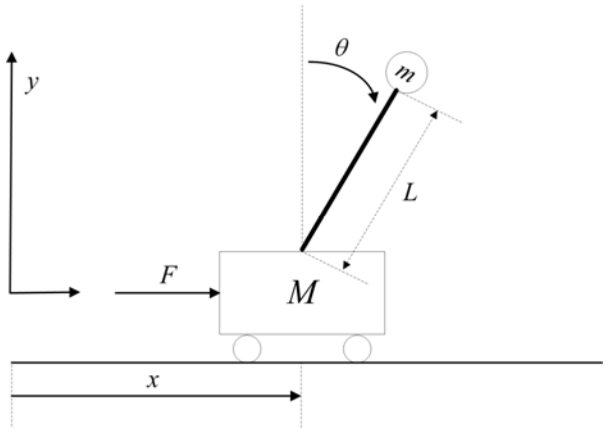


Figure 4.1: Single-link inverted pendulum system.

Figure 4.1 depicts the cart-pole balancing system. The bottom of the pole is hinged to a cart that travels along a finite-length track to its right or left. Both the cart and pole can move only on the vertical plane; that is, each has only one degree of freedom. The only control action is *F*, which is the amount of force (in Newtons) applied to the cart to move it left or right. The system fails when the cart runs into the bounds of its track (the distance is 2.4 m from the center to each bound of the track) or when the pole deviates more than 90 degrees. Using Lagrange's method, the model of the cart-pole balancing system can be obtained as follows:

$$x:\quad (m+M)\ddot{x}+mL(\ddot{\theta}\cos\theta-\dot{\theta}^2\sin\theta)=F\,, \tag{4.1}$$

$$\theta:\quad \ddot{x}\cos\theta+L\ddot{\theta}-g\sin\theta=0\,, \tag{4.2}$$

where L = *0.5* m, the length of the pole; M = 1.0 kg, the mass of the cart; m = 0.1 kg, the mass of the pole, and g = 9.8 m/s, the acceleration due to gravity. $[m_{\min}, m_{\max}]$, $[\sigma_{\min}, \sigma_{\max}]$ and $[w_{\min}, w_{\max}]$ are set as [0, 2], [0, 2] and [-30, 30], respectively.

By letting $q=(x,\theta)^T$, we can rewrite Eqs. (4.1) and (4.2) into general dynamic forms as follows:

$$D(q)\ddot{q}+C(q,\,\dot{q})\dot{q}+G(q)=\tau\,, \tag{4.3}$$

$$D(q)=\begin{bmatrix} m+M & mL\cos\theta \\ mL\cos\theta & mL^2 \end{bmatrix}, \tag{4.4}$$

$$C(q,\dot{q})=\begin{bmatrix} 0 & -mL\dot{\theta}\sin\theta \\ 0 & 0 \end{bmatrix}, \tag{4.5}$$

$$G(q)=\begin{bmatrix} 0 \\ -mgL\sin\theta \end{bmatrix}, \tag{4.6}$$

$$\tau=\begin{bmatrix} F & 0 \end{bmatrix}^T. \tag{4.7}$$

The total mechanical energy of the system can be derived from:

$$E(q,\ \dot{q})=\frac{1}{2}\dot{q}^T D(q)\dot{q}+P(q), \tag{4.8}$$

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 the sequence of forces applied to the cart to balance the pole upright and keep the cart as stationary as possible. Hence, we define a goal set comprising near-upright and near-stationary states as $G_1=\left\{(q,\dot{q}):\ \left\|(\dot{x},\theta,\dot{\theta})\right\|\le 0.001\right\}$. When the state of the cart-pole balancing system is in G_1, according to Eq. (4.8), the total mechanical energy E of the system is mgL, denoting E_{top}. We define a Lyapunov function $L(q,\ \dot{q})=\frac{1}{2}\left(E_{\text{top}}-E(q,\ \dot{q})\right)^2$. The purpose of this control problem can be transformed from "balancing the pole upright and keeping the cart as stationary as possible" to "guiding the system's mechanical energy $E(q,\ \dot{q})$ to reach E_{top} and maintaining it near E_{top} as long as possible;" that is, achieving $L(q,\ \dot{q})=0$. In order to achieve the aforementioned goal, we have to make sure that the Lyapunov function of the system decrease at all time steps. The time derivative of $L(q,\ \dot{q})$ with respect to time is

$$\dot{L}(q,\ \dot{q})=-\left(E_{\text{top}}-E(q,\ \dot{q})\right)\dot{E}(q,\ \dot{q}), \tag{4.9}$$

and the time derivative of E with respect to time is

$$\begin{aligned}\dot{E}(q,\ \dot{q}) &= \dot{q}^T D(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \dot{q}^T G(q) \\ &= \dot{q}^T\left(\text{-}C(q,\ \dot{q})\dot{q} - G(q) + \tau\right) + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \dot{q}^T G(q) \\ &= \dot{q}^T\tau \\ &= \dot{x}F,\end{aligned} \tag{4.10}$$

which shows that the derivative of E is proportional to the product of the speed of the cart and input force. The time derivative of $L(q,\ \dot{q})$ with respect to time can be obtained from combing Eq. (4.9) and (4.10), which reads

$$\dot{L}(q,\ \dot{q}) = -\left(E_{\text{top}} - E(q,\ \dot{q})\right)\dot{x}F, \tag{4.11}$$

from which we can see that in order to make sure the Lyapunov function of the system $L(q,\ \dot{q})$ decrease at all time steps, the direction of the control force has to be coherent with the sign of $\left(E_{\text{top}} - E(q,\ \dot{q})\right)\dot{x}$. Hence, for the QPSO, following [57], a Lyapunov-based control law for the learning agent based on the Lyapunov analysis can be derived as follows:

$$F = sgn((E_{\text{top}} - E)\dot{x})u, \tag{4.12}$$

where $sgn(\dot{x}) = \{1 \text{ if } x \geq 0, \text{ and -1 otherwise}\}$ and u is the output force of the NFS limited in [-10,10]. Initial parameters of the QPSO and TSR-EA for controlling cart-pole balancing system are listed in the following two tables:

Table 4.1: The initial parameters of the QPSO for cart-pole balancing system.

Parameters	Value	Parameters	Value
$[\sigma_{\min}, \sigma_{\max}]$	[0, 2]	c_1	2.01
$[m_{\min}, m_{\max}]$	[0, 2]	c_1	2.01
$[w_{\min}, w_{\max}]$	[-20, 20]	s	50
R	4	ϕ	4.02
α	0.01	χ	0.99
γ	0.9	*max_gen*	300

Table 4.2 : The initial parameters of the TSR-EA for cart-pole balancing system.

Parameters	Value	Parameters	Value
$[\sigma_{\min}, \sigma_{\max}]$	[0, 2]	*Thres _TimeStep*	1000
$[m_{\min}, m_{\max}]$	[0, 2]	*Crossover Rate*	0.5
$[w_{\min}, w_{\max}]$	[-20, 20]	*Mutation Rate*	0.2
R	5	s	50
A	10	N_{cs}	250

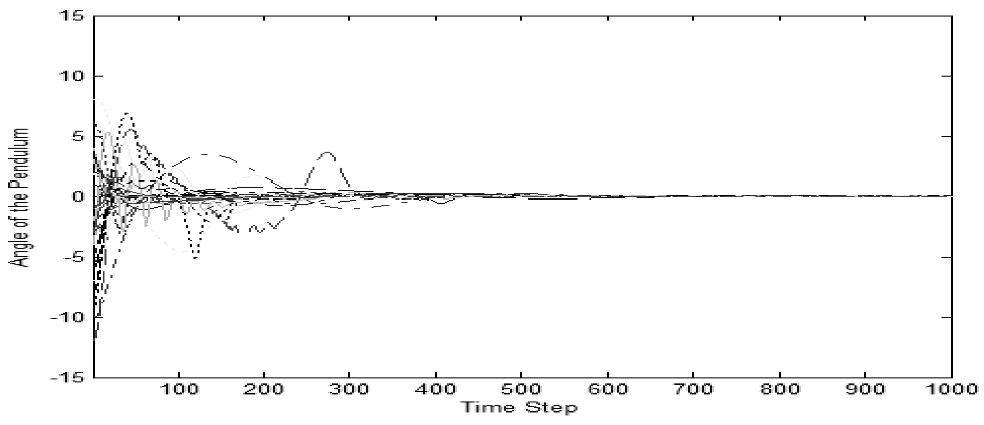
To verify with the performance of the QPSO, the TD and GA based reinforcement learning (TDGAR) [53], the on-line clustering and Q-value based GA reinforcement learning (CQGAF) [54] and the recurrent wavelet-based NFS with a reinforcement group cooperation-based symbiotic evolution (R-GCSE) algorithm [55] are applied to the same control task. In the TDGAR, there are five hidden nodes and five rules in the critic network and the action network. The population size is set as 200 and the maximum perturbation is set as 0.0005. In the CQGAF, after trial-and-error tests, the final average number of rules from 50 runs was 6 by using the on-line clustering algorithm. The population size is set as 50. The parameters for Q-learning are set as α=0.01 and γ=0.9. In the R-GCSE, the population size is set as 50 and the mutation rate is set as 0.1.

The control goal defined here is "bringing the plant's state to G_1 within 1,000 time steps." The original successful region *Original_Range* of the variables are $-12^\circ \le \theta \le 12^\circ$ and -2.4m$\le x \le$2.4m. The initial state of the plant is set within *Original_Range*. A trail ends when the control goal is met or a failure occurs. For, the QSPO, TDGAR, CQGAF and R-GCSE, a failure learning trial if the cart or the pendulum deviates beyond the *Original_Range*. For the TSR-EA, a failure learning trial occurs if the cart or the pendulum deviates beyond the *Original_Range* or the strict successful region defined in Eq. (3.7). The constraints of the output force is -10N$\le F \le$10N. If each algorithm is executed for 50 times to compute the average. The performances of all these compared methods are shown in Table 4.3, from which we can see that the QPSO and TSR-EA has superior control rate and requires fewer CPU-time cost. The reason could be due to the incorporating of the Lyapunov design principles in the QPSO, and the proposed TSR mechanism provides a more distinguishable performing index to the individuals that can accelerate their evolution process.

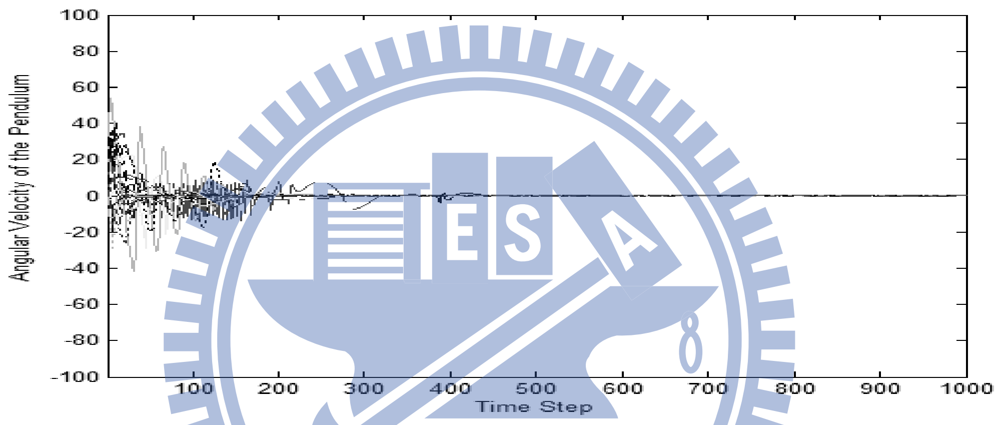
Table 4.3: Summary Statistics of Example 1.

Methods	**QPSO**	**TSR-EA**	TDGAR	CQGAF	R-GCSE
% of learning trials meet the control goal.	**100**	**96**	68	74	88
Average Time to goal.	**9.8 ± 0.7**	**12.2 ± 0.3**	80.2 ± 9.1	33.6 ± 2.7	58.9 ± 6.8

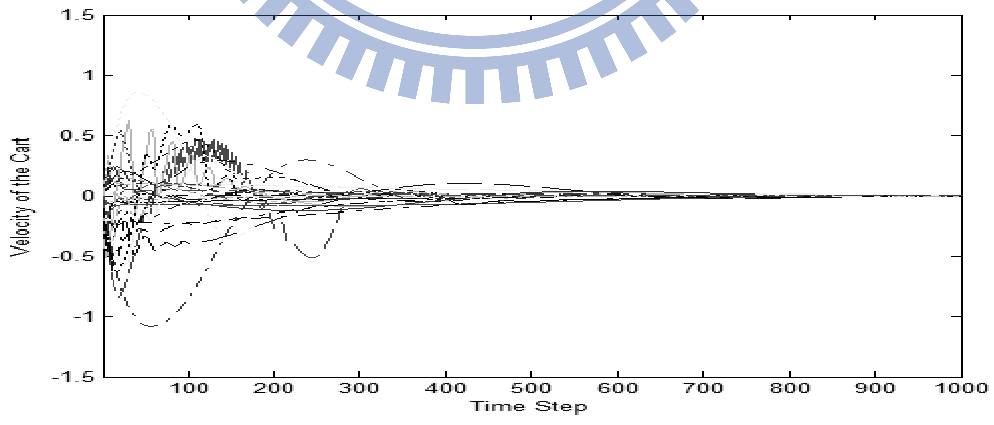
The testing results, which consist of the pendulum angle, pendulum angular velocity (in degrees/seconds), and cart velocity (in meters/seconds) of the TSR-EA, TDGAR, CQGAF and R-GCSE are shown in Fig. 4.2-4.5 as follows. Each line in Fig. 4.2-4.5 represents a single run.



(a)

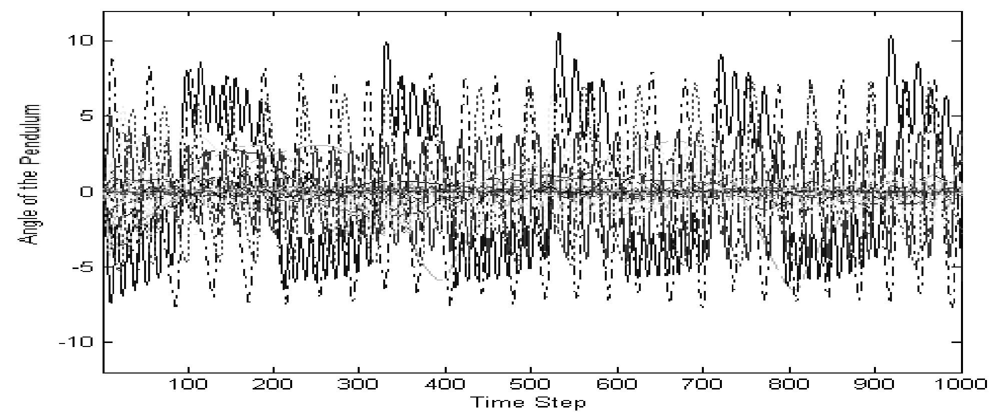


(b)

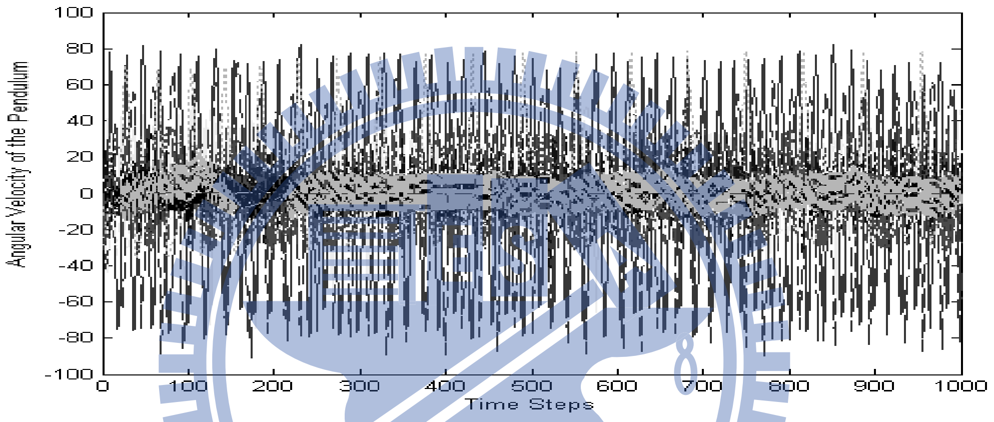


(c)

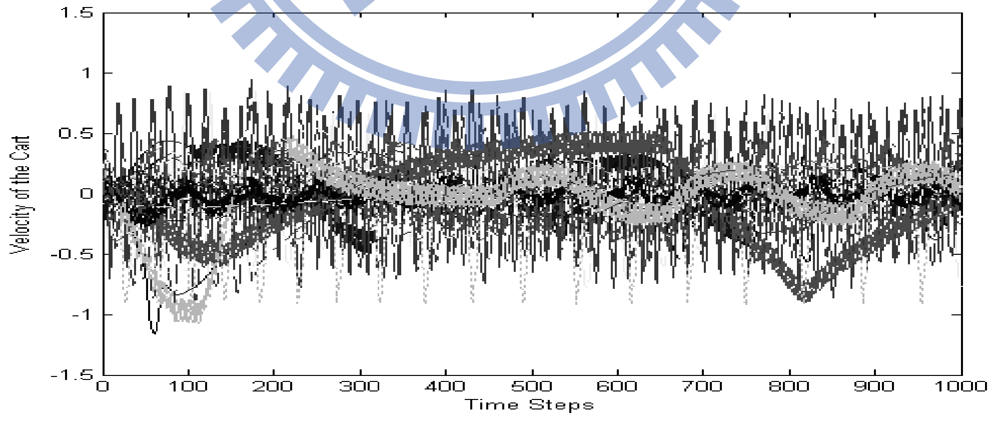
Figure 4.2: 50 control results of the cart-pole balancing system using the TSR-EA in Example 1. (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.



