

國立交通大學

資訊科學與工程研究所

碩士論文

引入重組運算子與動態鏈結開發技術於
粒子群最佳法



Introducing Recombination with Dynamic Linkage
Discovery to Particle Swarm Optimization

研究生：簡明昌

指導教授：陳穎平 教授

中華民國九十五年六月

引入重組運算子與動態鏈結開技術於粒子群最佳化演算法
Introducing Recombination with Dynamic Linkage Discovery to
Particle Swarm Optimization

研究生：簡明昌

Student : Ming-chung Jian

指導教授：陳穎平

Advisor : Ying-ping Chen

國立交通大學
資訊科學與工程研究所
碩士論文



Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

引入重組運算子與動態鏈結開發技術

於粒子群最佳化演算法

學生：簡明昌

指導教授：陳穎平

國立交通大學資訊科學與工程研究所

摘要

粒子群最佳化演算法(PSO)為一有效率之演化計算演算法，近年更有許多改良粒子群最佳化演算法的研究持續進行中，其中針對粒子群最佳化演算法與基因演算法(GAs)之結合的主題亦已成為熱門研究。另一方面，於基因演算法中，如何有效處理基因鏈結(Genetic linkage)問題亦成為有效改良基因演算法效能的重要議題。因此於本論文中，主要欲達成之目標有二。第一，我們透過引入基因鏈結之概念於粒子群最佳化演算法中，藉以提昇其搜尋之效能。為了完成之目標，我們必須瞭解粒子群最佳化演算法及基因鏈結問題之特性，並尋找適當的結合模式以有效發揮兩者的功能。除此之外，我們希望能完成之另一目標為有效處理、辨識實數問題之基因鏈結，而為達成此一目標，我們必須設計一個特別的基因鏈結辨識技術。

於此篇論文中，我們假設基因鏈結於實數問題中是存在且隨著搜尋過程中而產生變動。於此假設前提下，我們設計了動態基因鏈結開發技術(dynamic linkage discovery technique)來處理基因鏈結問題，此技術為根據適者生存之自然淘汰為概念所設計，為有效且計算成本低廉的基因鏈結辨識技術。此外，為了有效提昇粒子最佳化演算法及基因鏈結辨識結合的效能，我們亦設計了重組運算子(recombination operator)，透過操作使用動態基因鏈結開發技術所辨識出的建構基石(building blocks)，粒子群最佳化演算法便能有效的處理存在於實數問題中的基因鏈結，於此研究中，我們將結合重組運算子、動態鏈結開發技術與粒子群最佳化的演算法稱為PSO-RDL。

因此於此研究中，我們藉由引入動態基因鏈結開發技術來處理實數問題中的基因鏈結，搭配重組運算子的運作以提昇粒子群最佳化演算法的效能。我們透過仔細設計的各项數

學函式做為評估實驗，針對所提出的演算法做效能上的評估分析。另外，我們也將 PSO-RDL 應用於電力系統上的經濟調配問題，而由各項實驗的結果也可知，我們所設計的演算法確有完成提昇效能的目標。

關鍵詞： 粒子群最佳化演算法；基因演算法；基因鏈結；建構基石；動態基因鏈結開發技術；重組運算子。



Introducing Recombination with Dynamic Linkage Discovery to Particle Swarm Optimization

Student: Ming-chung Jian

Advisor: Ying-ping Chen

Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University



There are two main objectives in this thesis. The first goal is to improve the performance of the particle swarm optimizer by incorporating linkage concept which is an essential mechanism in genetic algorithms. To achieve this purpose, we need to know the characteristics of the particle swarm optimizer and the genetic linkage problem. Through survey of the particle swarm optimization and the linkage problem, we then figure out how to introduce the linkage concept to particle swarm optimizer. Another goal is to address the linkage problem in real-parameter optimization problems. We have to study different linkage learning techniques, and understand the meaning of genetic linkage in real-parameter problems. After that, we design a novel linkage identification technique to achieve this objective.

In this thesis, the existence of genetic linkages in real-parameter optimization problem and

that genetic linkages are dynamically changed through the search process are the primary assumptions. With these assumptions, we develop the dynamic linkage discovery technique to address the linkage problem. Moreover, a special recombination operator is designed to promote the cooperation of particle swarm optimizer and linkage identification technique. In the consequence, we introduce the recombination operator with the technique of dynamic linkage discovery to particle swarm optimization (PSO). Dynamic linkage discovery is a costless, effective linkage recognition technique adapting the linkage configuration by utilizing the natural selection without incorporating extra judging criteria irrelevant to the objective function. Furthermore, we employ a specific recombination operator to work with the building blocks identified by dynamic linkage discovery. The whole framework forms a new efficient search algorithm and is called PSO-RDL in this study. Numerical experiments are conducted on a set of carefully designed benchmark functions and demonstrate good performance achieved by the proposed methodology. Moreover, we also applied the proposed algorithm on the economic dispatch problem which is an essential topic in power control systems. The experimental results show that PSO-RDL can performs well both on numerical benchmark and real-world applications.

keywords:

Particle swarm optimization, genetic algorithms, genetic linkage, building blocks, dynamic linkage discovery, recombination operator

誌謝

本論文得以順利完成，首先特別要感謝我的指導教授陳穎平老師的細心指導與諄諄教誨，感謝老師於兩年來細心與耐心的指導以及對我孜孜不倦的教誨，使我不僅在研究過程中受益良多，且在待人處世各方面均有許多的成長，謹致上最誠摯的謝意。

口試期間，承蒙孫春在老師、鍾雲恭老師，洪炯宗老師提供許多寶貴的意見並逐字斧正，使本論文呈現更完整的面貌，他們對學生的提攜以及謙沖的風範，將永記在心。

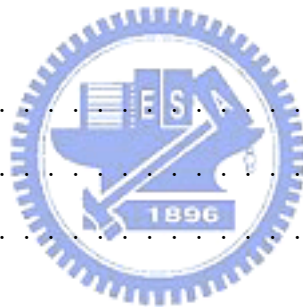
感謝於論文撰寫期間，自然計算實驗室的成員劉柏甫、謝長泰、洪秉竹、陳宏偉、林盈吟、李文瑜等各位學長、同學及學弟妹的協助，讓我在遇到困難及心情低落時能獲得適時的支持與鼓勵，有了你們的陪伴讓我兩年的研究生生活顯得多采多姿且值得回憶。

最後，感謝我一生中最重要的家人在此期間的體諒、支持與鼓勵，讓我可以無後顧之憂的專心於學業上，你們是我在論文寫作過程中最重要的精神後盾，謹將此論文獻給你們，謝謝你們。

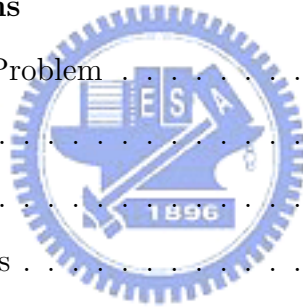
簡明昌 謹誌
中華民國九十五年六月

Contents

Abstract	i
Table of Contents	vi
List of Figures	viii
List of Tables	ix
1 Introduction	1
1.1 Motivation	1
1.2 Thesis Objectives	2
1.3 Road Map	3
2 Particle Swarm Optimization	5
2.1 Historical Background	5
2.2 Particle Swarm Optimization	6
2.3 Parameters of PSO	8
2.4 Recent Advances in PSO	9
2.5 Summary	10
3 Genetic Linkage	12
3.1 What Is Genetic Linkage?	12
3.2 Why Is Genetic Linkage Important?	14
3.3 Genetic Linkage Learning Techniques	15
3.4 Summary	16



4	Framework	17
4.1	Dynamic Linkage Discovery Technique	17
4.2	Recombination Operator	20
4.3	Introducing Recombination with Dynamic Linkage Discovery in PSO	22
4.4	Summary	23
5	Experimental Results	26
5.1	Test Functions	26
5.2	Parameter Setting	29
5.3	Experimental Results	29
5.4	Discussion	30
5.5	Summary	43
6	Real-world Applications	46
6.1	Economic Dispatch Problem	46
6.2	Our Solution	48
6.3	Experiments	49
6.3.1	Test Problems	49
6.3.2	Experimental Results	50
7	Conclusions	56
7.1	Summary	56
7.2	Future Work	57
7.3	Main Conclusions	58
A	CEC'05 25 Real-Parameter Functions	60
B	Global Optimum for CEC'05 25 Real-Parameter Functions	71
C	Solutions found by PSO-RDL for CEC'05 25 Real-Parameter Functions	73



List of Figures

2.1	Flowchart of Particle Swarm Optimizer	11
4.1	flowchart of the dynamic linkage discovery process	19
4.2	An illustration of linkage identification	20
4.3	The procedure of how a new particle is generated through the recombina- tion operator	21
4.4	The process of constructing new population through recombination operator	22
4.5	Pseudocode of PSO-RDL	24
4.6	The flow of the PSO-RDL	25
5.1	Fitness convergence and linkage dynamics of the Sphere function	41
5.2	Fitness convergence and linkage dynamics of the Shifted Rotated Griewank's function	44
5.3	Fitness convergence and linkage dynamics of the Shifted Expanded Griewank's plus Rosenbrock's function.	44
5.4	Fitness convergence and linkage dynamics of the Shifted Rastrigin's function.	45