(a)

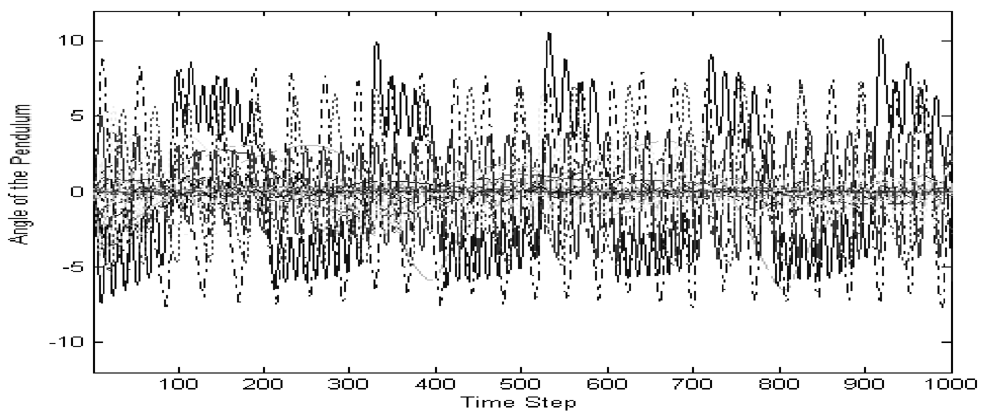


(b)

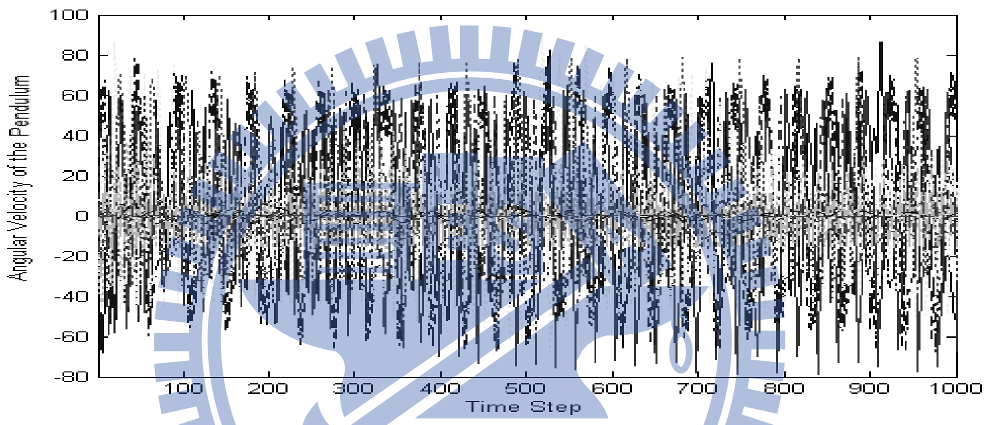


(c)

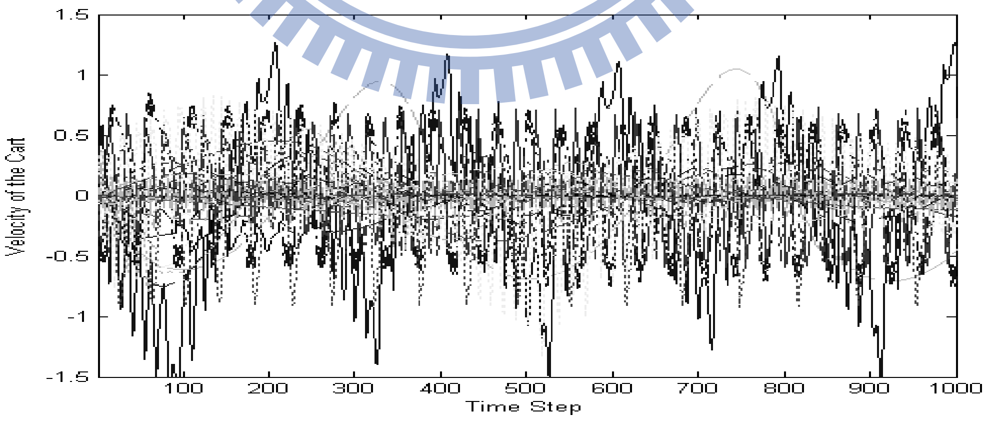
Figure 4.3: 50 control results of the cart-pole balancing system using the TDGAR in Example 1. (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.



(a)

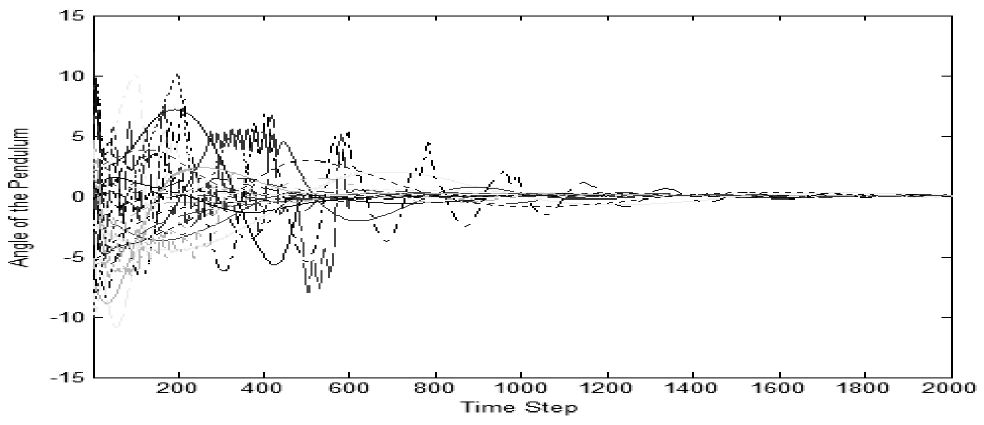


(b)

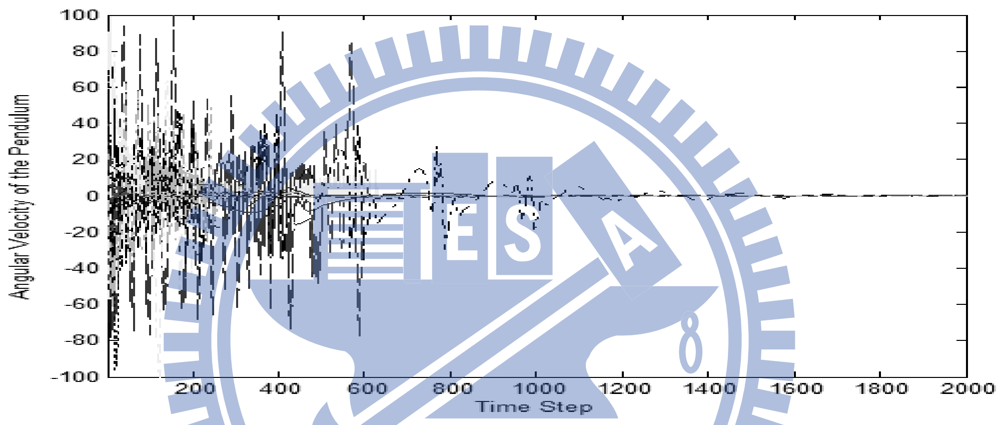


(c)

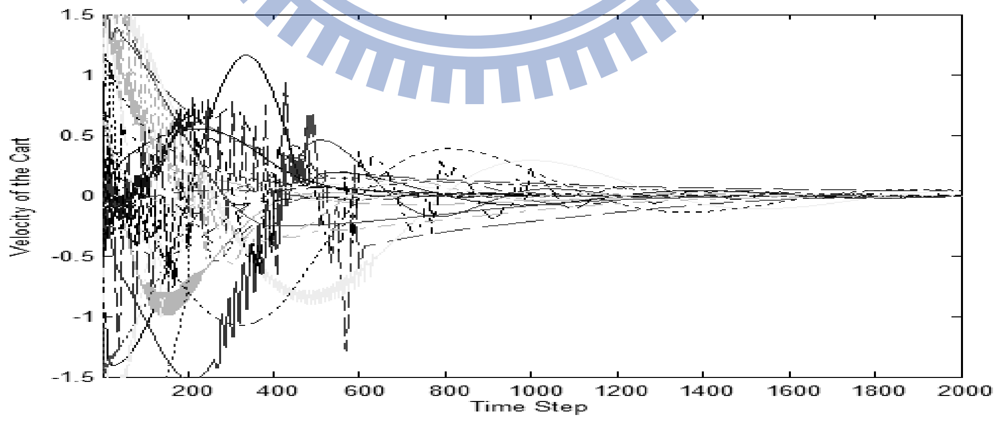
Figure 4.4: 50 control results of the cart-pole balancing system using the CQGAF in Example 1. (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.



(a)



(b)



(c)

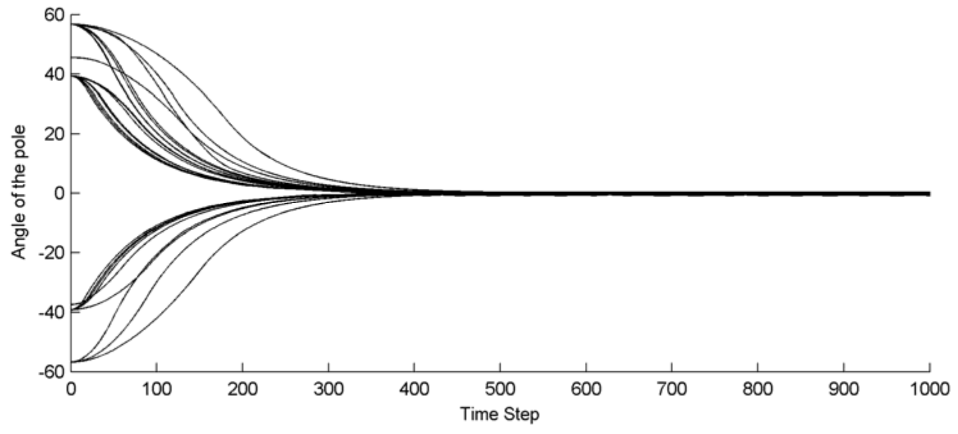
Figure 4.5: 50 control results of the cart-pole balancing system using the R-GCSE in Example 1. (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.

Furthermore, we complicate our control goal to "bringing the plant's state to G_1 within 5,000 time steps, and maintaining the state within G_1 for 100,000 time steps," and the original successful region *Original_Range* of the variables are modified to $-90^{\circ} \leq \theta \leq 90^{\circ}$ and -2.4m$\leq x \leq$2.4m. The initial state of the plant is set within *Original_Range*. A trail ends when the control goal is met or a failure occurs. For the QSPO, TSR-EA, TDGAR, CQGAF and R-GCSE, a failure learning trial occurs if the cart or the pendulum deviates beyond the *Original_Range*. Each algorithm is still executed for 50 times to compute the average. The performances of all these compared methods are shown in Table 4.4.

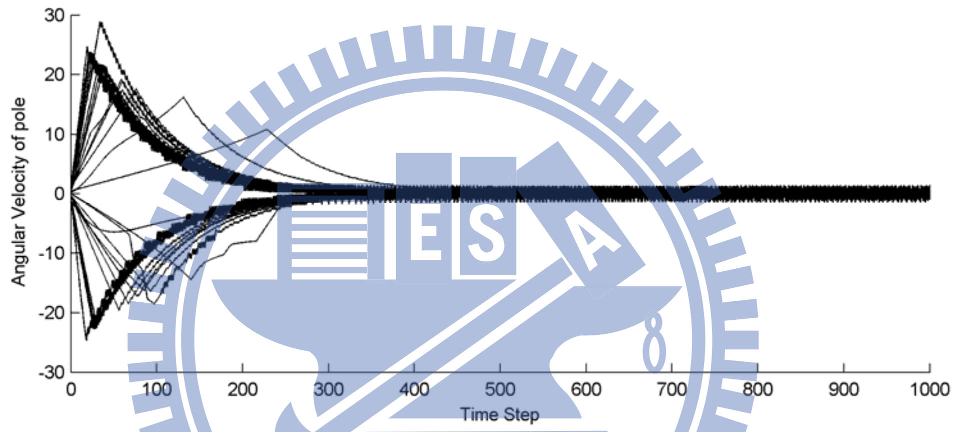
Table 4.4: Summary Statistics of Example 1 under a difficult control goal.

Methods	**QPSO**	TSR-EA	TDGAR	CQGAF	R-GCSE
% of first 10% trials meeting goal.	**92**	32	56	70	78
% of trials meeting goal.	**98**	94	84	90	94
Time to goal, first 10% trials.	**24.2** ± **0.8**	44.5 ± 6.6	200.2 ± 0	50.6 ± 7.2	78.9 ± 8.8
Average Time to goal.	**21.6** ± **0.3**	38.9 ± 2.5	169.8 ± 12.9	34.2 ± 6.1	46.1 ± 4.9

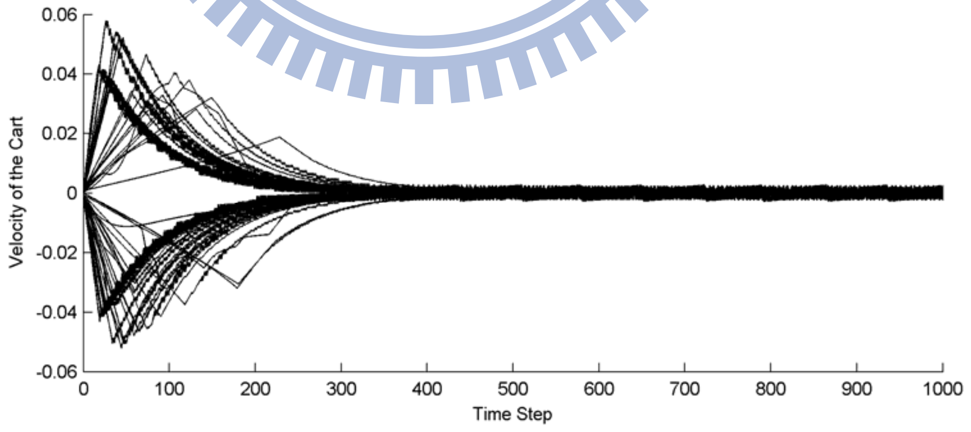
From Table 4.4 we can see that the QPSO has the most successful control rate. The superiority can be seen especially from the first 10% learning trials where learning agents are not fully trained yet. The QPSO is able to apply a safe, reliable control result during initial leanings, which is crucial important in many applications. The testing results of the QPSO are shown in Fig. 4.6 and Fig. 4.7. Each line in Fig. 4.6 and Fig. 4.7 represents a single run that starts form a increased range of initial states. Figure 4.6 shows the results the first 1,000 of 100,000 control time steps while Fig. 4.7 shows the last 1,000. From Fig. 4.6 we can see that with the aid of Lyapunov design, the QPSO is able to control the single-link inverted pendulum system well under different initial conditions. Trajectories shown in Fig. 4.7 verify the ability of the QPSO marinating the environment into G_1.



(a)

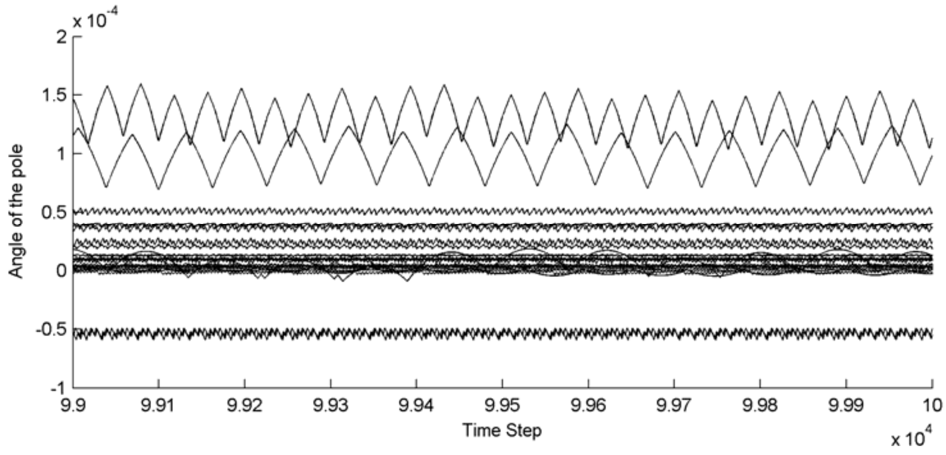


(b)

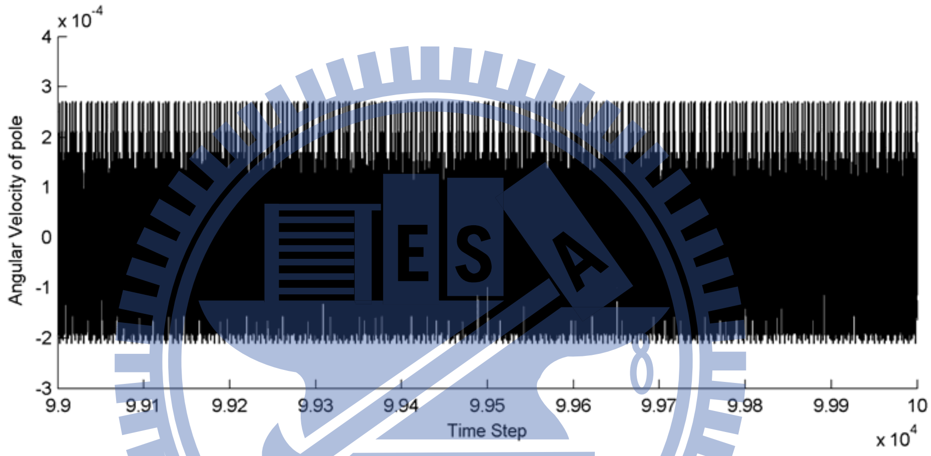


(c)

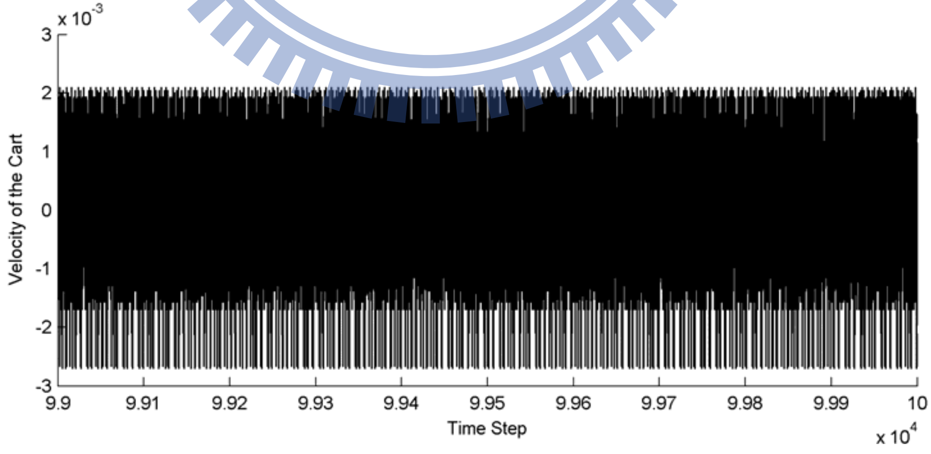
Figure 4.6: 50 first 1000 time steps control results the QPSO of the cart-pole balancing system. . (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.



(a)



(b)



(c)

Figure 4.7: 50 last 1000 time steps control results the QPSO of the cart-pole balancing system. . (a) Angle of the pendulum. (b) Angular velocity of the pendulum. (c) Velocity of the cart.

Example 2: Control of a double-link inverted pendulum system

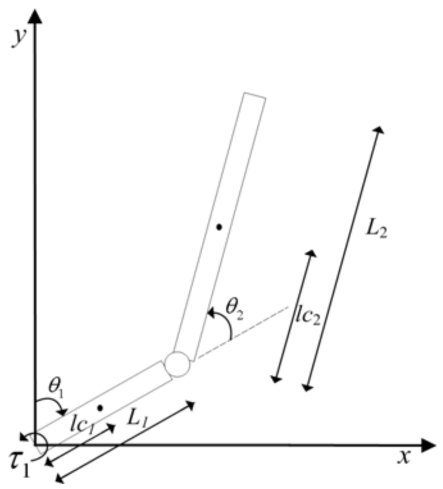


Figure 4.8: Double-link inverted pendulum system.

Consider the double-link inverted pendulum system: m_1 is the mass of link 1, m_2 is the mass of link 2, θ_1 is the angle that link 1 makes with the vertical, θ_2 is the angle that link 2 makes with link 1, l_1 and l_2 are the lengths of link 1 and 2, lc_1 is the distance of the center of mass of link 1, lc_2 is the distance of the center of mass of link 2, I_1 and I_2 are the moments of inertia of link 1 and link 2 about their centroids and τ_1 is the only control torque applied to the joint of link 1. We introduce the following five parameter equations:

$$\begin{cases} p_1 = m_1 {lc_1}^2 + m_2 l_1^2 + I_1 \\ p_2 = m_2 {lc_2}^2 + I_2 \\ p_3 = m_2 l_1 lc_1 \\ p_4 = m_1 lc_1 + m_2 l_1 \\ p_5 = m_2 lc_2 \end{cases} . \tag{4.13}$$

The model of the system can be obtained by using Lagrange’s method:

$$D(q)\ddot{q} + C(q,\ \dot{q})\dot{q} + G(q) = \tau , \tag{4.14}$$

where

$$q = [q_1,\ q_2]^T = [\theta_1,\ \theta_2]^T ,\quad \tau = [\tau_1,\ 0]^T , \tag{4.15}$$

$$D(q) = \begin{bmatrix} p_1 + p_2 + 2p_3 \cos q_2 & p_2 + p_3 \cos q_2 \\ p_2 + p_3 \cos q_2 & p_2 \end{bmatrix}, \tag{4.16}$$

$$C(q,\dot{q}) = p_3 \sin q_2 \begin{bmatrix} -\dot{q}_2 & -\dot{q}_2 - \dot{q}_1 \\ \dot{q}_1 & 0 \end{bmatrix}, \tag{4.17}$$

$$G(q) = \begin{bmatrix} p_4 \cos q_1 + p_5 g \cos(q_1 + q_2) \\ p_5 g \cos(q_1 + q_2) \end{bmatrix}. \tag{4.18}$$

The potential energy of the double-link inverted pendulum system can be defined as

$$P(q) = p_4 g \sin q_1 + p_5 g \sin(q_1 + q_2), \tag{4.19}$$

and the total mechanical energy of the system is given by

$$\begin{aligned} E(q,\ \dot{q}) &= \frac{1}{2}\dot{q}^T D(q)\dot{q} + P(q) \\ &= \frac{1}{2}\dot{q}^T D(q)\dot{q} + p_4 g \sin q_1 + p_5 g \sin(q_1 + q_2). \end{aligned} \tag{4.20}$$

The control objective is to stabilize the system around its top position, i.e. $(q_1, \dot{q}_1, q_2, \dot{q}_2)$=(0,0,0,0). Hence, another goal set is defined by

$$G_2 = \left\{ (q_1, \dot{q}_1, q_2, \dot{q}_2) :\ \left\| (q_1, \dot{q}_1, q_2, \dot{q}_2) \right\| \le 0.01 \right\}. \tag{4.20}$$

When the state of double-link inverted pendulum system is in G_2, the total mechanical energy E of the system is given by

$$E(0,\ 0,\ 0,\ 0) = E_{\text{top}} = (p_4 + p_5)g. \tag{4.21}$$

By defining a Lyapunov function $L(q,\ \dot{q}) = \frac{1}{2}\left(E_{\text{top}} - E(q,\ \dot{q})\right)^2$. The control objective can be either considered as guiding the system state into G_2 or achieving $L(q,\ \dot{q}) = 0$. The action selection of the QPSO is to make sure that the Lyapunov function of the system decrease at all time steps. The time derivative of $L(q,\ \dot{q})$ with respect to time is given by

$$\dot{L}(q,\ \dot{q}) = -\left(E_{\text{top}} - E(q,\ \dot{q})\right)\dot{E}(q,\ \dot{q}), \tag{4.22}$$

Where the time derivative of E with respect to time is

$$\dot{E}(q,\dot{q}) = \dot{q}^T D(q)\ddot{q} + \frac{1}{2}\dot{q}^T \dot{D}(q)\dot{q} + \dot{q}^T G(q)$$

$$= \dot{q}^T\left(-C(q,\dot{q})\dot{q} - G(q) + \tau\right) + \frac{1}{2}\dot{q}^T\dot{D}(q)\dot{q} + \dot{q}^T G(q) \quad (4.23)$$

$$= \dot{q}^T\tau \ = \dot{q}_1\tau_1,$$

which shows that the derivative of E is proportional to the product of the angular velocity of the first pole. The time derivative of $L(q,\dot{q})$ with respect to time is derived as follows:

$$\begin{aligned}\dot{L}(q,\dot{q}) &= -\left(E_{\text{top}} - E(q,\dot{q})\right)\dot{q}^T\tau \\ &= -\left(E_{\text{top}} - E(q,\dot{q})\right)\dot{q}_1\tau_1.\end{aligned} \quad (4.24)$$

In order to make sure the Lyapunov function of the system $L(q,\dot{q})$ decrease at all time steps, the direction of the control torque is assigned to be coherent with the sign of $\left(E_{\text{top}} - E(q,\dot{q})\right)\dot{q}_1$. A Lyapunov-based control law for the QPSO can be derived as follows:

$$\tau_1 = sgn((E_{\text{top}} - E)\dot{q}_1)z\,, \quad (4.25)$$

where z is the output of the NFS limited in [-10,10]. Double-link inverted pendulum system parameters are L_1=1m, L_2=2m, m_1=1kg, m_2=2kg, g=9.8m/s. In designing the NFS, the four controller input $(\theta,\dot{\theta},x,\dot{x})$ are normalized between 0 and 1, the output z is limited between -10 and 10. Initial parameters of the QPSO and TSR-EA for controlling two-pole inverted pendulum system are listed in the following two tables:

Table 4.5: The initial parameters of the QPSO for two-pole inverted pendulum system.

Parameters	Value	Parameters	Value
$[\sigma_{\min},\sigma_{\max}]$	[0, 2]	c_1	2.01
$[m_{\min},m_{\max}]$	[0, 2]	c_1	2.01
$[w_{\min},w_{\max}]$	[-30, 30]	s	50
R	5	ϕ	4.02
α	0.01	χ	0.99
γ	0.9	*max_gen*	300

Table 4.6 : The initial parameters of the TSR-EA for two-pole inverted pendulum system.

Parameters	Value	Parameters	Value
$[\sigma_{\min}, \sigma_{\max}]$	[0, 2]	*Thres_TimeStep*	5000
$[m_{\min}, m_{\max}]$	[0, 2]	*Crossover Rate*	0.5
$[w_{\min}, w_{\max}]$	[-30, 30]	*Mutation Rate*	0.2
R	7	s	50
A	10	N_{cs}	350

In the TDGAR, there are five hidden nodes and five rules in the critic network and the action network. The population size is set as 300 and the maximum perturbation is set as 0.0005. In the CQGAF, after trial-and-error tests, the final average number of rules from 50 runs was 8 by using the on-line clustering algorithm. The population size is set as 50. The parameters for Q-learning are set as α=0.01 and γ=0.9. In the R-GCSE, the population size is set as 50 and the mutation rate is set as 0.1.

For the TDGAR, CQGAF and R-GCSE, the original successful region of the variables is $-36^{\circ} \le \theta_1 \le 36^{\circ}$, and $-36^{\circ} \le \theta_2 \le 36^{\circ}$. Initial states of the plant are set within the original successful region. The control goal is defined to "maintaining the plant's state within G_2 for 100,000 time steps." A trail ends when the control goal is met or a failure occurs, which means that either pendulum deviates beyond the original successful region.

For the TSR-EA, the original successful region of the variables is $-36^{\circ} \le \theta_1 \le 36^{\circ}$, and $-36^{\circ} \le \theta_2 \le 36^{\circ}$. The strict successful region designed by the TSR is defined in Eq. (3.7). Initial states of the plant are set within the original successful region. The control goal is defined to "maintaining the plant's state within G_2 for 100,000 time steps." A trail ends when the control goal is met or a failure occurs, which means that either pendulum deviates beyond the either the original successful region or the strict successful region.

For the Q-PSO, the original successful region of the variables is $-90^{\circ} \le \theta_1 \le 90^{\circ}$, and

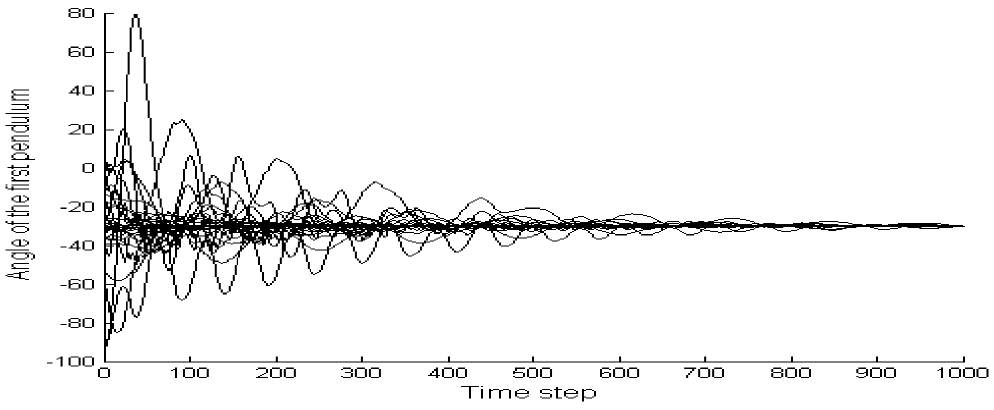
$-90^\circ \le \theta_2 \le 90^\circ$. Initial states of the plant are set within the original successful region, which represents the whole input space. The control goal is defined to "bringing plant's state to G_2 within 5,000 time steps and maintaining it within G_2 for 100,000 time steps." A trail ends when the control goal is met or a failure occurs, which means that it exceeds 105,000 time steps.

Each algorithm is executed for 50 times to compute the average. The performances of all these compared methods are shown in Table 4.7, from which we can see that the QPSO and TSR-EA has better control rate. The ability of the QPSO to provide reliable control result during initial learning is still obvious from control result of the first 10% learning trials.

Table 4.7: Summary Statistics of Example 2.

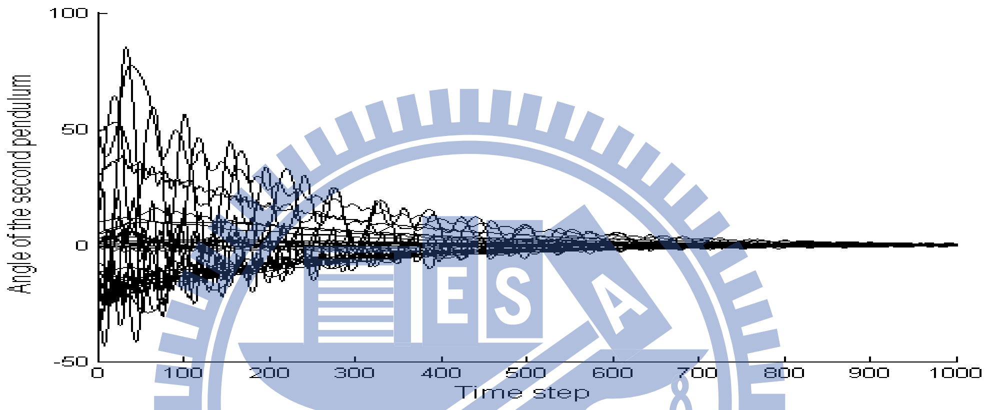
Methods	**QPSO**	**TSR-EA**	TDGAR	CQGAF	R-GCSE
% of first 10% trials meeting goal.	**86**	**14**	2	32	56
% of trials meeting goal.	**94**	**88**	46	68	82
Time to goal, first 10% trials.	**40.8 ± 1.9**	**66.3 ± 2.4**	308.2 ± 0	90.6 ± 7.2	145.9 ± 19.8
Average Time to goal.	**34.6 ± 2.2**	**57.7 ± 6.6**	276.8 ± 31.9	76.2 ± 13.1	131.7 ± 16.5

The testing results, which consist of the angle and angular velocity of both pendulums are shown in Fig. 4.9-4.13 as follows. Each line in Fig. 4.9-4.13 represents the first 1,000 control time steps of a single run.