List of Tables

5.1	Related parameters of test functions	28
5.2	Parameter setting in this numerical experiments	29
5.3	Experimental results for test functions 1-5	31
5.4	Experimental results for test functions 6-10	32
5.5	Experimental results for test functions 11-15	33
5.6	Experimental results for test functions 16-20	34
5.7	Experimental results for test functions 21-25	35
5.8	Comparison with other algorithms for functions 1-5	36
5.9	Comparison with other algorithms for functions 6-10	37
5.10	Comparison with other algorithms for functions 11-15	38
5.11	Comparison with other algorithms for functions 16-20	39
5.12	Comparison with other algorithms for functions 21-25	40
5.13	Problems solved in different evolutionary algorithms	41
6.1	Parameters to 3-unit system	50
6.2	Parameters to 40-unit system	52
6.3	Result for 3-unit system	53
6.4	Result for 40-unit system	53
6.5	Generation output for 40-unit system	54
6.6	100 trial result for 40-unit system	55
6.7	t-Test results for PSO-RDL and MPSO I	55
6.8	t-Test results for PSO-RDL and MPSO II	55
B.1	Global optimum solution for CEC's benchmark	72

C.1 PSO-RDL's solution for CEC's benchmark 74



Chapter 1

Introduction

1.1 Motivation

Particle swarm optimizer (PSO), introduced by Kennedy and Eberhart in 1995 [1, 2], emulates flocking behavior of birds to solve optimization problems. The PSO algorithm is conceptually simple and can be implemented in a few lines of codes. In PSO, each potential solution is considered as a particle. All particles have their own fitness values and velocities. These particles fly through the D -dimensional problem space by learning from the historical information of all the particles. There are global and local versions of PSO. Instead of learning from the personal best and the best position discovered so far by the whole population as in the global version of PSO, in the local version, each particle's velocity is adjusted according to its own best fitness value and the best position found by other particles within its neighborhood. Focusing on improving the local version of PSO, different neighborhood structures are proposed and discussed in the literature. Moreover, the position and velocity update rules have been modified to enhance the PSO's performance as well.

On the other hand, genetic algorithms (GAs), introduced by John Holland [3, 4], are stochastic, population-based search and optimization algorithms loosely modeled after the paradigms of evolution. Genetic algorithms guide the search through the solution space by using natural selection and genetic operators, such as crossover, mutation, and the like. Furthermore, the GA optimization mechanism is theorized by researchers [3, 4, 5] with building block processing, such as creating, identifying and exchanging. Building blocks are conceptually non-inferior sub-solutions which are components of the superior

complete solutions. The building block hypothesis states that the final solutions to a given optimization problem can be evolved with a continuous process of creating, identifying, and recombining high-quality building blocks. According to that the GA's search capability can be greatly improved by identifying building blocks accurately and preventing crossover operation from destroying them [6, 7], linkage identification, the procedure to recognize building blocks, plays an important role in GA optimization.

The two aforementioned optimization techniques are both population-based that have been proven successful in solving a variety of difficult problems. However, both models have strength and weakness. Comparisons between GAs and PSOs can be found in the literature [8, 9] and suggest that a hybrid of these two algorithms may lead to further advances. Hence, a lot of studies on the hybridization of GAs and PSOs have been proposed and examined. Most of these research works try to incorporate genetic operators into PSO [10, 11], while some try to introduce the concept of genetic linkage into PSO [12]. Based on the similar idea employed by linkage PSO[12], our work is to introduce recombination working on building blocks to enhance the performance of PSO with the concept of genetic linkage.



1.2 Thesis Objectives

This thesis presents a research project that aims to address the genetic linkage problem in real-parameter optimization problems and introduce the genetic linkage concept to particle swarm optimizer. Thus, there are two primary objectives:

1. With the assumption that linkage problems exist in real-parameter problems , a linkage identification technique is needed to address the genetic linkage problems. This thesis provides both the linkage identification mechanism and observation of experiments to support the initial assumption.
2. To improve the performance of particle swarm optimizer, the genetic linkage concept is introduced. An optimization algorithm that incorporates these mechanism is developed and numerical experiments are done to evaluate the performance.

Focusing on the two objectives, in this research work, we propose a dynamic linkage discovery technique to effectively detect the building blocks of the objective function. This technique differs from those traditional linkage detection technique in that the evaluation cost is eliminated. The idea is to dynamically adapt the linkage configuration according to the search process and the feedback from the environment. Thus, this technique is costless and easy to be integrated into existing search algorithms. Our method introduces the linkage concept and the recombination operator to the operation of particle swarm optimizer.

1.3 Road Map

This thesis is composed by six chapters. The detailed organization is given as follows:

- Chapter 1 consists of the motivation, objectives and organization of this study. It describes why this research work is important and the main tasks to be accomplished.
- Chapter 2 provides a complete overview of the particle swarm optimization algorithm. The background and traditional particle swarm optimizer is introduced, and the parameter controls that have an impact on the performance are discussed. Moreover, recent advances of related work in particle swarm optimization are also surveyed in this chapter.
- Chapter 3 presents the concept of genetic linkage in genetic algorithms. The definition and importance of genetic linkage are described. Linkage learning techniques in the literature are also briefly discussed.
- Chapter 4 described the proposed method in detail. The three mechanisms including particle swarm optimizer, dynamic linkage discovery technique and recombination operator, are introduced. The algorithm composed by the above three components is then presented.
- Chapter 5 shows the numerical experimental results that evaluate the performance of the proposed algorithm. The description of the test functions, parameter settings,

and simulated results are given. The discussion and observation of the experiments are covered in this chapter certainly.

- Chapter 6 applied the designed algorithm on economic dispatch problems, which are a significant topic in the power system. It describes the objectives and formulations of economic dispatch problems, and then gives the proposed solution and experimental results.
- Chapter 7 give a summary of this research work. The future works and the main conclusions of the study are proposed.



Chapter 2

Particle Swarm Optimization

The particle swarm optimizer (PSO), introduced by Kennedy and Eberhart in 1995 [1, 2], emulates flocking behavior of birds to solve the optimization problems. The PSO algorithm is conceptually simple and can be implemented in a few lines of codes. In PSO, each potential solution is considered as a particle. All particles have their own fitness values and velocities. These particles fly through the D -dimensional problem space by learning from the historical information of all the particles. In the following sections, we will give a complete overview of PSO. Section 1 introduces the historical background of PSO, section 2 describes how PSO works, section 3 discusses the parameter control in PSO, section 4 has a brief survey of PSO related to our work, and finally, section 5 gives a summary of this chapter.

2.1 Historical Background

Many scientists have studied and created the computer simulation of various interpretations of the movement of organisms in a bird flock or fish school [13, 14]. From simulations, it is considered that there might be a local process that underlies the group dynamics of bird social behavior and relies heavily on manipulation of inter-individual distance. That is, the movement of the flock was an outcome of the individuals' efforts to maintain an optimum distance from their neighborhood [1].

The social behavior of animals and in some cases of humans, is governed by similar rules. However, human social behavior is more complex than a flock's movement for at least one obvious reason: collision. Two individuals can hold identical attitudes and

beliefs without banging together, while two birds cannot occupy the same position in space without colliding. Such an abstraction in human social behavior has comprised a motivation for developing a model for it.

As sociobiologist E.O. Wilson [15] has written, "In theory at least, individual members of the school can profit from the discoveries and previous experience of all other members of the school during the search for food. This advantage can become decisive, outweighing the disadvantages of competition for food items, whenever the resource is unpredictably distributed in patches". This statement and numerous examples coming from nature enforce the view ,that social sharing of information among the individuals of a population may offers an evolutionary advantage. This belief has formed a fundamental of the development of particle swarm optimization, which will be introduced in the next section.

2.2 Particle Swarm Optimization

As mentioned above, PSO began as a simulation of a simplified social behavior that was used to visualize the movement of a birds' flock. Considering such as nearest-neighbor velocity matching, the cornfield vector and acceleration by distance, several variation of the simulation model has been through a trial and error process and finally results in a first simplified version of PSO [1].

PSO is similar to genetic algorithm in that both of them are population based search algorithms. A population of individuals is randomly initialized where each individual is considered as a potential solution of the problem. Especially an individual is called a "particle" and a population is called a "swarm" in PSO [1]. However, in PSO, each potential solution is also assigned an adaptable velocity that enables the particle to fly through the hyperspace. Moreover, each particle has a memory that keeps track of the best position in the search space that it has ever visited [2]. Thus the movement of a particle is an aggregated acceleration towards its best previously visited position and the best individual of its neighborhood.

There are mainly two variants of PSO algorithm were developed [2]. The major

difference between the two variant is that one with a global neighborhood while the other with a local neighborhood. According to the global variant, particle's movement is influenced by its previous best position and the best particle of the whole swarm. On the other hand, each particle moves according to its previous best position and the best particle of its restricted neighborhood in the local variant. Because the local version of PSO can be derived from the global variant through minor changes. In the next paragraph, we will have a complete introduction of the global version PSO.

In PSO, each particle is treated as a point in a D -dimensional space. The i th particle is represented by a D -dimensional vector, $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$. The best previous position of any particle is represented as $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$, and the best particle's position of the whole swarm is represented by $P_g = (p_{g1}, p_{g2}, \dots, p_{gD})$. The velocity of particle i is also represented as a D -dimensional vector, $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$. The position and velocity of each particle is updated according to the following equation:

$$v_{id}^{n+1} = v_{id}^n + c_1 * rand_1^d() * (p_{id}^n - x_{id}^n) + c_2 * rand_2^d() * (p_{gd}^n - x_{id}^n) \quad (2.1)$$

$$x_{id}^{n+1} = x_{id}^n + v_{id}^{n+1} \quad (2.2)$$

where $d = 1, 2, \dots, D$; $i = 1, 2, \dots, N$, and N is the size of the swarm; c_1 and c_2 are two positive constants, called the acceleration constant; $rand_1()$ and $rand_2()$ are two uniformly distributed random numbers in the range $[0, 1]$; and $n = 1, 2, \dots$, determines the iteration number. The second part of the Equation 2.1 is the "cognition" part, which represents the private thinking of the particle itself, and the third part is the "social" part, which represents the collaboration among the particles [16]. A flowchart of how the particle swarm optimization works on a swarm is shown in Figure 2.1.

Equation 2.1 and 2.2 define the initial version of the PSO algorithm. Referring to Equation 2.1, the second and third part have an important influence on the movement of each particle. Without these two parts, the particles will keep on flying at the current speed in the same direction until they hit the boundary. On the other hand, without the first part of equation 2.1, the particle's velocity is determined by the current best position

and previous best position. Thus all the particles will tend to move toward the same position. In such a case, it is more likely that the second and third part of the Equation 2.1 play as a local searcher, while the first part plays as a global searcher. There is a tradeoff between the local and global search. For different problems, different balance between them should be considered. In order to achieve this goal, a parameter the called inertia weight 'w' is introduced, and the Equation 2.1 is modified as [17]:

$$v_{id}^{n+1} = w * v_{id}^n + c_1 * rand_1^n() * (p_{id}^n - x_{id}^n) + c_2 * rand_2^n() * (p_{gd}^n - x_{id}^n) \quad (2.3)$$

Since the particle swarm optimization is conceptually simple and can be easily implemented, the fine-tune of parameters become an important topic which have great impact on the performance of the particle swarm optimization. Discussion about the parameters will be given in the next section.

2.3 Parameters of PSO

In the previous section, we have introduced PSO algorithm and given a modified version. In order to facilitate the efficiency of PSO algorithm, it is important to understand how the parameters would influence PSO. Thus, in this section, we will have a brief discussion on this topic.

Since the particles are "flying" through the search space, it is necessary to have a maximum value V_{max} on it. The parameter has been proven to be crucial because it actually serves as a constraint that controls the maximum global exploration ability that PSO can have [18]. Moreover, the inertia weight described in Equation 2.2 is also important for the balance of global and local search ability. When V_{max} is large, PSO can have a large range of exploration by selecting a proper inertia weight. By setting a small maximum velocity, PSO would act as a local searcher whatever the inertia weight is selected. As Shi and Eberhart suggested in [18], since the maximum velocity affects global exploration indirectly, it is considered better to controls the global exploration ability through the inertia weight only. Furthermore, choosing a large inertia weight to facilitate greater

global exploration is not a good strategy, and a smaller inertia weight should be selected to achieve a balance between global and local exploration for a faster search.

The parameters c_1 and c_2 represent the acceleration rates of cognitive and social parts of each particle. Thus, fine-tuning could result in faster convergence of the swarm. As default values in [1], $c_1 = c_2 = 2$ were proposed. Recent work has suggested that it might be better to choose a larger cognitive parameter, c_1 , but $c_1 + c_2 \leq 4$ [19].

Finally, $rand_1()$ and $rand_2()$ are random numbers uniformly distributed in the range $[0,1]$ and are used to maintain the diversity of the whole swarm.

In the past decade, many studies of improving the performance of particle swarm optimization were done. Variants of particle swarm optimization are discussed and proposed. In the next section, related works in the literature will be given.

2.4 Recent Advances in PSO

In this section, we will have a brief survey of PSO which is related to our research. As mentioned previously, there are mainly two variants of PSO developed and we have given an overview of the global one. However, many researches on the local version of PSO have been working on. In the local variant, the P_g has been replaced by P_l , the best position achieved by a particle within its neighborhood. Focusing on improving the local version of PSO, different neighborhood structures have been proposed and discussed [20, 21, 22].

Furthermore, studies on modifying the rule of updating position and velocity are also conducted [12, 23, 24]. Devicharan and Mohan [12] first computed the elements of linkage matrix based on observation of the results of perturbations performed in some randomly generated particles. These elements of the linkage matrix were used in a modified PSO algorithm in which only strongly linked particle positions were simultaneously updated. Liang et al [23, 24] proposed a learning strategy where each dimension of a particle learned from particle's historical best information, while each particle learned from different particles' historical best information for different dimensions.

In order to enhance the performance of PSO by introducing the genetic operators and/or mechanisms, many hybrid GA/PSO algorithms have been proposed and tested on

function minimization problems [10, 11, 25, 26] Lvbjerg et al [10] incorporated a breeding operator into the PSO algorithm, where breeding occurred inline with the standard velocity and position update rules. Robinson et al [25] tested two hybrid version. The first used the GA algorithm to initialize the PSO population while another used the PSO to initialize the GA population. Shi et al [26] proposed two approaches. The main idea of the proposed algorithm was to parallelly integrate PSO and GA. Settles and Soule [11] combined the standard velocity and position update rules of PSO with the concept of selection, crossover, and mutation from GAs. They employed an additional parameter, the *breeding ratio*, to determine the proportion of the population which underwent the breeding procedure (selection, crossover, and mutation) in the current generation.

Based on the brief literature review, we know that since PSO was proposed, many research focusing on improving the performance of PSO were conducted. By incorporating different mechanisms such as special neighborhood structure, modified update equation or hybridizing with GA concepts, many different models of PSO were developed. In this research work, we have also proposed a particular model of PSO which dissimilar to those described in this section. A detailed description will be given in Chpater 3.

2.5 Summary

In this chapter, particle swarm optimization algorithms were briefly introduced, including its historical background and working principles. The initial and modified global version PSO were described. In order to understand how parameters affect the PSO, we make a short discussion of the parameter control in PSO. Finally, recent advances of PSO that are related to our research work were surveyed. From the survey, we know that the hybridization of PSO and GAs has become a popular research topic. It inspired us to improve the particle swarm optimizer by incorporating the linkage concept in GAs. A complete review of the genetic linkage in GAs will be provided in the next chapter.

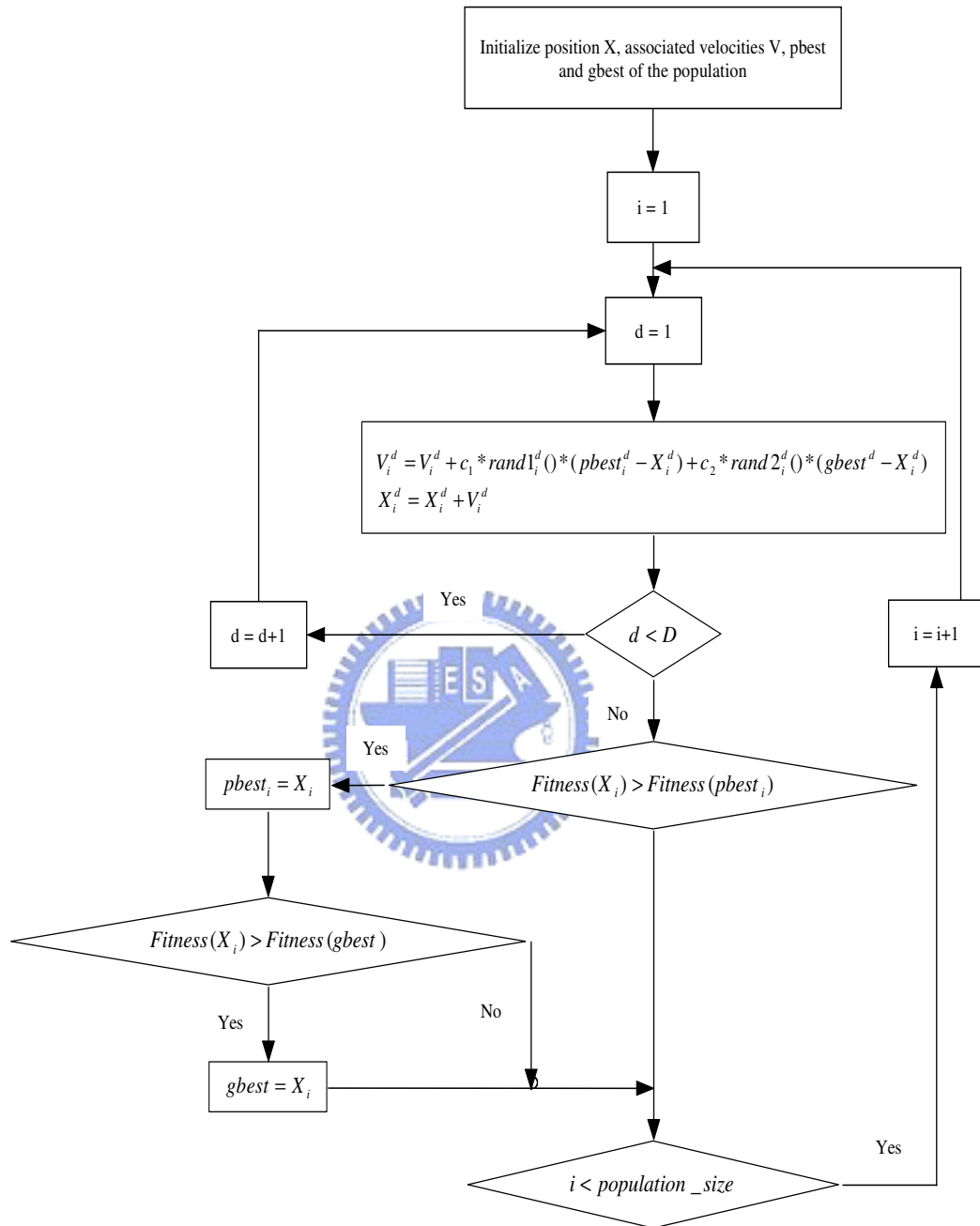


Figure 2.1: A flowchart of particle swarm optimizer, in each generation, PSO manipulate each particle through updating their position and velocity according to Equation 2.1 and Equation 2.2. After updating particle's position and velocity, PSO then evaluates particle's fitness value and decides that the previous best individual and the global best individual should be replaced or not

Chapter 3

Genetic Linkage

In this chapter, we discuss about the topic of genetic linkage in genetic algorithms. We will present the definition and the importance of genetic linkage. The genetic linkage learning techniques are also discussed. Particularly, the following topics are presented:

- The definition of genetic linkage: Describes what genetic linkage is in genetic algorithms.
- The importance of genetic linkage: Describes why linkage learning is an essential topic in genetic algorithms.
- The linkage learning techniques: Describes what kinds of techniques have been developed to address the genetic linkage problems.