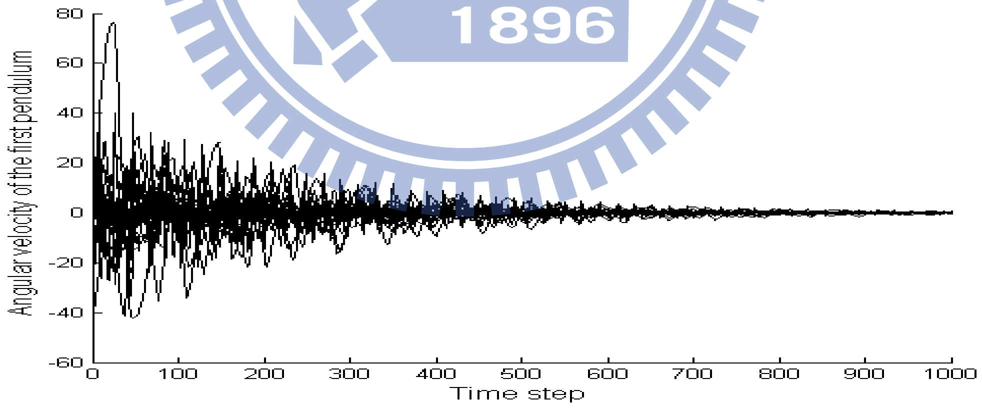
Angle of the first pendulum
Time step

(a)



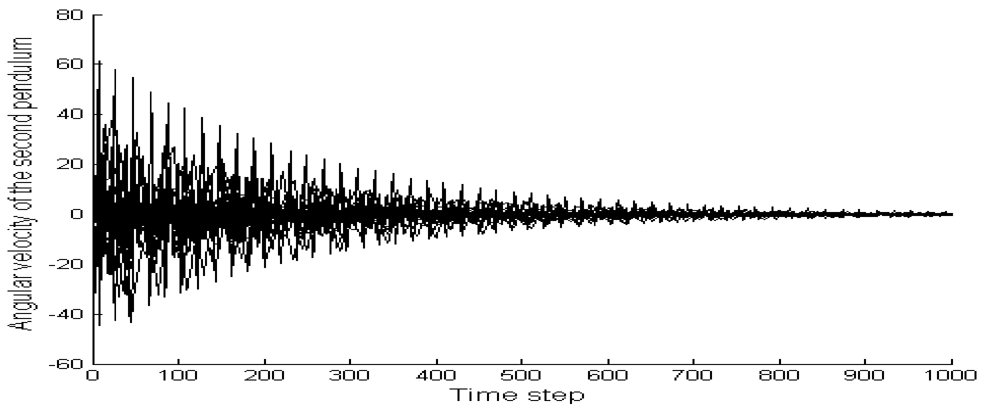
Angle of the second pendulum
Time step

(b)



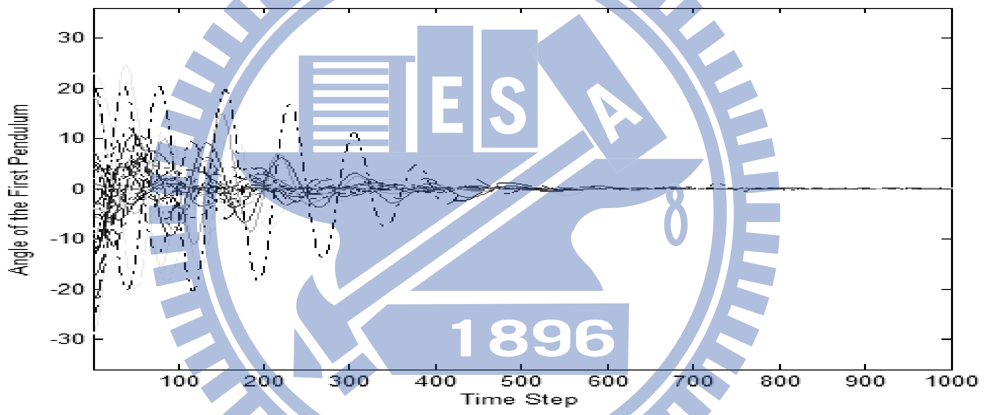
Angular velocity of the first pendulum
Time step

(c)

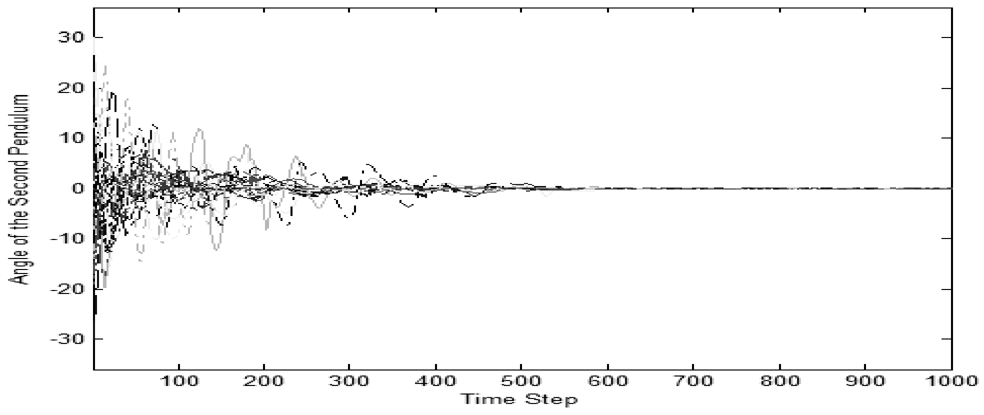


(d)

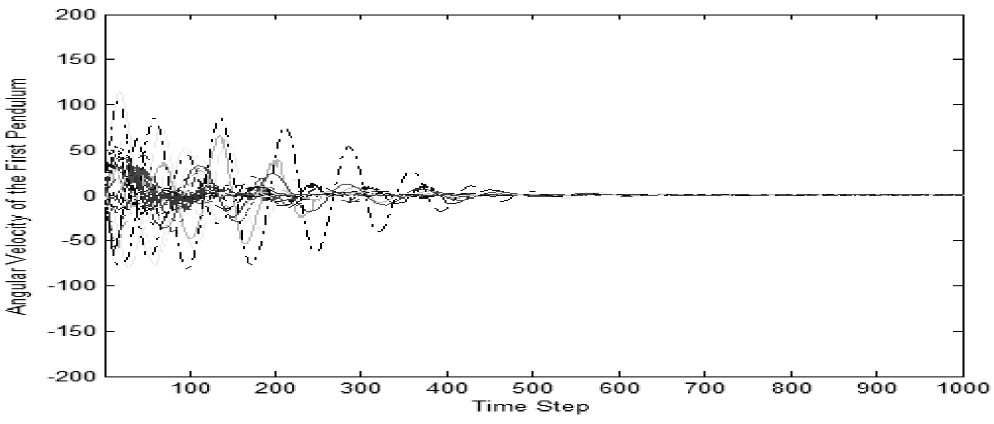
Figure 4.9: 50 first 1000 time steps control results of the double-link inverted pendulum system using the QPSO. (a) Angle of link 1. (a) Angle of link 2. (c) Angular velocity of link 1. (d) Angular velocity of link 2.



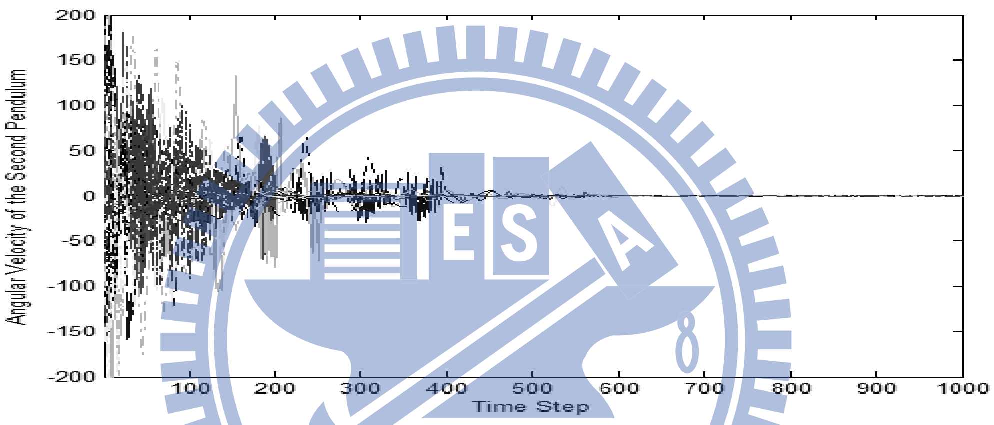
(a)



(b)

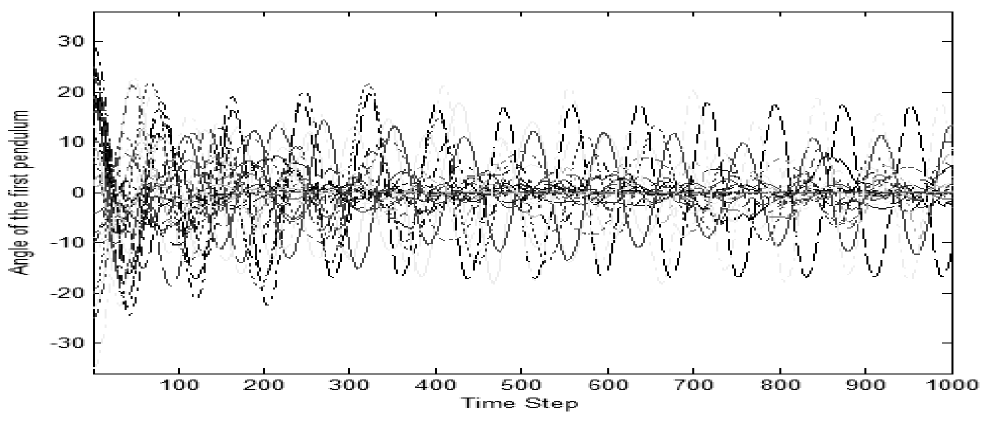


(c)

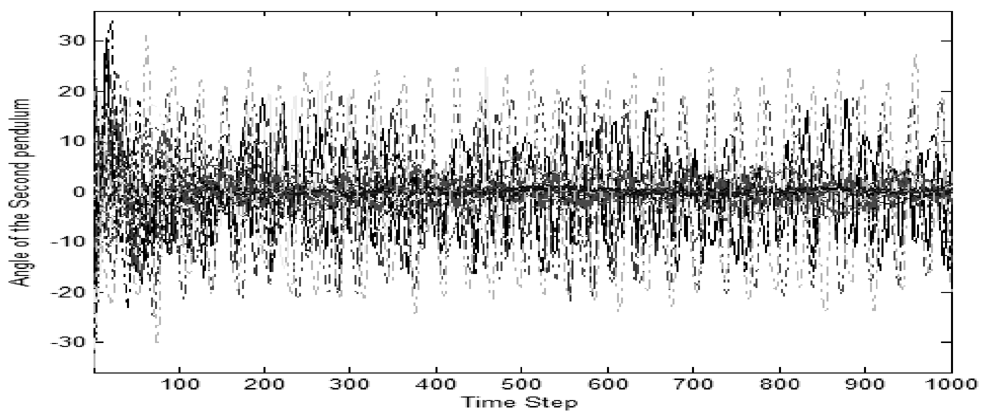


(d)

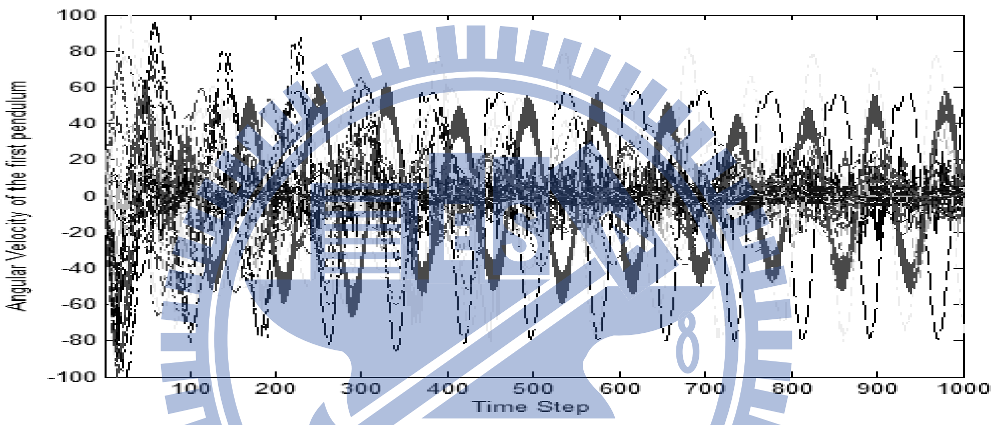
Figure 4.10: 50 first 1000 time steps control results of the double-link inverted pendulum system using the TSR-EA. (a) Angle of link 1. (a) Angle of link 2. (c) Angular velocity of link 1. (d) Angular velocity of link 2.



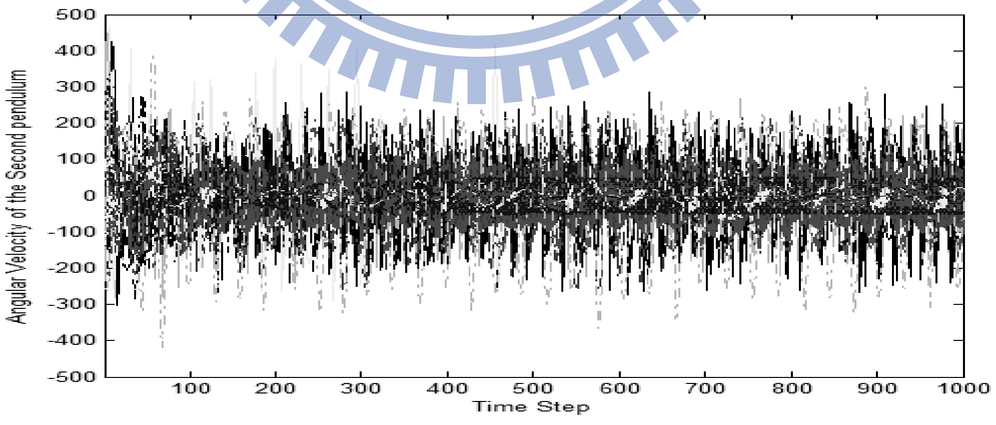
(a)



(b)

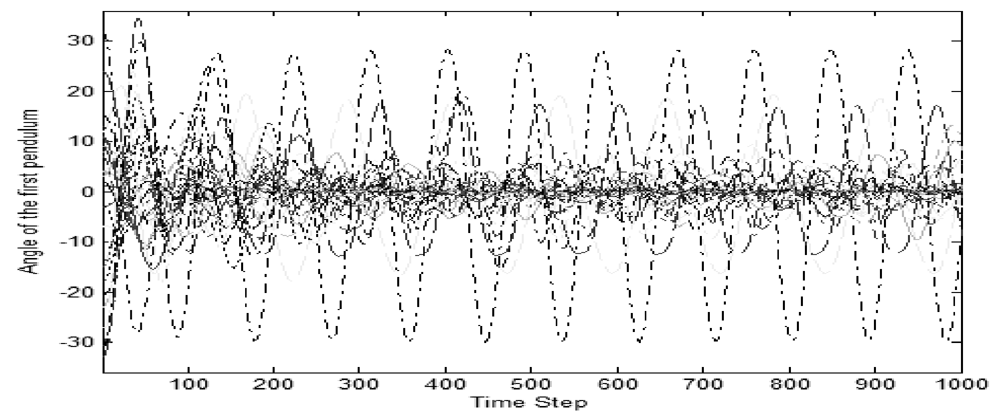


(c)



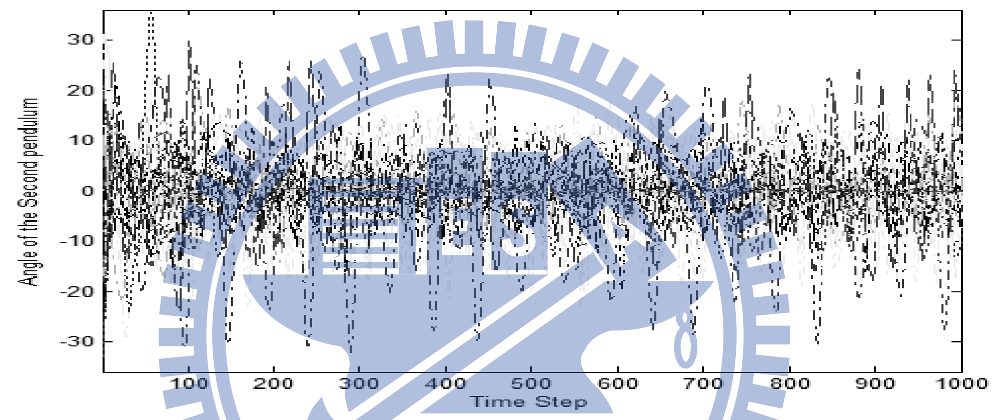
(d)

Figure 4.11: 50 first 1000 time steps control results of the double-link inverted pendulum system using the TDGAR. (a) Angle of link 1. (a) Angle of link 2. (c) Angular velocity of link 1. (d) Angular velocity of link 2.