3.1 What Is Genetic Linkage?

Since the central topics in this chapter is genetic linkage, we first give the definition of genetic linkage in genetic algorithms. The basic idea and assumption of genetic algorithms will be given, and then the definition of genetic linkage in genetic algorithms will be explained.

Genetic algorithms (GAs), introduced by John Holland [3, 4], are stochastic, population-based search and optimization algorithms loosely modeled after the paradigms of evolution. Genetic algorithms search the solution space by using natural selection and genetic operators, such as crossover, mutation, and the like. Furthermore, the GA optimization mechanism is theorized by researchers [3, 4] with building block processing, such as

creating, identifying and exchanging. Building blocks are conceptually non-inferior sub-solutions which are components of the superior complete solutions. The building block hypothesis states that the final solutions to a given optimization problem can be evolved with a continuous process of creating, identifying, and recombining high-quality building blocks.

For genetic algorithms, the chromosome is represented as a string of characters, and we use genetic operators like crossover and mutation to manipulate these chromosomes. Holland indicated that crossover operator in genetic algorithms induce a *linkage* phenomenon [3]. In [27], the term *genetic linkage* has been loosely defined for a set of genes as follows:

If the genetic linkage between these genes is tight, the crossover operator disrupts them with a low probability and transfers them all together to the child individual with a high probability. On the other hand, if the genetic linkage between these genes is loose, the crossover operator disrupts them with a high probability and transfers them all together to the child individual with a low probability.

This definition implies the genetic linkage of a set of genes depend on the chromosome representation and the crossover operator.

From the definition of genetic linkage given above, we can infer that the linkage phenomenon is induced by using crossover operator with string type representation. For example, consider a 6-bit function consisting of two independent subfunctions. For $x = [x_1, x_2, x_3, x_4, x_5, x_6]$, two possible combinations of subfunctions are shown as follows:

$$F_1(x) = f_1(x_1, x_2, x_3) + f_2(x_4, x_5, x_6)$$

$$F_2(x) = f_1(x_1, x_3, x_5) + f_2(x_2, x_4, x_6)$$

Taking one-point crossover as an example, it is obviously to see that genes belonging to the same subfunction are likely to stay or to be transferred together in $F_1(x)$, while in $F_2(x)$, genes belonging to the same subfunction are split almost every time when a crossover event occurs.

In Goldberg’s design decomposition [5], the first step to design a competent genetic algorithm is to know what genetic algorithms process. It emphasizes that genetic algorithms work through the components of good solutions - identified as building blocks by Holland [3]. Therefore, from the viewpoint of genetic algorithms, genetic linkage can be used to describe and measure the relation of genes, i.e, how close those genes belonging to a building block are on a chromosome.

With the definition of genetic linkage, we can understand that handling genetic linkage is important to genetic algorithms. Hence, we will discussed the influence of the genetic linkage to genetic algorithms in the next section.

3.2 Why Is Genetic Linkage Important?

In the previous section, we give the explanation of what genetic linkage is and the linkage problem occurs when the crossover operator is used. In this section, we will discussed the importance of genetic linkage and how it affects the performance of genetic algorithms.

In many problems, because of the interactions between parameters, to optimize each dimension of candidate solutions separately could not lead to a global optimum. As described in the previous section, *linkage*, i.e, interrelationships existing between genes needed to be considered when genetic algorithms are used. Moreover, according to Goldberg’s design decomposition theory, building block identification or genetic linkage learning is critical to the success of genetic algorithms.

Goldberg, Korb, and Deb [6] have used an experiment to demonstrate how genetic linkage dictate the success of a simple genetic algorithm. In the experiment, the objective function is composed of 10 uniformly scaled copies of an order-3 fully deceptive function [28, 29] Three types of codings scheme were tested: tightly ordering, loosely ordering, and randomly ordering. For tightly ordering, genes of the same subfunction are arranged to one another on the chromosome. The loosely ordering coding scheme means that all genes are distributed evenly so that genes belonging to the same subfunction are divided by other genes. The randomly ordering indicates that genes are arranged randomly in an arbitrary order. From the experimental results, it is shown that genetic

algorithms perform well when tightly ordering coding scheme, i.e. genes belonging to the same building block are tightly linked, is used. Moreover, some other studies [30, 31, 4] have also reached similar conclusions. With tight building blocks on the chromosome, genetic algorithms could work better.

From the related work described above, it is clear that one of the essential keys for genetic algorithms to success is to handle genetic linkage well. The genetic linkage problem including all kinds of building block processing such as creation, identification and recombination. However, in the real world problem, information about the genetic linkage can not often be known in advance. Thus, studies on handling genetic linkage are extremely important and have long been discussed and recognized in the field of genetic algorithms.

3.3 Genetic Linkage Learning Techniques

The definition and importance of genetic linkage are described in the previous sections. One of the key elements to success in genetic algorithms is to address the linkage problem. In Goldberg's design decomposition, genetic algorithms capable of learning genetic linkages and identifying good building blocks can solve problems quickly, accurately and reliably. Thus, many research efforts have been concentrated on linkage identification of genetic algorithms, which can be broadly classified into three categories[5]:

- Perturbation techniques include the messy GA [6], fast messy GA [32], gene expression messy GA [33], linkage identification by nonlinearity check GA, and linkage identification by monotonicity detection GA [34], and dependency structure matrix driven genetic algorithms(DSMDGA) [35]
- Linkage adaptation techniques [36, 37]
- Probabilistic model building techniques include population-based incremental learning [38], the bivariate marginal distribution algorithm [39], the extended compact GA (eCGA) [40], iterated distribution estimation algorithm [41], Bayesian optimization algorithm(BOA) [42, 43].

For perturbation techniques, the relations between genes are detected by perturbing the value of gene, and analyze the differences before and after the perturbation. For example, linkage identification by nonlinearity check for real-coded GAs (LINC-R) [34] tests nonlinearity by random perturbations on each locus and identifies the linkage configuration according to the nonlinearity relation between locus.

In the linkage adaptation technique, linkage learning GA(LLGA) [36, 37] applies a two-point-like crossover operator to ring-shaped chromosomes and constructs linkages dynamically.

According to the probabilistic model building technique, a probabilistic model is built through analyzing the properties and statistics of the population. As an example, Bayesian optimization algorithm (BOA) [42, 43] constructs a Bayesian network based on the distribution of individuals in a population and identifies linkages.

3.4 Summary

In this chapter, we focus on an important issue in genetic algorithms—genetic linkage. By providing the definition and discussing the importance of the genetic linkage, it is obvious that addressing genetic linkage problem plays an important role for using genetic algorithms. Thus many genetic linkage identification techniques were proposed in the literature. Here we broadly classified them into three categories and had brief explanations for them. The influences of genetic linkage to genetic algorithms inspired us to introduce such a concept to the particle swarm optimizer. Particularly we design a special linkage identification technique which is quite different from those mentioned in this chapter. The proposed framework will be discussed in detail in the next chapter.



Chapter 4

Framework

In this chapter, we will give the overview of the algorithm proposed in this study. This algorithm introduces the recombination operator with the technique of dynamic linkage discovery to particle swarm optimization (PSO) in order to improve the performance of PSO. Dynamic linkage discovery is a costless, effective linkage recognition technique adapting the linkage configuration by utilizing the natural selection without incorporating extra judging criteria irrelevant to the objective function. Furthermore, we employ a specific recombination operator to work with the building blocks identified by dynamic linkage discovery. The following topic will be discussed in this chapter:

- **Dynamic linkage discovery technique:** Describe how the linkage problem is addressed through the natural selection and dynamic linkage discovery.
- **Recombination operator:** In order to make good use of building blocks , a special recombination operator which is similar to multi-parental crossover operators is designed and presented here.
- **Introducing recombination with dynamic linkage discovery in PSO:** A new optimization algorithm which is composed by particle swarm optimizer, dynamic linkage discovery and recombination operator is described in detail.

4.1 Dynamic Linkage Discovery Technique

From chapter 3, the importance of genetic linkage has been discussed. Many mechanisms have been proposed to address the linkage problem. Most linkage identification techniques

were proposed and tested on trap functions [44]. However, there are relatively fewer studies on handling genetic linkage in real number optimization problems. From the survey of linkage learning, Tezuka identified linkage by nonlinearity check on real-coded genetic algorithm [45]. Such a technique has also been incorporated with particle swarm optimizer [12]. Different from this perturbation based linkage identification technique, in this study, we propose the *dynamic linkage discovery* technique. Following paragraph will have a detailed introduction of this mechanism.

In the particle swarm optimization, particles are encoded as real number vectors. With this representation, genetic linkages indicate the interrelation among dimensions. In different stages of optimization process, we consider that the linkage configuration should be different according to the fitness landscape and the population distribution. Hence in the current work, we assume that the relation between different dimensions is dynamically changed along with the optimization process. The linkage configuration should be updated accordingly. For most problems, it is difficult to exactly identified the linkage configuration, especially when the linkage configuration changed dynamically. Instead of incorporating extra artificial criteria for linkage adaptation, we again entrust the task to the mechanism of natural selection. As a consequence, we propose the dynamic linkage discovery technique and we call the PSO combined with recombination and dynamic linkage discovery as PSO-RDL. The dynamic linkage discovery technique is costless, effective, and easy to implement. The idea is to update the linkage configuration according to the fitness feedback. During the whole search process, PSO-RDL first assigns a set of random linkage groups and then adjusts the linkage groups according to the fitness feedback from the optimization problem. If the average fitness value of the current population is improved over a specified threshold, the current linkage configuration is considered appropriate and stays unchanged. Otherwise, the linkage groups will be reassigned at random. A flowchart of the dynamic linkage discovery process is shown in Figure 4.1. An illustration of how we decide the linkage configuration is shown in Figure 4.1.

The dynamic linkage discovery technique is developed based on the idea of natural selection, and can equip with optimization algorithm naturally. In this study, we integrate it into the particle swarm optimizer. Furthermore, we introduce a recombination operator

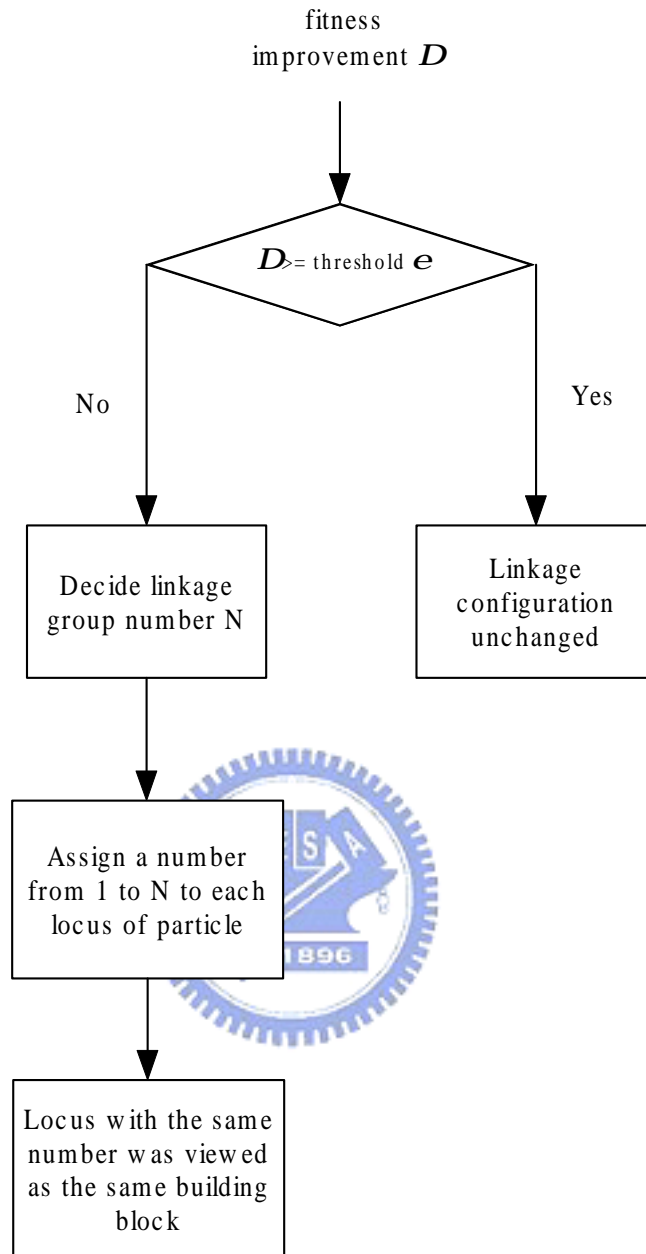


Figure 4.1: The flowchart of the dynamic linkage discovery process. This flow illustrates that every time a particle swarm optimization process is done, the fitness improvement is checked if it improves over the predefined threshold. Based on the results, the linkage configuration can be decided to change or not.

which manipulates the building blocks to construct the new population. The detail of this operator will be given in the next section.

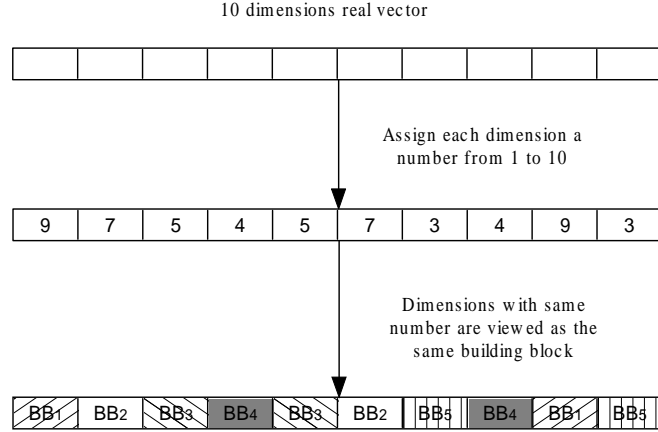


Figure 4.2: An illustration shows that in dynamic linkage discovery, the linkage configuration is assigned at random.

4.2 Recombination Operator

Since Holland's original propose of genetic algorithms, the merits of crossover has been an essential research topic. Instead of the traditional two-parent recombinatory chromosome reproduction, there has been considerable discussion of multi-parent crossover mechanisms [46, 47, 48]. Work by Eiben, Raue and Ruttkay [47] on multi-parental recombination techniques (with fixed numbers of parents) showed that for many standard test-bed functions, N-parental inheritance (with N greater than 2 but less than the size of the population) can be advantageous. From the previous research work, we decide to develop a multi-parental recombination operator for reconstructing the population.

In this study, since we have explicitly identified the linkage group, in order to make good use of linkage information, we design a special recombination operator. The recombination operator is designed according to the idea of multi-parental recombination. In the recombination process, individuals with good fitness are selected and we consider the selected individuals as a building block pool. Every offspring is created by choosing and recombining building blocks from the pool at random. We use this recombination process to generate the whole next population. An illustration of how a new individual is generated is shown as Figure 4.3. By repeating the process shown in Figure 4.3, we can reconstruct a new population in which each particle is composed by the good building blocks. A flowchart of the population reconstructing process is shown in Figure 4.4.

A seemingly similar operator has been proposed by Smith and Fogarty [48]. In [48], the representation on which the recombination operator works takes the form of markers on the chromosome which specify whether or not a gene is linked to its neighbors. Different chromosomes form different numbers of building blocks. However, our recombination operator keeps a global linkage configuration such that every individual in the pool is decomposed into the same building blocks.

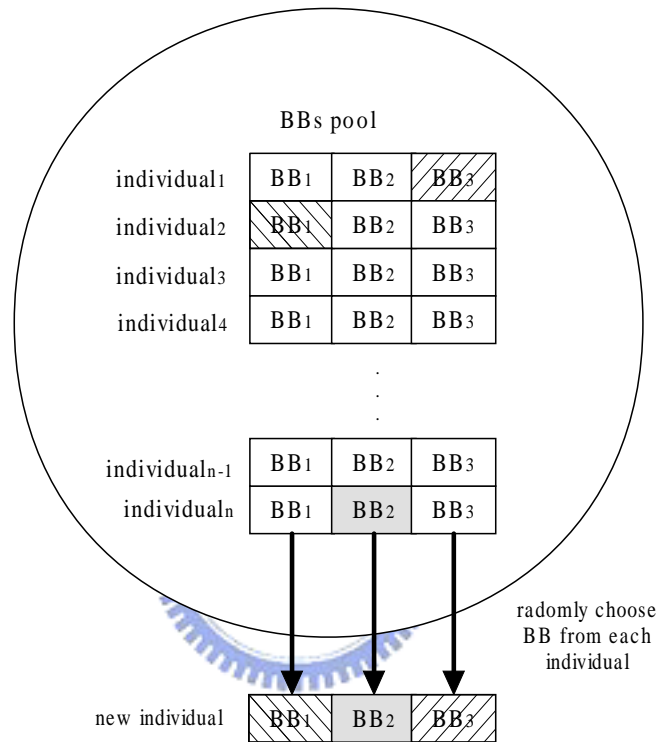


Figure 4.3: The procedure of how a new particle is generated through the recombination operator

In this algorithm, the recombination operator is used to mix building blocks, and to construct a new population. In the beginning of optimization process, this procedure can be viewed as a global search, while the particle swarm optimizer serves as a local searcher that fine tune the building blocks. As the optimization goes on, the population starts to converge and the building blocks become similar. Thus the recombination operator plays as a local searcher at this time. The cooperation between them and the complete flow of this algorithm will be described in the next section.