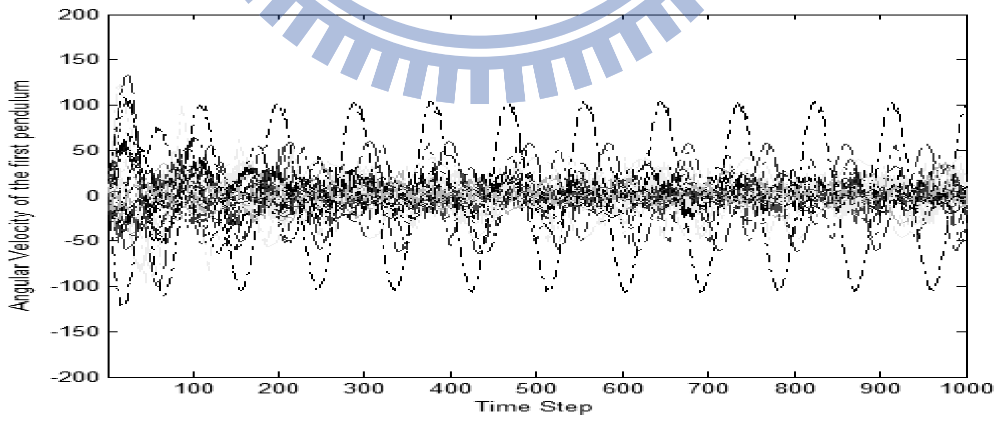
Angle of the first pendulum
30
20
10
0
-10
-20
-30
100
200
300
400
500
600
700
800
900
1000
Time Step

(a)



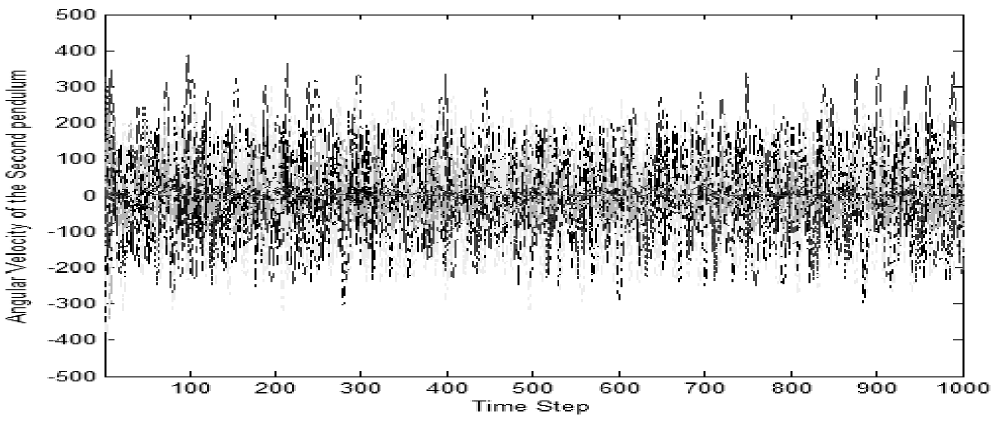
Angle of the Second pendulum
30
20
10
0
-10
-20
-30
100
200
300
400
500
600
700
800
900
1000
Time Step

(b)



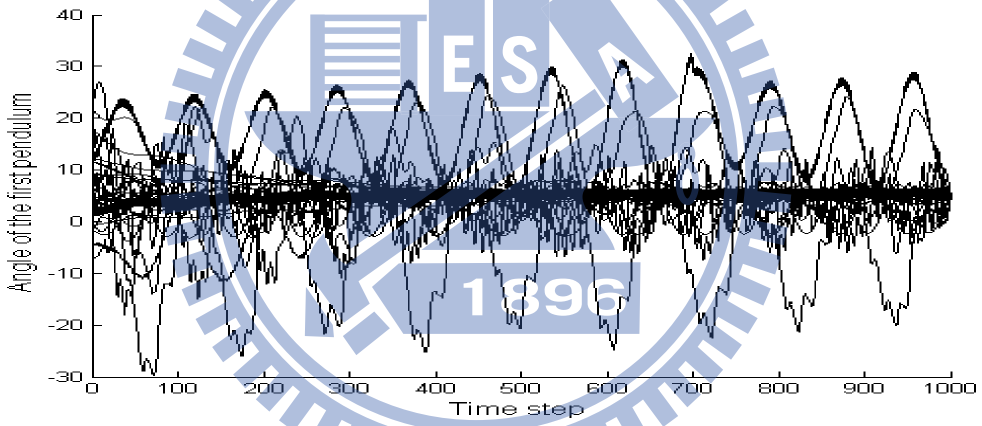
Angular Velocity of the first pendulum
200
150
100
50
0
-50
-100
-150
-200
100
200
300
400
500
600
700
800
900
1000
Time Step

(c)

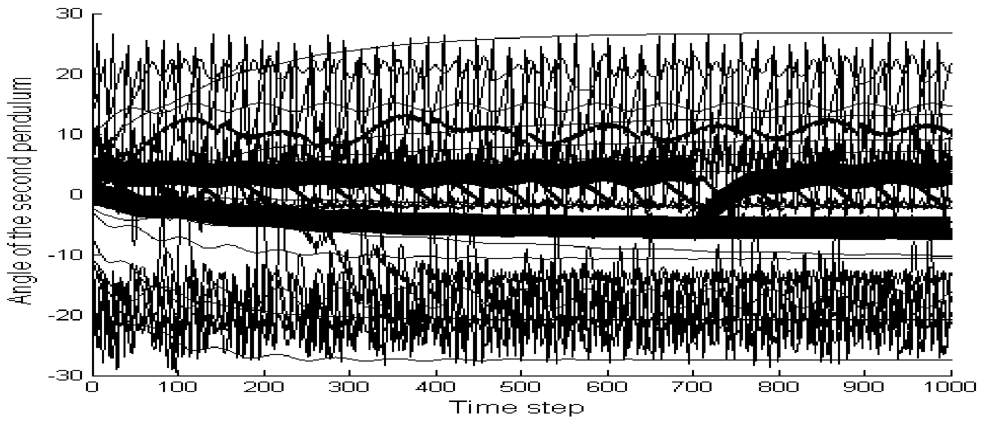


(d)

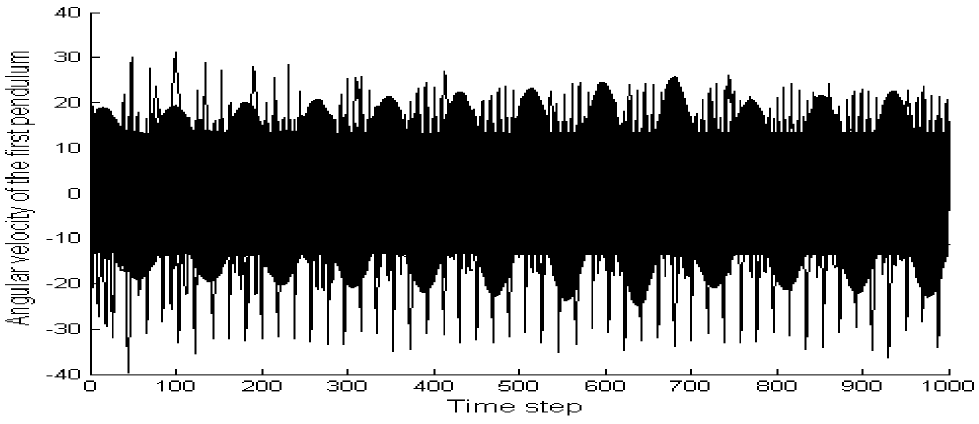
Figure 4.12: 50 first 1000 time steps control results of the double-link inverted pendulum system using the CQGAF. (a) Angle of link 1. (a) Angle of link 2. (c) Angular velocity of link 1. (d) Angular velocity of link 2.



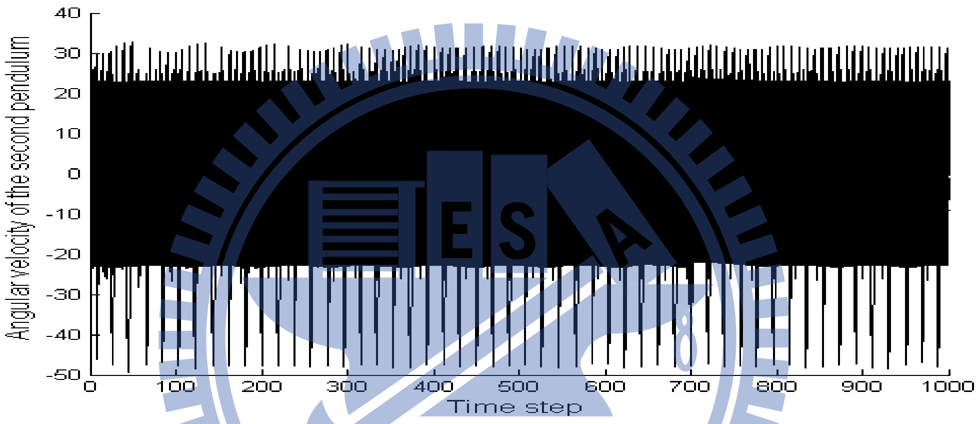
(a)



(b)



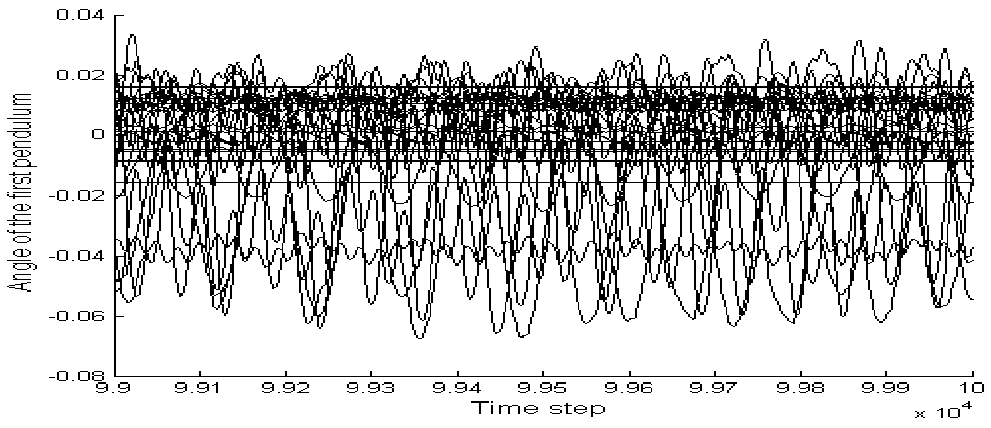
(c)



(d)

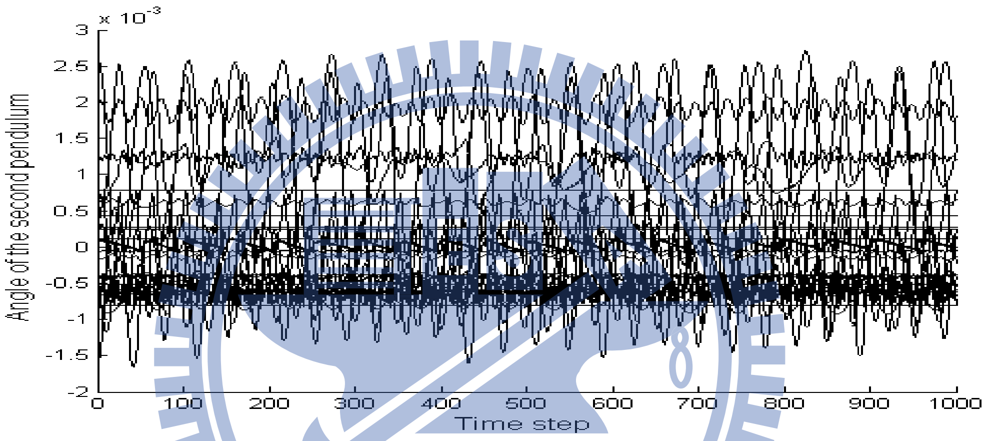
Figure 4.13: 50 first 1000 time steps control results of the double-link inverted pendulum system using the R-GCSE. (a) Angle of link 1. (a) Angle of link 2. (c) Angular velocity of link 1. (d) Angular velocity of link 2.

From Fig. 4.9-4.13 we can see that the proposed QPSO and TSR-EA have better control accuracy, which is one the major benefits of applying Lyapunov design principles or the TSR mechanism. The testing results of the last 1,000 control time steps of the QPSO and TSR-EA are shown in Fig. 4.14 and Fig. 4.15 as follows. From Fig. 4.14 and Fig. 4.15 we can see that, with two different kinds of mechanism, the QPSO and TSR-EA are able to attain accurate control results. Trajectories shown in Fig. 4.14 and Fig. 4.15 verify the ability of the QPSO and TSR-EA marinating their environment into G_2.



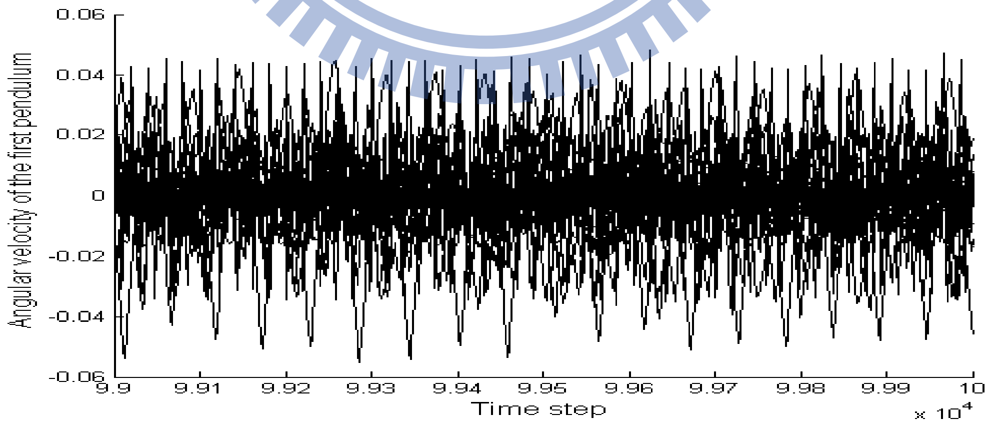
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
Angle of the first pendulum
9.9
9.91
9.92
9.93
9.94
9.95
9.96
9.97
9.98
9.99
10
Time step
x 10^4

(a)



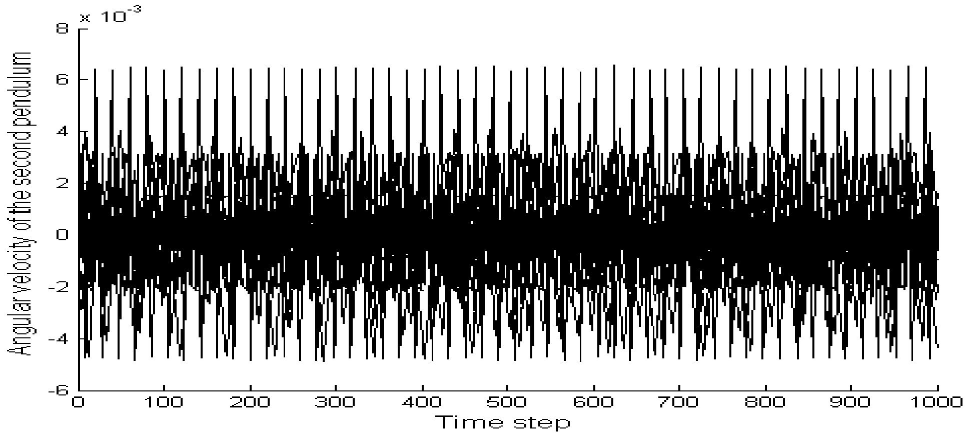
x 10^-3
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Angle of the second pendulum
0
100
200
300
400
500
600
700
800
900
1000
Time step

(b)



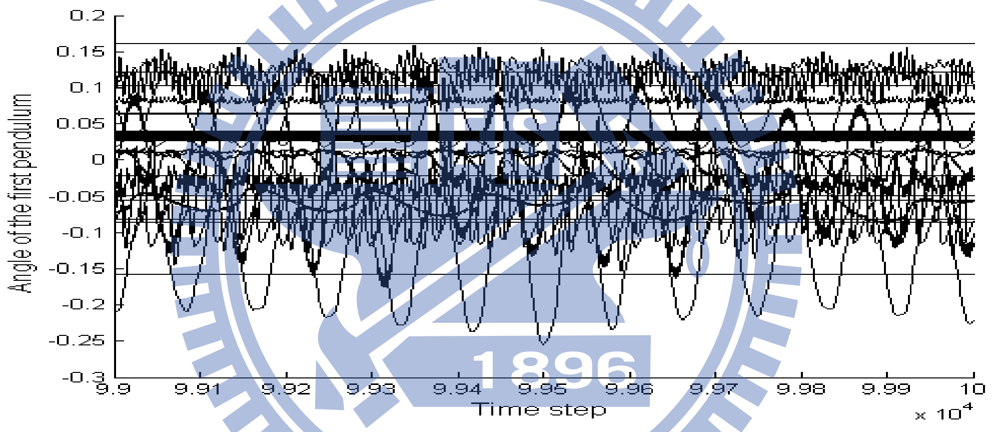
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
Angular velocity of the first pendulum
9.9
9.91
9.92
9.93
9.94
9.95
9.96
9.97
9.98
9.99
10
Time step
x 10^4