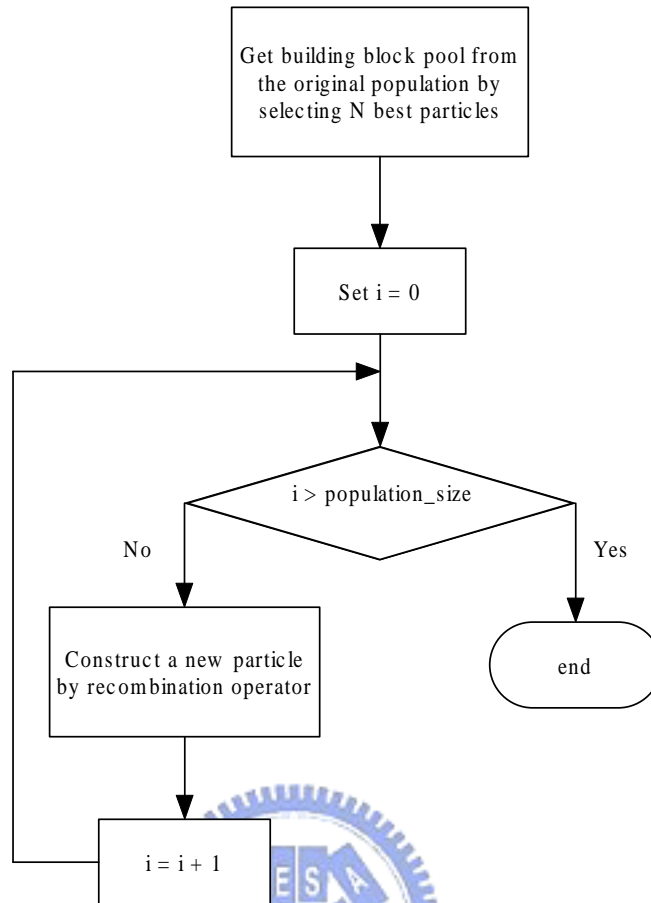


Figure 4.4: The process of constructing new population through recombination operator

4.3 Introducing Recombination with Dynamic Linkage Discovery in PSO

The main purpose in this study is to enhance the PSO's performance by introducing the genetic operator with linkage concept. To achieve this goal, we design the dynamic linkage discovery technique and the corresponding recombination operator. Although there were many variant of the particle swarm optimization proposed in the literature. For the convenience of analyzing and development, in this algorithm, we applied only a modified version of the particle swarm optimization which proposed by Shi and Eberhart [17]. A composition of these three components is described in the following paragraph.

In the proposed algorithm, we repeat the PSO procedure for a certain number of generations, we term such a period a *PSO epoch* in the rest of this report. After each PSO epoch, we select the N best particles from the population to construct the building block pool and conduct the recombination operation according to the building blocks

identified by dynamic linkage discovery. After the recombination process, the linkage discovery step is executed when necessary. We calculate the average fitness of the current epoch, compare the average with the one calculated during last epoch, and check if the improvement is significant enough. When the specified threshold is reached, the current linkage groups are considered suitable and remain unchanged for the next PSO epoch. Otherwise, it is considered that the building blocks do not work well for the current search stage. Thus, the linkage discovery process restarts, and the linkage configuration is randomly reassigned. The pseudo code and flow of the algorithm are shown in Figures 4.5 and 4.6, respectively.

Similar research works have been done in the literature, such as PSO with learning strategy [23, 24] and PSO with adaptive linkage learning [12]. The main difference between the proposed algorithm and them is that we introduce the recombination operator specifically designed to work with the identified building blocks. In addition, we propose a new linkage discovery technique to dynamically adapt the linkage during the search process.



4.4 Summary

In this chapter, we first described how we deal with the genetic linkages. Based on the natural selection concept, a dynamic linkage discovery technique is designed. This technique is designed to identify the linkage configurations in real-parameter optimization problems. In order to make good use of linkages, we then propose the recombination operator. In this algorithm, we introduce the recombination with genetic linkage concept to particle swarm optimization. As a result, a new optimization algorithm is proposed and numerical experiments are also conducted. The experimental results will be shown and discussed in Chapter 5.

PSO with Recombination Operator & Dynamic Linkage Discovery

Step1: Do Finding the linkage group.

1. Generate an integer number N from 1 to D (D : problem dimensions).
2. Assign each dimension an integer number from 1 to N .
3. Dimensions with the same number grouped as the same building block.

Step2: Do PSO algorithm on the population.

1. For each particle
Evaluate fitness value
If the fitness value is better than the best fitness value (pBest) in history set current value as the new pBest
End
2. Choose the particle with the best fitness value of all the particles as the gBest
3. For each particle
Calculate particle's velocity.
Update particle's position.
End
4. Repeat 1 to 3 until maximum iterations is attained.

Step3: Do Recombination to generate next population.

1. Select M best particles from the population.
2. For $i = 1$ to N (N : number of building blocks)
Select i th building block from particles 1 to M .
Put the selected building block to the i th slot of the new generated particle.
End
3. Repeat 2 for S times, S means the swarm size.

Step4: If fitness value improves over the specified threshold, then go to Step2, else go to Step3.

Repeat until the maximum iteration is reached.

Step5: Do local search on the best particle.

Figure 4.5: Pseudocode of PSO-RDL

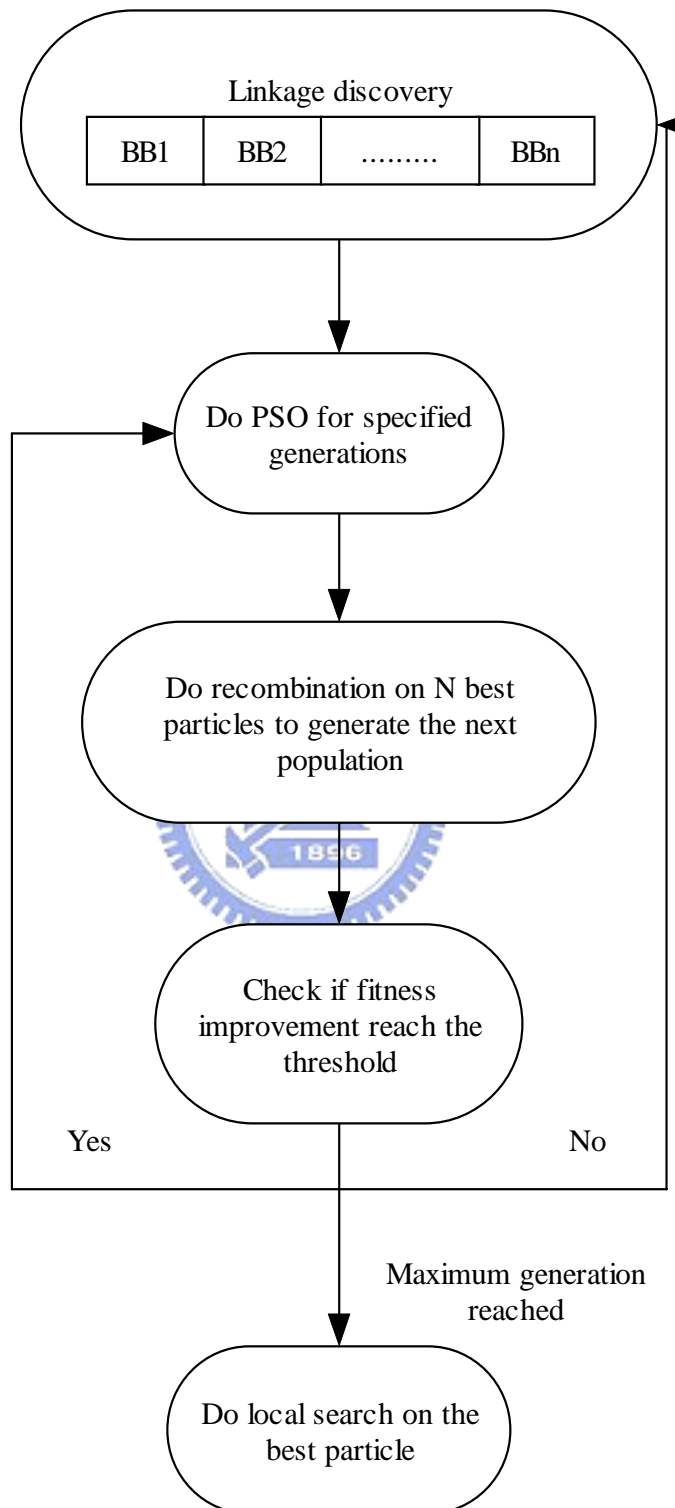


Figure 4.6: The flow of the PSO-RDL

Chapter 5

Experimental Results

Computer simulations are conducted to demonstrate the performance of PSO-RDL. The experiments are focused on the real-valued parameter optimization. The test problems are proposed in the special session on real-parameter optimization in CEC2005 aimed at developing high-quality benchmark functions to be publicly available to the researchers around the world for evaluating their algorithms. The following topics will be covered in this chapter:

- Test Functions: The description of test problems
- Parameter Setting: The parameter settings used in the experiment.
- Experimental Results: Show the numerical results of the experiments as well as the linkage dynamics during optimizing several functions of different characteristics.
- Discussion: Discuss the results and observations from the experiments.