(c)

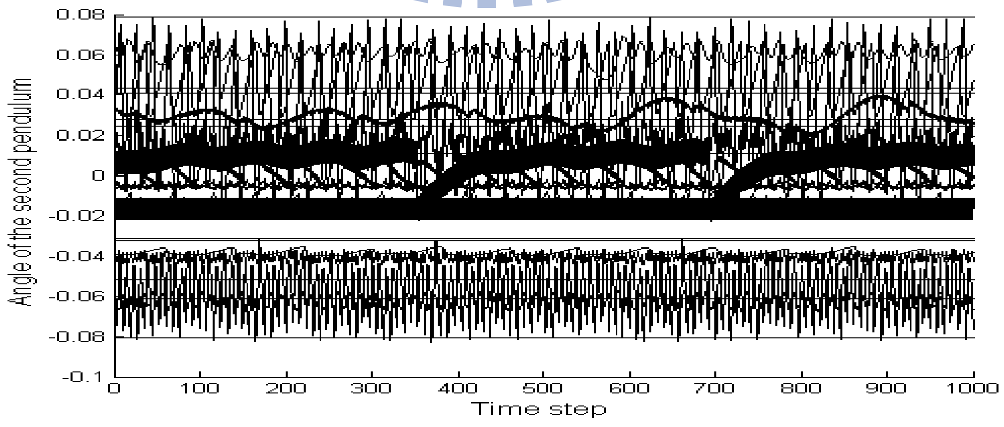


(d)

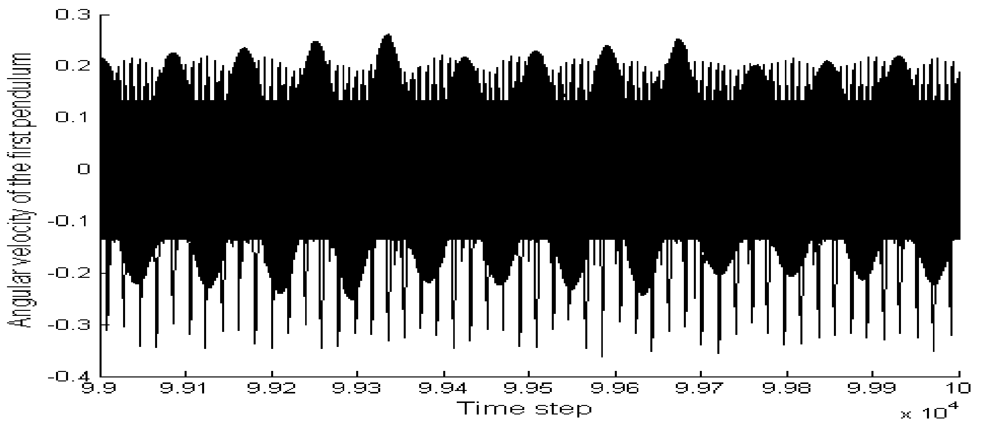
Figure 4.14: 50 last 1000 time steps control results of the double-link inverted pendulum system using the QPSO. (a) Angle of link 1. (b) Angular velocity of link 1. (c) Angle of link 2. (d) Angular velocity of link 2.



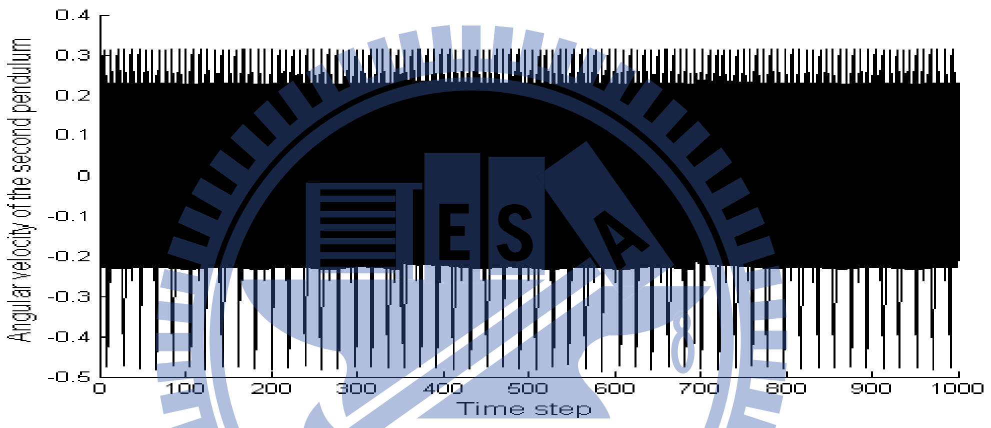
(a)



(b)



(c)



(d)

Figure 4.15: 50 last 1000 time steps control results of the double-link inverted pendulum system using the TSR-EA. (a) Angle of link 1. (b) Angular velocity of link 1. (c) Angle of link 2. (d) Angular velocity of link 2.

4.2 Real-valued Function Optimization Task

In this section, we will verify the performance of the proposed MS-CMA-ES and the SD-CPSO through real-valued function optimization task. In section 4.2.1 we introduce a simple computer simulation that illustrates the improvement of the MS-CMA-ES on global search ability, and the design of the environment for testing the MS-CMA-ES and other comparing algorithms. In section 4.2.2 we give computer simulation and comparison results that will state the improvement of the MS-CMA-ES over standard CMA-ES, SD-CPSO over standard PSO and CPSO on multi-funnel functions optimization.

4.2.1 Test Functions Introduction

The performance of the proposed MS-CMA-ES and SD-CPSO are verified by real-parameter minimization tasks, which contains totally nine test functions covering all types. By there nature they can be divided into two parts: unimodal and multi-modal functions. The first two functions are unimodal, followed by seven multimodal functions with three of them have simple global structures (single-funnel functions) and another four have complex global structures (multi-funnel functions). The types and names of functions are described in Table 4.8, and a detailed definition of test functions can be seen in [103, 104].

Table 4.8: Type and name of test functions.

Unimodal Functions
f_1: Sphere Function
f_2: High Conditioned Ellipsoidal Function
Multimodal Functions
f_3: Rosenbrock Function
f_4: Rastrigin Function
f_5: Griewank Function
Multi-Funnel Functions
f_6: Schwefel Function
f_7: Double-Rastrigin Function
f_8: Weierstrass Function
f_9: Michalewicz Function

First, we propose a simple computer simulation by executing both the MS-CMA-ES and the CMA-ES on 2-dim Double-Rastrigin function multiple times with there initial search points even distributed at the search space. We adjusted the selected Double-Rastrigin function to zero global optimum. Then we calculate on both algorithms the probability for each initial search location successfully finds the global optimum. The computer simulation process is shown as follows:

1. Initialize a set of initial search locations $X=\{x_1, x_2,\ldots, x_n\}$. Define the run times for each initial point N and the stopping criterion: maximum calculation times and minimum fitness threshold.

2. Execute the algorithm N times at initial point x_i and record the number of times $N_s(x_i)$ the algorithm successfully finds global optimum with initial search location x_i:

$$P_s(x_i) = N_s(x_i)/N \text{, for } i=1, 2,\ldots, n. \tag{4.26}$$

3. Calculate the average probability of success E_s:

$$E_s = \sum_{i=1}^{n} P_s(x_i) \Big/ n. \tag{4.27}$$

The contour details of Double-Rastrigin is shown in Fig. 4.16, from which we can see that there is a global optimum resting on the lower left corner, a local optimal solution resting on the upper right corner, and a spread of the noise-type local minima.

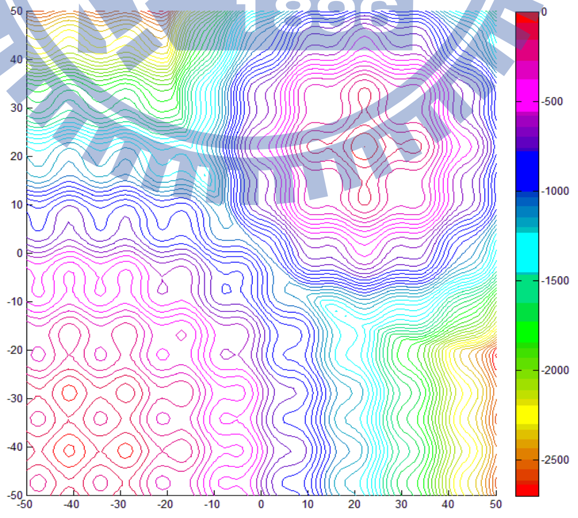
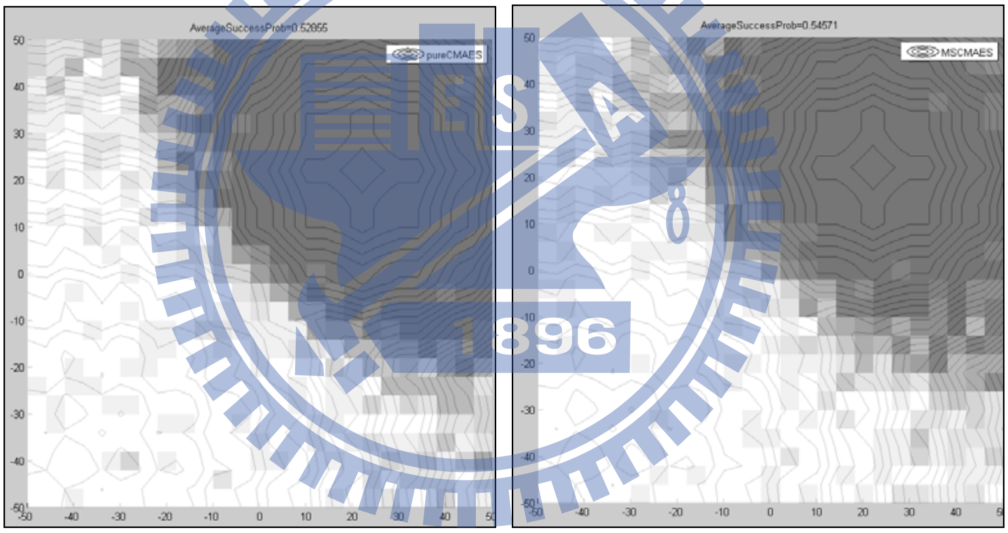


Figure 4.16: Contour details of double-Rastrigin function.

In this simulation, the run times for each initial point N is set as 20, and each run ends when the number of calculation times reaches 400, or when the function value of current search point declines to 0.01. The computer simulation result is shown in Fig. 4.17 which depicts the probability of success $P_s(x_i)$ of both algorithms. The search range for both algorithms are defined as $[-50,50]^2$, and the search space is discretized with 4x4 grid size as each initial search location. The color of each grid from dark to light corresponds to the value of $P_s(x_i)$ from 0 to 1. From Fig. 4.17 we can see that the white region of the MS-CMA-ES is larger than that of the CMA-ES especially in the mountain ridge part, which reveals the superior global search ability on the MS-CMA-ES. Improvement can also been seen from the average probability of success E_s of the MS-CMA-ES is 0.64571, which is larger than 0.52055 of the CMA-ES.



(a) (b)

Figure 4.17: Graph of global search ability test of (a) CMA-ES. (b) MS-CMA-ES.

4.2.2 Function Optimization Simulation

The problem dimension of the simulation is set 50. All functions have been adjusted to zero optimal solution respectively. The number of maximum fitness calculation times, initial search range, initial search position and minimum fitness threshold are detailed in Table 4.9.

Table 4.9: Parameters of the simulation.

	Number of maximum fitness calculation	Initial search range	Initial position	Minimum fitness threshold
f_1	10000	$\boldsymbol{x} \in [0,100]^d$	$\boldsymbol{x}=[50]^d$	1e-6
f_2	10000	$\boldsymbol{x} \in [0,100]^d$	$\boldsymbol{x}= [50]^d$	1e-6
f_3	10000	$\boldsymbol{x} \in [0,100]^d$	$\boldsymbol{x}= [50]^d$	1e-2
f_4	3000	$\boldsymbol{x} \in [0, 5]^d$	$\boldsymbol{x}= [2.5]^d$	1e-2
f_5	8000	$\boldsymbol{x} \in [0,600]^d$	$\boldsymbol{x}= [300]^d$	1e-2
f_6	4000	$\boldsymbol{x} \in [0,3]^d$	$\boldsymbol{x}= [1.5]^d$	1e-2
f_7	2000	$\boldsymbol{x} \in [-20,20]^d$	$\boldsymbol{x}= [0]^d$	1e-2
f_8	4000	$\boldsymbol{x} \in [0,0.5]^d$	$\boldsymbol{x}= [0.25]^d$	1e-2
f_9	5000	$\boldsymbol{x} \in [0,5]^d$	$\boldsymbol{x}= [2.5]^d$	1e-2

One half of the initial search range is defined as the initial standard deviation of CMA-ES and MS-CMA-ES, and the initial particles of PSO are evenly distributed in the initial search range. The proposed MS-CMA-ES and SD-CPSO are based on traditional CMA-ES and CPSO respectively. As a result, the MS-CMA-ES is compared with the standard CMA-ES and two of its famous improvements, a local restart CMA-ES (LR-CMA-ES) [60] and a CMA-ES with iteratively increasing population size (IPOP-CMA-ES) [61]. As to the SD-CPSO we compare it with standard PSO and comparing algorithms of this computer simulation include traditional CMA-ES and , PSO [8] and CPSO-S [9]. As to the parameter setting of participant algorithms, the parameter setting that the PSO and CPSO use refers to previous research [105]; The setting of parameters of CMA-ES is designed by [12]; MS-CMA-ES algorithm use the same parameters as CMA-ES except that the number of sample size is 1.5 times larger to the CMA-ES, and the parameter c_α introduced in the MS-CMA-ES is set to be 0.1. Table 4.10 outlined the computer simulation parameters. The formulas of parameters are listed below. The computer simulation data is obtained by executing each 50 dimensional test functions until the stopping criterion is met. The procedure was repeated 50 times to compute the average fitness value. In the paper, instead of the actual numeric fitness value, the rank of the minimum average fitness value is defined as the standard of comparison. The reason is that we want to exclude the impact of the different degree of scale on the raw numeric difference between each test function. For example, some functions have very large fitness gap between the best and the second best local minimum, some of them don't even have local minima. Therefore, the numeric difference may not be a good performing index for evaluating algorithms.

Table 4.10: MS-CMA-ES and CMA-ES parameters.

Parameters of Selection operator	
CMA-ES $\lambda = 4 + \lfloor 3\ln n \rfloor$	MS-CMA-ES $\lambda = 1.5(4 + \lfloor 3\ln n \rfloor)$
$w_i = \frac{w_i'}{\sum_{j=1}^{\lambda} w_j'},\ w_i' = \begin{cases} \ln(\frac{\lambda}{2}+0.5) - \ln i\ , \text{for i=1,...} \left\lfloor \frac{\lambda}{2} \right\rfloor \\ 0\ , \text{for else i} \end{cases},$	
Parameters of Covariance adaptation:	
c_{cov}=0.7	
μ_{cov}=10	
$c_c = \frac{4+\mu_{\mathrm{eff}}/n}{n+4+2\mu_{\mathrm{eff}}/n},$	
$c_\sigma = \frac{\mu_{\mathrm{eff}}+2}{n+\mu_{\mathrm{eff}}+5}$	
$d_\sigma = 1 + 2\max\left(0\ ,\ \sqrt{\frac{\mu_{\mathrm{eff}}-1}{n+1}} - 1\right) + c_\sigma$	
MS-CMA-ES mixed weighting:	
c_α=0.1	
Parameters of PSO:	
c_1=c_2=2.01	
s=50	
Parameters of SD-CPSO:	
ρ_{thres}=0.8	

The comparison result of the proposed MS-CMA-ES is shown in Table 4.11-4.13. In order to verify the performance of the MS-CMA-ES, we think it is important to evaluate the algorithms at early, middle, and the late stage of the test. So in this computer simulation we take three check points, at 20%, 50%, and 100% of the number of the maximum fitness calculation for rank comparison. The early, middle and later stage comparison results of the MS-CMA-ES are shown in Table 4-6 as follows.

Table 4.11: Average fitness at 20% number of fitness calculations.

	CMA-ES	**MS-CMA-ES**	LR-CMA-ES	IPOP-CMA-ES
f_1	1.380e-021(1)*	**1.918e-009(3)**	7.665e-017(2)	2.390e-008(4)
f_2	0.004611(2)	**1145(4)**	1.660e-010(1)*	0.009175(3)
f_3	51.02(1)*	**202.8(4)**	76.98(2)	99.09(3)
f_4	13.27(1)*	**21.64(2)**	33.87(4)	25.88(3)
f_5	0.06198(1)*	**0.01962(2)**	0.3861(3)	0.7785(4)
f_6	171 (2)	**516.2 (4)**	391.8(3)	139.9(1)*
f_7	13.95(1)*	**65.54(3)**	65.4(2)	108.5(4)
f_8	0.2643(1)*	**0.7184(4)**	0.4763(2)	0.5725(3)
f_9	5.75(3)	**33.65(4)**	1.254e-001(1)*	0.36(2)

Table 4.12: Average fitness at 50% number of fitness calculations.

	CMA-ES	**MS-CMA-ES**	LR-CMA-ES	IPOP-CMA-ES
f_1	1.512e-058(1)*	**1.335e-028(4)**	8.877e-049(3)	8.443e-050(2)
f_2	2.716e-040(2)	**6.94e-013(3)**	1.408e-044 (1)*	1.646e-010(4)
f_3	0.9815(2)	**2.643(1)***	21.68(3)	79.85(4)
f_4	9.754(2)	**8.649(1)***	11.9(3)	15.09(4)
f_5	0.06198(2)	**0.04725(1)***	0.1451(4)	0.06753(3)
f_6	169.9(4)	**144.2(2)**	87.78(1)*	119.7(3)
f_7	12.57(2)	**10.17(1)***	55.82(3)	79.74(4)
f_8	0.1199(1)*	**0.134(2)**	0.7343(3)	0.7444(4)
f_9	5.75(4)	**7.864e-008(1) ***	6.408e-003(3)	5.983e-007(2)

Table 4.13: Average fitness at 100% number of fitness calculations.

	CMA-ES	**MS-CMA-ES**	LR-CMA-ES	IPOP-CMA-ES
f_1	1.311e-120(1)*	**2.632e-062(3)**	7.854e-105(2)	8.443e-017(4)
f_2	4.489e-103(1)*	**3.478e-046(4)**	2.043e-097(2)	1.821e-056(3)
f_3	0.7862 (1)*	**0.8434 (2)**	18.85 (3)	82.45(4)
f_4	9.751(2)	**7.721(1)***	11.86(2)	15.09(2)
f_5	0.06198(2)	**0.03893(1)***	0.3769(3)	0.3861(3)
f_6	169.9(4)	**69.38(2)**	87.78(2)*	66.21(1)*
f_7	12.57(4)	**6.652e-003(1)***	8.98(2)	11.76(4)
f_8	0.1188(4)	**5e-004(1)***	0.06875 (3)	3.876e-003(2)
f_9	5.75(4)	**7.864e-008(1) ***	6.326e-003(3)	5.983e-007(2)

The results to be discussed are divided into three parts in accordance with the function types:

1) Unimodal Function:

Under the sphere function f_1, CMA-ES has the best performance, owing to its property of rapid convergence. As to ellipsoid function f_2, at first, LR-CMA-ES is better than the others, but worse than CMA-ES at the end. The reason the MS-CMA-ES has the worst performance may be that its clustering mechanism generates too many components on such simple unimodal functions. But from the optimization result, all three algorithms are capable of finding optimal solution within short times of fitness calculation.

2) Multimodal Function:

The MS-CMA-ES is better than other algorithms at the later stage under the f_4 and f_5 test functions except for f_3, f_4 and f_5 have single-funnel and noisy-like local minimums; however, f_3 doesn't have obvious single-funnel structure. From the optimization result we can see that the MS-CMA-ES is suitable of solving multimodal function optimization tasks.

3) Multi-Funnel Function:

In this dissertation, we focus on the optimization of this type of function. At early test stage, the other three algorithms outperform the MS-CMA-ES algorithm. The mechanism of the MS-CMA-ES is designed to generate multiple CMA-ES instances for exploring different regions of search space simultaneously. As a result, the reason the MS-CMA-ES loses at early stage may due to the scattering of sampling resources for finding the optimal solution in parallel. However, at the later stage, especially on the f_7 and f_8 functions, the global solution search capability of the MS-CMA-ES is beyond those of other comparing methods. The overall convergence rate of the MS-CMA-ES is its most obvious shortcomings due to the adopted parallel searching mechanism, but it is inevitable cost for improving the global searching ability on multi-funnel functions.

The comparison result of the proposed SD-CPSO is shown in Table 4.14 as follows.

Table 4.14: Average fitness value

	CPSO-S	**SD-CPSO**	PSO
f_1	6.361e-99(1)*	**2.634e-062(3)**	9.653-76(2)
f_2	4.481e-84(1)*	**3.464e-033(3)**	2.876e-75(2)
f_3	18.8764 (3)	**0.8872 (1)***	1.4356(2)
f_4	11.871(1) *	**17.721(2)**	26.65(3)
f_5	9.6198(3)	**0.6893(1)***	6.3769(2)
f_6	469.9(3)	**288.3(2)**	87.36(1)*
f_7	12.57(2)	**7.659(1)***	95.03(3)
f_8	1.2287(2)	**0.6643(1)***	1.254(3)
f_9	5.75(3)	**0.897(1) ***	4.08(2)

The results to be discussed are divided into three parts in accordance with the function types:

1) Unimodal Function:

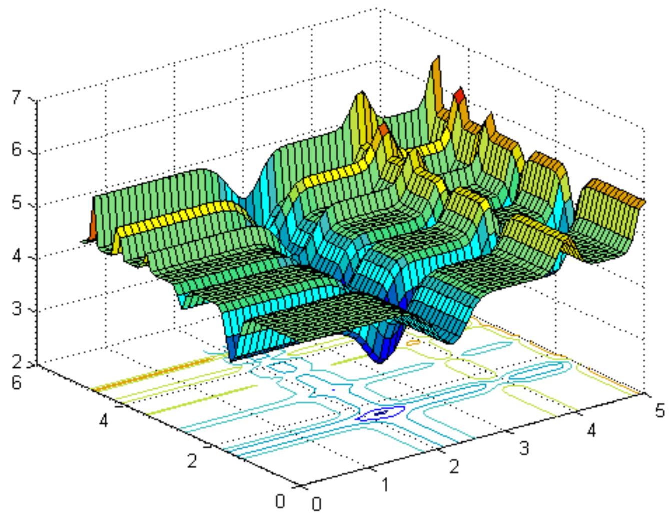
Under the sphere function f_1, CPSO-S has the best performance, owing to its property of rapid convergence. As to ellipsoid function f_2, at first, PSO is better than the other two algorithms. As shown from the computer simulation result, all three algorithms are capable of solving unimodal optimization task, and no improvement of performance can be found by applying our method.

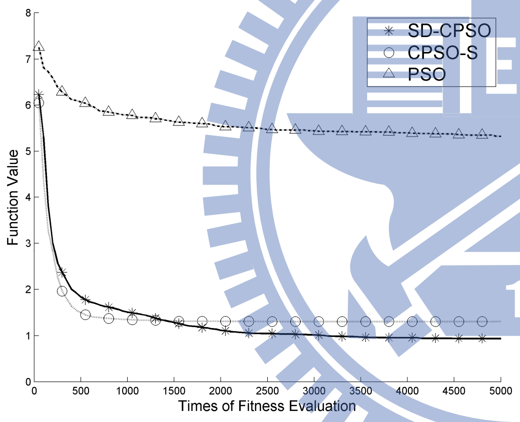
2) Multimodal Function:

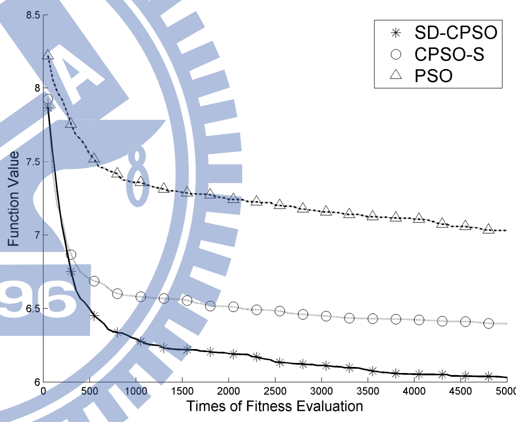
The SD-CPSO is better than other algorithms under the f_3 and f_5 test functions except for f_4, the Rastrigin’s function. We think it might due to the fact that Rastrigin’s function is nearly the same after rotation, which makes our effort trying to find a special trend to the global optimum irrelevant. However, the superiority of the proposed SD-CPSO in finding global optima of multimodal functions can be seen in substance.

3) Multi-Funnel Function:

From Table 4.14 we can see that in coping with multi-funnel function optimization tasks, the superiority of the proposed SD-CPSO is obvious. In general, the optimization of multi-funnel function is difficult as we can see especially from the optimization result of the f_6 function. Despite the proposed SD-CPSO has better performance on the optimization tasks of f_7 and f_8 function, the improvement is not very obvious. However, in the optimization of f_9, the Michalewicz's function, the improvement is remarkable. A visualization of a 2-D Michalewicz's function is shown in Fig. 4.18. We will illustrate the optimization results of applying Michalewicz's function in both its unrotated and rotated form in Fig. 4.19. Figure 4.19(a) represents the result of applying unrotated Michalewicz's function. Michalewicz's function introduces many valleys into the plain, and the function values for points in the space outside the narrow valleys give very little information about the location of the global optimum. Thus, the swarms need to follow through these valleys to find minimums. In its rotated version, these narrow valleys are too correlated to follow through from the perspective of the CPSO. In Fig. 4.19(b), the SD-CPSO in evidence overcomes the drawback.


Fugure 4.18: Visualization of a 2-D Michalewicz's function.





(a) (b)

Figure 4.19: Computer simulation results of applying Michalewicz's function in its (a) unrotated form, (b) rotated form.

CHAPTER 5
CONCLUSION

In this dissertation, four algorithms are proposed, including a Q-valued based particle swarm optimization (QPSO), a two-strategy reinforcement evolutionary algorithm (TSR-EA), a mean shift based evolution strategy with covariance matrix adaption (MS-CMA-ES) and a separability detection approach to cooperative particle swarm optimization (SD-CPSO). In this dissertation, the performance of the QPSO and TSR-EA are verified through reinforcement learning control tasks while the performance of the MS-CMA-ES and SD-CPSO are verified through real-valued function optimization tasks. Advantages and future works on these algorithms are described as follows.

The proposed QPSO adopts the concept of Lyapunov design for constructing safe reinforcement learning agents. The advantages of the QPSO can be shown from that it provides a reliable initial learning performance and accurate control result due to the Lyapunov design of learning agents. But one drawback of the QPSO is that it requires additional priori knowledge. In order to apply Lyapunov-based control laws, we have to identify the Lyapunov function of a plant first; furthermore, conventionally during the learning phase, we also requires more information about the system’s state, which may be difficult or too costly to access. The TSR-EA provides an alternative to attain accurate control result by the TSR mechanism. It requires less prior knowledge about the control plant compared with the QPSO. By simply shrinking the operating range of a control system as time step increases, the TSR mechanism can help learners to obtain an accurate control results on one hand and improves the learning rate on the other. Besides, the usage of the TSR is not limited by the TSR-EA algorithm. It is simply an design of reinforcement learning signal, so it is applicable to all time-step fashioned reinforcement learning. Another advantage of the

TSR-EA can be shown from adopting the group-based symbiotic evolution (GSE) to evaluate the fuzzy rule on a NFS locally.

One advantage of the MS-CMA-ES lies in improving the mutation mechanism of the traditional CMA-ES. The mutation mechanism of traditional CMA-ES is based on its self-adaption behavior. Despite good mutation directions can be determined by moderately self-adapting the tactic parameters of the CMA-ES, the mutation is still limited by normal distribution sampling. In the MS-CMA-ES, we propose a group mutation mechanism, adopting the concept that sampling from mixture probability yields larger flexibility. Search points sampled from mixture probability model on multiple directions can diminish the restriction of the local search. Another advantage of the MS-CMA-ES can be seen from adopting the mean shift-based clustering method for applying multiple CMA-ES instances to search the space in parallel. The parallel search mechanism can enhance the global search ability of the CMA-ES. Only one extra parameter compared to the original CMA-ES is required, the learning rate of mixture weightings, which reduces challenges of applying our methodology. Computer simulation results have shown better performances on optimization of multi-modal and multi-funnel functions.

The purpose of the SD-CPSO is to solve the issue that CPSO encounters when independent changes made by different swarms on correlated variables will deteriorate the performance of the algorithm. In the SD-CPSO, we propose a self-organization approach to the CPSO. This approach determines the separability between variables by covariance matrix adaptation, so that non-separable variables can be placed in the same swarm for evolution. Simulations show reasonable performance.

As to the future work in this dissertation, from the perspective of NFS design, the proposed QPSO and TSR-EA lack of the mechanism of self-determining the number of fuzzy rules on a NFS. The number of fuzzy rules in both algorithms have to be assigned by trial and error tests which would increase the difficulty of applying these two algorithms.

As to the future work of the proposed MS-CMA-ES, its convergence rate seems to be an issue to be investigated. From Table 4.11-13 we can see that despite the MS-CMA-ES has a good performance in finding global optimal of multi-funnel functions, it requires more calculation times to find it. In our opinion, we think there are two possible directions of enhancing its convergence rate.

The first direction is to alter its mixture weight updating rule. In this dissertation, as introducing mean shift procedure to the CMA-ES, an additional parameter, mixture weight, is also introduced. The updating rule of mixture weighting can be shown in Eq. (3.34)-(3.36). In this dissertation, we haven't probed into the connection between the convergence rate and the mixture weighting rule yet.

Another possible direction of enhancing the convergence rate of the MS-CMA-ES is to modify its bandwidth selection mechanism. In the MS-CMA-ES, the optimal bandwidth is derived from the AMISE theorem. The derived bandwidth determines both the number of clusters obtained and the number of samples in clusters. In this dissertation, if we can modify the fixed bandwidth selection that the MS-CMA-ES adopts into variable bandwidth selection mechanism, there might be a chance of increasing its convergence rate. The appended flexibility could arouse the issue of relationships among clusters, from which we could try to apply further manipulations to the clusters, such as separating or lumping. Moreover, under the premise of not explosively increase the computational cost, incorporating a iteratively increasing population size mechanism is also considerable. As to the future work on the SD-CPSO, the issue of pseudominima caused by the split of swarm still remains to be investigated.

Bibliography

[1] T. Baeck and H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation,* vol. 1, pp. 1-23, Spr. 1993.

[2] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation,* vol. 3, pp. 1-16, Spr. 1995.

[3] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation,* vol. 4, pp. 1-32, Spr. 1996.

[4] A. E. Eiben, Z. Michalewicz, and M. Schoenauer, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation,* vol. 3, pp. 124-141, Jul. 1999.

[5] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach," *IEEE Transactions on Evolutionary Computation,* vol. 3, pp. 257-271, Nov. 1999.

[6] E. Zitzler, D. Kalyanmoy, and T. Lothar, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation,* vol. 8, pp. 173-195, Sum. 2000.

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

[8] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, pp. 1942-1948, 1995.

[9] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281-295, Jun. 2006.

[10] M. Clerc and J. Kennedy, "The particle swarm—explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 225-239, 2004.

[11] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation,* vol. 9, pp. 159-195, Sum. 2001.

[12] N. Hansen, S. D. Muller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation,* vol. 11, pp. 1-18, Spr. 2003.

[13] N. Hansen, "The CMA evolution strategy: A Tutorial," 2008.

[14] J. J. Buckley and Y. Hayashi, "Neural nets for fuzzy-systems," *Fuzzy Sets And System*, vol. 71, no. 3, pp. 265-276, May 1995.

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

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

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

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning*. Cambridge, MA: MIT Press, 1998.

[19] J. N. Tsitsiklis and B. V. Roy, "An analysis of temporal-difference with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 834-836, Sep. 1983

[20] C. J. Wakins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, U.K., 1989.

[21] E. Banard, "Temporal-difference methods and Markov models," *IEEE Transactions on Systems Man and Cybernetics*, vol. 23, no. 2, pp. 357-365, Mar. 1993

[22] C. J. Wakins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[23] C. L. Muller and I. F. Sbalzarini, "A tunable real-world multi-funnel benchmark problem for evolutionary optimization and why parallel island models might remedy the failure of CMA-ES on it," in *Proceedings of the 1st International Joint Conference on Computational Intelligence*, Funchal, Portgual, pp. 248-253, Oct. 2009 ,

[24] D. J. Wales, "Energy landscapes and properties of biomolecules, " *Physical Biology,* vol. 2, pp. 86-93, Dec 2005.

[25] P. L. Clark, "Protein folding in the cell: reshaping the folding funnel," *Trends in Biochemical Sciences,* vol. 29, pp. 527-534, Oct. 2004.

[26] C. L. Muller, B. Benedikt, and I. V. Sbalzarini, "Particle swarm CMA evolution strategy for the optimization of multi-funnel landscapes," in *2009 IEEE Congress on Evolutionary Computation,* New York, vol. 1-5, pp. 2685-2692, 2009.