5.1 Test Functions

The newly proposed set of test problems includes 25 functions of different characteristics. Five of them are unimodal problems, and others are multimodal problems [49]. A technical report of detail description of test problems is available at <http://nclab.tw/TR/2005/NCL-TR-2005001.pdf>. All test functions are tested on 10 dimensions in this study. The summary of the 25 functions is shown as follows:

- Unimodal Functions (5):

- F_1 : Shifted Sphere Function
- F_2 : Shifted Schwefel's Problem 1.2
- F_3 : Shifted Rotated High Conditioned Elliptic Function
- F_4 : Shifted Schwefel's Problem 1.2 with Noise in Fitness
- F_5 : Schwefel's Problem 2.6 with Global Optimum on Bounds
- Multimodal Functions (20):
 - Basic Functions (7):
 - * F_6 : Shifted Rosenbrock's Function
 - * F_7 : Shifted Rotated Griewank's Function without Bounds
 - * F_8 : Shifted Rotated Ackley's Function with Global Optimum on Bounds
 - * F_9 : Shifted Rastrigin's Function
 - * F_{10} : Shifted Rotated Rastrigin's Function
 - * F_{11} : Shifted Rotated Weierstrass Function
 - * F_{12} : Schwefel's Problem 2.13
 - Expanded Function (2)
 - * F_{13} : Expanded Extended Griewank's plus Rosenbrock's Function (F8F2)
 - * F_{14} : Shifted Rotated Expanded Scaffer's F6
 - Hybrid Composition Function (11)
 - * F_{15} : Hybrid Composition Function
 - * F_{16} : Rotated Hybrid Composition Function
 - * F_{17} : Rotated Hybrid Composition Function with Noise in Fitness
 - * F_{18} : Rotated Hybrid Composition Function
 - * F_{19} : Rotated Hybrid Composition Function with a Narrow Basin for the Global Optimum
 - * F_{20} : Rotated Hybrid Composition Function with the Global Optimum on the Bounds

- * F_{21} : Rotated Hybrid Composition Function
- * F_{22} : Rotated Hybrid Composition Function with High Condition Number Matrix
- * F_{23} : Non-Continuous Rotated Hybrid Composition Function
- * F_{24} : Rotated Hybrid Composition Function
- * F_{25} : Rotated Hybrid Composition Function without Bounds

The properties and the formulas of these functions are presented in the appendix A. The bias of fitness value for each function $f(x^*)$, the search ranges $[X_{min}, X_{max}]$ and the initialization range of each function are given in Table 5.1. The individuals of global optimum for each function are given in Table B.1.

Table 5.1: Global optimum, search ranges and initialization ranges of the test functions

f	$f(x^*)$	Search Range	Initialization Range
f_1	-450	[-100,100]	[-100,100]
f_2	-450	[-100,100]	[-100,100]
f_3	-450	[-100,100]	[-100,100]
f_4	-450	[-100,100]	[-100,100]
f_5	-310	[-100,100]	[-100,100]
f_6	390	[-100,100]	[-100,100]
f_7	-180	$[-\infty, \infty]$	[0,600]
f_8	-140	[-32,32]	[-32,32]
f_9	-330	[-5,5]	[-5,5]
f_{10}	-330	[-5,5]	[-5,5]
f_{11}	90	[-0.5,0.5]	[-0.5,0.5]
f_{12}	-460	$[-\pi, \pi]$	$[-\pi, \pi]$
f_{13}	-130	[-5,5]	[-5,5]
f_{14}	-300	[-100,100]	[-100,100]
f_{15}	120	[-5,5]	[-5,5]
f_{16}	120	[-5,5]	[-5,5]
f_{17}	120	[-5,5]	[-5,5]
f_{18}	10	[-5,5]	[-5,5]
f_{19}	10	[-5,5]	[-5,5]
f_{20}	10	[-5,5]	[-5,5]
f_{21}	360	[-5,5]	[-5,5]
f_{22}	360	[-5,5]	[-5,5]
f_{23}	360	[-5,5]	[-5,5]
f_{24}	260	$[-\infty, \infty]$	[-5,5]
f_{25}	260	$[-\infty, \infty]$	[2,5]

Table 5.2: Parameter setting in this numerical experiments

Parameter Setting	
Parameter description	Value
Swarm size	20
Inertia weight (w)	$w \in [0.6, 0.9]$
Acceleration rate of cognitive part(C_1)	$C_1 \in [0.5, 2.0]$
Acceleration rate of social part(C_2)	$C_2 \in [0.5, 2.0]$
Maximum velocity	25% of the search range
Size of selected particle for recombination	25% of the swarm size
Improvement threshold	2% over the previous best fitness

Numerical test problems described above were simulated to evaluate the performance of the proposed algorithm. This benchmark with multiple types of functions such as unimodal, multimodal, expanded and composition functions, so that the strength and weakness of the algorithm could be analyzed comprehensively.

5.2 Parameter Setting

The parameter setting in this study is described as follows:

The number of particles is set to 20, $0.6 \leq w \leq 0.9$, $0.5 \leq \vec{\varphi}_1 \leq 2.0$, $0.5 \leq \vec{\varphi}_2 \leq 2.0$, and V_{\max} restricts the particles' velocity, where V_{\max} is equal to 25% of the initialization range. N , the number of particles selected for the recombination, is set to 25% of the swarm size. The threshold which decides if the linkage configuration should be changed is set to 2% of the previous best fitness value. A list of the parameter setting is shown in Table 5.2.

5.3 Experimental Results

The complete experimental results are listed in Tables 5.3, 5.4, 5.5, 5.6 and 5.7. According to the definition in the special session, PSO-RDL successfully solved problems 1, 2, 4, 5, 6, 7 and 12 in the experimental results. Moreover, comparable results are achieved in solving problems 3, 8, 11, 13 and 14. Unfortunately, PSO-RDL failed to solve problems 9, 10 and 15-25. Table 5.8, 5.9, 5.10, 5.11 and 5.12 show the experimental results compared with other evolutionary algorithms proposed in the special session. Table 5.13 gives the

number of successfully solved problems. From these comparisons, it can be observed that PSO-RDL has a good performance for most problems. Table C.1 shows the solution found by PSO-RDL for each function in this benchmark. Figures 5.1, 5.2, 5.3 and 5.4 show how the dynamic linkage discovery technique changes the linkage configuration during the optimization process. Detailed discussion on the experimental results is presented in the next section.

5.4 Discussion

From the experimental results listed in Table 5.3, it can be considered that the proposed algorithm is able to provide good results for the benchmark. The first five functions are unimodal functions. Function 1 is shifted sphere function, Function 2 is shifted Schwefel's problem 1.2, and Function 3 is shifted rotated high condition elliptic function. These three functions have different condition numbers which make Function 3 much harder than Functions 1 and 2. Function 4 is shifted Schwefel's problem 1.2 with noise in fitness. Function 5 is Schwefel's problem 2.6 with global optimum on bounds. From the results, we can observe that PSO-RDL reaches the predefined error tolerance level for Functions 1, 2, 4, and 5. For Function 3, PSO-RDL achieves an error of $1e-4$ but does not meet the $1e-6$ criterion. It may be caused by the multiplier 10^6 in the objective function which greatly amplifies the error. In summary, PSO-RDL provides a sufficiently good performance for the unimodal functions in this benchmark.

Functions 6-14 are multimodal problems. Function 6 is shifted Rosenbrock's function, a problem with a very narrow valley from the local optimum to the global optimum, and solved by PSO-RDL. Function 7 is shifted rotated Griewank's function without bounds, and this function makes the search easily away from the global optimum. Fortunately, PSO-RDL solved it twice in 25 trials and can achieve a comparable result in average for this function. Function 8 is shifted rotated Ackley's function with global optimum on bounds, which has a very narrow global basin and half of the dimensions of this basin are on the boundaries. Hence, the search algorithm cannot easily find the global basin when the recombination operator is used. The PSO-RDL failed on this problem in all 25

FES	1	2	3	4	5	
1E+03	1st(Min)	5.6975E-02	2.3397E+01	4.8244E+05	1.6839E+02	1.1410E+00
	7th	2.2013E+00	1.5546E+02	1.3215E+06	8.4541E+02	1.1389E+01
	13th(Median)	1.2539E+01	2.9970E+02	2.0604E+06	2.4966E+03	3.4230E+01
	19th	1.0878E+02	5.8835E+02	3.6532E+06	4.9921E+03	1.3766E+03
	25th(Max)	6.4912E+02	7.7404E+03	8.6580E+07	1.8695E+04	9.4677E+03
1E+04	mean	1.1711E+02	9.8902E+02	7.4499E+06	4.3445E+03	1.7836E+03
	Std	2.0425E+02	1.7074E+03	1.7630E+07	5.2837E+03	3.2111E+03
	1st(Min)	0.0000E+00	1.8605E-10	1.9327E+04	6.3928E+00	0.0000E+00
1E+05	7th	5.6843E-14	5.1810E-09	4.9197E+04	1.1922E+02	0.0000E+00
	13th(Median)	5.6843E-14	4.0501E-08	1.0152E+05	4.2557E+02	1.2733E-10
	19th	1.1369E-13	9.8918E-08	1.7427E+05	9.0310E+02	2.0135E-07
	25th(Max)	1.0800E-12	5.6686E-07	6.7518E+05	1.5376E+04	4.5570E-04
	mean	1.7053E-13	7.9402E-08	1.5610E+05	1.4138E+03	2.3191E-05
1E+05	Std	2.5527E-13	1.2049E-07	1.7142E+05	3.1880E+03	9.2921E-05
	1st(Min)	0.0000E+00	5.6843E-14	3.8107E-04	1.4211E-12	0.0000E+00
	7th	0.0000E+00	5.6843E-14	4.6282E-04	5.5718E-10	0.0000E+00
	13th(Median)	0.0000E+00	1.1369E-13	4.7232E-04	4.3663E-09	0.0000E+00
	19th	5.6843E-14	1.7053E-13	4.7236E-04	1.8184E-08	3.6380E-12
1E+05	25th(Max)	5.6843E-14	7.9581E-13	1.2360E+00	3.3329E-06	2.6100E-06
	mean	2.5011E-14	1.7735E-13	9.6848E-02	2.4686E-07	2.0943E-07
	Std	2.8798E-14	2.0318E-13	3.3375E-01	7.0712E-07	7.2248E-07

Table 5.3: Best function error values achieved when FES = 1e+3, 1e+4, and 1e+5 for functions 1-5. The predefined error is 1e-6 for these five functions. These functions are all unimodal problems, and PSO-RDL successfully solved functions 1, 2, 4, and 5. Comparable results for function 3 were obtained.