[27] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225-239, Jun. 2004.

[28] C. K. Goh, K.C. Tan, and D.S. Liu, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *European Journal of Operational Research*, vol. 202, pp. 42-54, 2010.

[29] S. F Lin and Y.C. Cheng, "Two-strategy reinforcement evolutionary algorithm using data-mining based crossover strategy with TSK-type fuzzy controllers," *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 9, pp. 3863-3885, Sep. 2010.

[30] Y. C Hsu, S. F. Lin, and Y. C Cheng, “Multiple groups cooperation based symbiotic evolution for TSK-type fuzzy contrrollers,” *Expert Systems with Applications l*, vol. 37, no. 7, pp. 5320-5330, July 2010.

[31] M. A. Potter and K. A. De Jong, “A cooperative coevolutiouary approach to function optimization,” *in Proceedings of Parallel Problem Solving from Nature - Ppsn Iii - International Conference on Evolutionary Computation*, vol. 866, Berlin, Germany, pp. 249-257, 1994.

[32] C. L. Muller, B. Baumgartner, and I. F. Sbalzarini, “Particle swarm CMA evolution strategy for the optimization of multi-funnel landscapes,” in *2009 IEEE Congress on Evolutionary Computation*, vol. 1-5, New York, pp. 2685-2692, 2009.

[33] L. X. Wang and J. M. Mendel, “Generating fuzzy rules by learning from examples,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 6, pp. 1414-1427, 1992.

[34] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 15, no. 1, pp. 116-132, 1985.

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

[36] J. S. R. Jang, “ANFIS: Adaptive-network-based fuzzy inference system,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 3, pp. 665-685, 1993.

[37] F. J. Lin, C. H. Lin, and P. H. Shen, “Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive,” *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 5, pp. 751-759, 2001.

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

[39] E. Mizutani and J. S. R. Jang, “Coactive neuro-fuzzy modeling,” *in Proceedings of IEEE International Conference Neural Networks*, Perth, WA , USA, vol. 2, pp. 760-765, Nov. 1995.

[40] C. J. Lin and C. C. Chin, “Prediction and identification using wavelet-based recurrent fuzzy neural networks,” *IEEE Transactions on Systems, Man and Cybernetics, Part B,* vol. 34, no. 5, pp. 2144-2154, 2004.

[41] K. S. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks,” *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990.

[42] C. F. Juang and C. T. Lin, “A recurrent self-organizing neuro-fuzzy inference network,” *IEEE Transactions on Neural Networks*, vol. 10, no. 4, pp.828-845, 1999.

[43] P. A. Mastorocostas and J. B. Theocharis, “A recurrent fuzzy-neural model for dynamic system identification,” *IEEE Transactions on Systems, Man and Cybernetics, Part B,* vol. 32, no. 2, pp. 176-190, 2002.

[44] R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” in *Proceedings of International Conference of Machine Learning*, pp. 161-168 , 2006.

[45] 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.

[46] 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.

[47] M. Conforth and Y. Meng, “Reinforcement learning for neural networks using swarm Intelligence,” *IEEE Swarm Intelligence Symposium*, pp. 7, 2008

[48] M. C. Choy, D. Srinivasan, and R. L. Cheu, “Cooperative, hybrid agent architecture for real-time traffic signal control,” *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 33, no. 5, pp. 597-607, 2003.

[49] C. J. Lin, Y. M. Lin, and C.Y. Lee, “Nonlinear system control using a recurrent neural fuzzy network based on reinforcement particle swarm optimization,” *in Proceedings of 2010 3rd International Symposium on Computational Intelligence and Design (ISCID 2010)*, vol. 2, pp. 196-200, 2010

[50] C. J. Lin and Y. C. Hsu, “Reinforcement hybrid evolutionary learning for recurrent wavelet-based neuro-fuzzy systems,” *IEEE Transactions on Fuzzy Systems,* vol. 15, no. 4, 2007, pp. 729-745.

[51] G. R. Harik, F. G. Lobo, and D. E. Goldberg, “The compact genetic algorithm,” *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 287–297, Nov. 1999.

[52] S. Bandyopadhyay, C. A. Murthy and S. K. Pal, “VGA-classifier: design and applications,” *IEEE Transactions on Systems, Man and Cybernetics, Part B,* vol. 30, no. 6, pp. 890–895, 2000.

[53] C. T. Lin and C. P. Jou, “GA-based fuzzy reinforcement learning for control of a magnetic bearing system,” *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 30, no. 2, pp. 276-289, 2000.

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

[55] Y. C. Hsu and S. F. Lin, “Reinforcement group cooperation based symbiotic evolution for recurrent wavelet-based neuro-fuzzy systems,” *Neurocomputing*, vol. 72, no. 10-12, pp. 2418-2432, 2009.

[56] Y. C. Cheng, S. F. Lin, and C.Y. Hsu, “Q-Value based particle swarm optimization for reinforcement neuro-fuzzy system design,” *International Journal on Computer*

Science and Engineering, Vol. 3, No. 10, pp.3477-3489, 2011.

[57] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning." *Journal of Machine Learning Research*, vol. 3, pp. 803-832, 2003.

[58] M. Lunacek, D. Whitely, and A. Sutton, "The impact of global structure on search," *in Proceedings of Parallel Problem Solving from Nature - Ppsn X*, G. Rudolph, *et al.*, Eds., ed Berlin: Springer-Verlag Berlin, vol. 5199, pp. 498-507, 2008.

[59] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, pp. 603-619, May 2002.

[60] A. Auger and N. Hanson, "Performance evaluation of an advanced local search evolutionary algorithm," *in Proceedings of 2005 IEEE Congress on Evolutionary Computation,* New York, vol. 1-3, pp. 1777-1784, 2005.

[61] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," *in Proceedings of IEEE Congress on Evolutionary Computation,* vol. 1-3, pp. 1769-1776, 2005.

[62] N. Hansen and S. Kern, "Evaluating the CMA evolution strategy on multimodal test functions," in *Parallel Problem Solving from Nature - Ppsn Viii*. vol. 3242, X. Yao, *et al.*, Eds., ed, pp. 282-291, 2004.

[63] C. T. Hsieh, *Particle Swarm Guided Evolution Strategy*. New York: Assoc Computing Machinery, 2007.

[64] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory,* vol. IT-21, pp. 32-4040, Jan. 1975.

[65] Y. Z. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 17, pp. 790-799, Aug. 1995.

[66] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*: Wiley, 1973.

[67] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, pp. 381-396, Mar. 2002

[68] M. P. Wand and M. C. Jones, "Comparison of smoothing parameterizations in bivariate kernel density-estimation," *Journal of the American Statistical Association,* vol. 88, pp. 520-528, Jun. 1993.

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

[70] M. Sugeno and K. Tanaka, "Successive identification of a fuzzy model and its applications to prediction of a complex system," *Fuzzy Sets Systems*, vol. 42, no. 3, pp. 315–334, 1991.

[71] N. Chaiyaratana and A. M. S. Zalzala, "Recent developments in evolutionary and

genetic algorithms: theory and applications," *in Proceedings of IEEE International Conference Genetic Algorithms in Engineering Systems: Innovations and Applications*, Glasgow, United Kingdom, pp. 270-277, Sep. 1997.

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

[73] Y. P. Zou, Z. K. Mi, and M. H. Xu, "Dynamic load balancing based on roulette wheel selection," *in Proceedings of IEEE International Conference on Communications, Circuits and Systems*, Guilin, China, vol. 3, pp.1732-1734, June 2006.

[74] P. M. Godley, D. E. Cairns, J. Cowie, and J. McCall, "Fitness directed intervention crossover approaches applied to bio-scheduling problems," *in Proceedings of IEEE International Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, Sun Valley, USA, pp. 120-127, Sept. 2008.

[75] S. Su and D. H. Zhan, "New genetic algorithm for the fixed charge transportation problem," *in Proceedings of IEEE International World Congress on Intelligent Control and Automation*, Dalian, China, vol. 2, pp. 7039-7043, Jun. 2002.

[76] W. Y. Wang, T. T. Lee, C. C. Hsu, and Y. H. Li, "GA-based learning of BMF fuzzy-neural network," *in Proceedings of IEEE International Conference on Fuzzy Systems*, Honolulu, USA, pp. 1234-1239, May 2002.

[77] G. Lin and X. Yao, "Analyzing crossover operators by search step size," *in Proceedings of IEEE International Conference on Evolutionary Computation*, Indianapolis, USA, pp. 107-110, Apr. 1997.

[78] 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 Proceedings of IEEE International Symposium on Information Technology*, Kuala Lumpur, Malaysia, vol. 3, pp. 1-5, Aug. 2008.

[79] C. J. Hsu, C. Y. Huang, and T. Y. Chen, "A modified genetic algorithm for parameter estimation of software reliability growth models," *in Proceedings of IEEE International Symposium on Software Reliability Engineering*, Seattle, WA, USA, pp. 281-282, Nov. 2008.

[80] P. Luo, J. F. Teng, J. H. Guo and Q. Li, "An improved genetic algorithm and its performance analysis," *in Proceedings of IEEE International Conference on Info-tech and Info-net*, Beijing, China, vol. 4, pp. 329-333, Oct. 2001.

[81] G. W. Greenwood, "Adapting mutations in genetic algorithms using gene flow principles," *in Proceedings of IEEE Congress on Evolutionary Computation*, Canberra, Australia vol. 2, pp.1392-1397, Dec. 2003.

[82] H. J. Lee, Y. S. Ma, and Y. R. Kwon, "Empirical evaluation of orthogonality of class mutation operators," *in Proceedings of IEEE International Conference on Software*

Engineering, Busan, Korea, pp. 512-518, Nov. 2004

[83] N. S. Chaudhari, A. Purohit, and A. Tiwari, "A multiclass classifier using genetic programming," *in Proceedings of IEEE International Conference on Control, Automation, Robotics and Vision*, Hanoi, Vietnam, pp. 1884-1887, Dec. 2008.

[84] N. Gomez Bias, L. F. Mingo, and J. Castellanos, "Networks of evolutionary processors with a self-organizing learning," *in Proceedings of IEEE International Conference on Computer Systems and Applications*, Doha, Qatar, pp. 917-918, Mar. 2008.

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

[86] M. P. Wand, and M. C. Jones, *Kernel Smoothing*. Chapman & Hall, London, 1995.

[87] M. C. Jones, J. S. Marron, and S. J. Sheather, " A brief survey of bandwidth selection for density estimation," *Journal of the American Statistical Association* vol. 91, no.433, pp. 401-407, 1996.

[88] S. R. Stephan, "Multivariate locally adaptive density estimation," *Computational Statistics & Data Analysis*, vol. 39, no. 2, pp. 165-186, 2002

[89] S. R. Stephan and D. W. Scott, "On locally adaptive density estimation." *Journal of the American Statistical Association*, vol. 91, no. 436, pp. 1525-1534,1996..

[90] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition II*, 142-149, 2000.

[91] A. Mayer and H. Greenspan, "An adaptive mean-shift framework for MRI brain segmentation," *IEEE Transactions on Medical Imaging*, vol. 28, no. 8, pp. 1238-1250, 2009.

[92] J. Hwang, D. Lee, K. Huh, H, Na, and H. Kang, "Development of a path planning system using mean shift algorithm for driver assistance," *International Journal of Automotive Technology*, vol. 12, no. 1, pp. 119-124, 2011.

[93] S. T. Tokdar and R. E. Kass, "Importance sampling: a review," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 1, pp. 54-60, 2010.

[94] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, New York, 2003.

[95] 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.

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

[97] 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.

[98] 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 *Proceedings of IEEE International Conference on Fuzzy Systems*, Reno, NV, USA, pp. 1092-1097, May 2005.

[99] 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.

[100] D. E. Moriarty and R. Miikkulainen, “Efficient reinforcement learning through symbiotic evolution,” *Machine Learning*, vol. 22, pp. 11-32, 1996.

[101] K. Furuta and M. Iwase, “Swing-up time analysis of pendulum,” *Bulletin of the Polish Academy of Sciences - Technical Sciences*, vol. 52, no. 3, pp. 153-163, 2004.

[102] C. Popescu, “Nonlinear control of underactuated horizontal double pendulum,” M.S. thesis, University of Florida Atlantic, Boca Raton, Florida, U.S., 2002.

[103] N. Hansen, A. Auger, S. Finck, and R. Ros, “Real-parameter block-box optimization benchmarking 2010: experimental setup,” INRIA Research Report RR-72152010, 2010.

[104] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, “Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization,” Nanyang Technological University, Singapore and KanGAL Report Number 2005005, 2005

[105] J. Kennedy, “The particle swarm: social adaptation of knowledge,” in *Proceedings of IEEE International Conference on Evolutionary Computation*, 303-308, 1997.

[106] S. F. Lin, Y. C. Cheng, Jyun-Wei Chang, and P. C. Hung, “Separability detection cooperative particle swarm optimizer based on covariance matrix adaptation,” *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 4, pp. 18-24, 2012.

Vita

Yi-Chang Cheng has received his B.S. degree in department of engineering science from National Cheng Kung University, Taiwan, in 2005. He proposed his Ph. D. oral exam at institute of electrical control engineering from the National Chiao Tung University, Taiwan, R.O.C. in May, 2012. His research interests lie in the areas of neural networks, neural fuzzy systems, evolutionary algorithms and pattern recognition.

Publication List

Accepted Journal Papers:

1. Sheng-Fuu Lin and Yi-Chang Cheng, "Two-Strategy Reinforcement Evolutionary Algorithm using Data-mining based Crossover Strategy with TSK-type Fuzzy Controllers," *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 9, pp. 3863-3885, 2010. (SCI/EI)

2. Yung-Chi Hsu, Sheng-Fuu Lin, and Yi-Chang Cheng, "Multiple Groups Cooperation based Symbiotic Evolution for TSK-type Fuzzy Controllers," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5320-5330, 2010. (SCI/EI)

3. Yi-Chang Cheng, Sheng-Fuu Lin, and Chi-Yao Hsu, "Q-Value Based Particle Swarm Optimization for Reinforcement Neuro-Fuzzy System Design," *International Journal on Computer Science and Engineering*, vol. 3, no. 10, pp. 3477-3489, 2011.

4. Chi-Yao Hsu, Yi-Chang Cheng, and Sheng-Fuu Lin, "Efficient and Accurate Image Alignment using TSK-type Neuro-Fuzzy Network with Data-Mining based Evolutionary Learning Algorithm," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 96, pp.1-22. (SCI/EI)

5. Chi-Yao Hsu, Yi-Chang Cheng, and Sheng-Fuu Lin, "Precise Image Alignment using Cooperative Neural-Fuzzy Networks with Association Rule Mining based Evolutionary Learning Algorithm," *Optical Engineering*, vol. 51, no. 2, pp. 027006:1-15, 2012. (SCI/EI)

6. Sheng-Fuu Lin, Yi-Chang Cheng, Jyun-Wei Chang, and Pei-Chia Hung, "Separability Detection Cooperative Particle Swarm Optimizer based on Covariance Matrix Adaptation," *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 4, pp. 18-24, 2012.

Submitted Journal Papers:

1. Sheng-Fuu Lin, Yi-Chang Cheng, Chi-Yao Hsu, and Jyun-Wei Chang, "Two-Strategy Reinforcement Hybrid Evolutionary Learning for Recurrent Wavelet-Based Neuro-Fuzzy Systems," revised in *International Journal of Adaptive Control and Signal Processing*. (SCI/EI), (3rd Revised on 2012/3/20)

2. Yi-Chang Cheng, Sheng-Fuu Lin, Wei-Ching Lin, and Jyun-Wei Chang, "Mean

Shift-Based Evolution Strategy with Covariance Matrix Adaptation for the Optimization of Multi-Funnel Landscapes," *Journal of Optimization Theory and Applications*. (SCI/EI)

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," in *International Conference on Information Systems Analysis and Synthesis*, Orlando, Florida, USA, Jul. 10-13, 2009.

2. Sheng-Fuu Lin, Jyun-Wei Chang, Yi-Chang Cheng, and Yun-Chi Hsu, "A Novel Self-Constructing Evolution Algorithm for TSK-type Fuzzy Model Design," in *Proceedings of IEEE World Congress on Computational Intelligence*, Barcelona, Spain, Jul. 18-23, 2010.

3. Yi-Chang Cheng, Wei-Ching Lin, and Sheng-Fuu Lin, "A Self-Organization Approach to Cooperative Particle Swarm Optimization," in *Proceedings of the 18th National Conference on Fuzzy Theory and Its Applications*, Hualien, Taiwan, Dec. 3-4, 2010.

4. Yi-Chang Cheng, Wei-Ching Lin, and Sheng-Fuu Lin, "A Separability Detection Approach to Cooperative Particle Swarm Optimization," in *The 7th International Conference on Natural Computation*, Shanghai, China, Jul. 26-28, 2011.