FES	6	7	8	9	10
1st(Min)	3.7957E+01	1.2672E+03	2.0477E+01	1.6208E+01	1.8937E+01
7th	1.2497E+03	1.2672E+03	2.0599E+01	2.8014E+01	4.0586E+01
13th(Median)	1.0455E+04	1.2678E+03	2.0703E+01	3.0172E+01	4.9847E+01
19th	2.4273E+05	1.2865E+03	2.0751E+01	4.3307E+01	5.9570E+01
25th(Max)	2.6047E+06	1.9419E+03	2.0991E+01	6.5808E+01	7.0826E+01
mean	3.9211E+05	1.3463E+03	2.0708E+01	3.4992E+01	5.0378E+01
Std	7.7244E+05	1.6289E+02	1.1955E-01	1.3668E+01	1.3604E+01
1st(Min)	5.5079E-09	5.1871E-01	2.0008E+01	3.9798E+00	1.5919E+01
7th	3.5402E-02	9.1688E-01	2.0022E+01	1.5919E+01	2.0894E+01
13th(Median)	1.6479E+00	9.5936E-01	2.0070E+01	2.3879E+01	3.5818E+01
19th	1.1236E+02	9.9956E-01	2.0093E+01	2.9849E+01	5.4722E+01
25th(Max)	5.8179E+02	1.3723E+00	2.0282E+01	6.5667E+01	6.9647E+01
mean	8.7012E+01	9.7299E-01	2.0084E+01	2.3933E+01	3.8564E+01
Std	1.6549E+02	1.8345E-01	7.7340E-02	1.2693E+01	1.7976E+01
1st(Min)	1.2784E-10	1.6993E-10	2.0000E+01	1.9899E+00	1.5919E+01
7th	9.0580E-10	1.4780E-02	2.0000E+01	7.9597E+00	2.0894E+01
13th(Median)	2.9976E-09	4.9257E-02	2.0000E+01	1.0945E+01	3.5818E+01
19th	2.6304E-07	1.0577E-01	2.0000E+01	1.6914E+01	5.4722E+01
25th(Max)	3.9866E+00	1.5520E-01	2.0001E+01	3.6813E+01	6.9647E+01
mean	9.5678E-01	5.7316E-02	2.0000E+01	1.2497E+01	3.8564E+01
Std	1.7377E+00	4.6600E-02	2.1685E-04	8.1748E+00	1.7976E+01

Table 5.4: Best function error values achieved when FES = 1e+3, 1e+4, and 1e+5 for functions 6-10. The predefined error is 1e-2 for these five functions. The functions are all multimodal problems, and PSO-RDL successfully solved function 6 and gave comparable results on functions 7 and 8. However, worse results were obtained on functions 9 and 10 due to the large number of local optima.

FES	11	12	13	14	15
1st(Min)	5.7283E+00	1.5379E+02	1.5402E+00	3.4972E+00	2.2397E+02
7th	6.8054E+00	9.7877E+02	2.8680E+00	3.9300E+00	3.1785E+02
13th(Median)	8.2136E+00	6.8587E+03	3.3114E+00	4.1025E+00	4.1720E+02
19th	9.3260E+00	1.7770E+04	4.0523E+00	4.1570E+00	4.9406E+02
25th(Max)	9.8271E+00	3.8007E+04	4.9076E+00	4.4537E+00	7.5028E+02
mean	8.0589E+00	1.0718E+04	3.3269E+00	4.0285E+00	4.2130E+02
Std	1.2377E+00	1.1550E+04	8.4003E-01	2.3182E-01	1.3274E+02
1st(Min)	2.8733E+00	9.1041E-05	3.7507E-01	3.2027E+00	1.2408E+02
7th	4.8607E+00	4.1626E-01	7.4933E-01	3.6311E+00	2.6996E+02
13th(Median)	5.4240E+00	6.9468E+00	1.1691E+00	3.8062E+00	3.5151E+02
19th	6.6473E+00	2.7252E+01	1.6247E+00	4.0219E+00	4.5882E+02
25th(Max)	9.0392E+00	1.6936E+03	8.9953E-01	4.4112E+00	6.0597E+02
mean	5.7585E+00	1.5910E+02	1.1972E+00	3.7981E+00	3.5718E+02
Std	1.5200E+00	4.4750E+02	5.7742E-01	3.2971E-01	1.3133E+02
1st(Min)	2.8287E+00	1.0967E-09	3.3823E-01	3.1359E+00	1.0065E+02
7th	4.8524E+00	4.9112E-09	6.1622E-01	3.5812E+00	1.3786E+02
13th(Median)	5.4229E+00	7.5948E-08	7.4910E-01	3.8060E+00	2.0857E+02
19th	5.9537E+00	4.4376E-06	1.1119E+00	4.0137E+00	4.2923E+02
25th(Max)	9.0392E+00	1.6936E+03	1.9851E+00	4.4108E+00	6.0596E+02
mean	5.5754E+00	1.3125E+02	8.8732E-01	3.7796E+00	2.7113E+02
Std	1.4161E+00	4.5006E+02	4.0601E-01	3.4364E-01	1.5881E+02

Table 5.5: Best function error values achieved when FES = 1e+3, 1e+4, and 1e+5 for functions 11-14. The predefined error is 1e-2 for these four functions. The functions are all multimodal problems, and functions 13 and 14 are extended functions. PSO-RDL successfully solved function 12 and obtained comparable results on functions 11, 13, and 14.

FES	16	17	18	19	20
1st(Min)	1.3947E+02	1.4536E+02	8.0049E+02	8.0270E+02	8.2370E+02
7th	1.8825E+02	2.1723E+02	1.0163E+03	9.9133E+02	9.7490E+02
13th(Median)	2.2334E+02	2.5107E+02	1.0489E+03	1.0180E+03	1.0230E+03
19th	2.5106E+02	2.8939E+02	1.1313E+03	1.0836E+03	1.0524E+03
25th(Max)	9.9560E+02	7.1139E+02	1.1991E+03	1.1178E+03	1.1251E+03
mean	2.5766E+02	2.7683E+02	1.0489E+03	1.0102E+03	9.9493E+02
Std	1.7382E+02	1.1460E+02	1.0500E+02	8.7227E+01	8.6921E+01
1st(Min)	1.2782E+02	1.1722E+02	8.0000E+02	6.9578E+02	8.0000E+02
7th	1.5093E+02	1.5412E+02	9.8800E+02	9.7805E+02	8.3695E+02
13th(Median)	1.7354E+02	1.8304E+02	1.0367E+03	1.0134E+03	1.0090E+03
19th	2.1744E+02	2.2036E+02	1.1182E+03	1.0517E+03	1.0416E+03
25th(Max)	9.8807E+02	5.9989E+02	1.1956E+03	1.1084E+03	1.1005E+03
mean	2.2028E+02	2.0832E+02	1.0248E+03	9.9023E+02	9.6112E+02
Std	1.7388E+02	1.0039E+02	1.2107E+02	1.0272E+02	1.0517E+02
1st(Min)	1.2782E+02	1.1866E+02	8.0000E+02	6.9526E+02	8.0000E+02
7th	1.5093E+02	1.6793E+02	9.8800E+02	9.7674E+02	8.2436E+02
13th(Median)	1.7342E+02	2.1170E+02	1.0362E+03	1.0021E+03	1.0090E+03
19th	2.1635E+02	2.3655E+02	1.1182E+03	1.0480E+03	1.0413E+03
25th(Max)	9.8632E+02	6.1468E+02	1.1905E+03	1.1084E+03	1.1002E+03
mean	2.1958E+02	2.2222E+02	1.0227E+03	9.8499E+02	9.5905E+02
Std	1.7350E+02	1.0019E+02	1.1883E+02	1.0189E+02	1.0567E+02

Table 5.6: Best function error values achieved when FES = 1e+3, 1e+4, and 1e+5 for functions 16-20. The predefined error is 1e-2 for Function 16 and 1e-1 for the other four functions.

FES	21	22	23	24	25	
1E+03	1st(Min)	5.0026E+02	7.8706E+02	5.5401E+02	8.1573E+02	1.7342E+03
	7th	1.0561E+03	9.1841E+02	1.1419E+03	9.7928E+02	1.7491E+03
	13th(Median)	1.2217E+03	9.5652E+02	1.2438E+03	1.0243E+03	1.7682E+03
	19th	1.2713E+03	9.9860E+02	1.2850E+03	1.2637E+03	1.7993E+03
	25th(Max)	1.2814E+03	1.0633E+03	1.3264E+03	1.3325E+03	1.9582E+03
	mean	1.0805E+03	9.4973E+02	1.1132E+03	1.0796E+03	1.7934E+03
	Std	2.7233E+02	7.8269E+01	2.7359E+02	1.6030E+02	6.5746E+01
	1st(Min)	3.0000E+02	7.5236E+02	4.2517E+02	5.0000E+02	1.7164E+03
	7th	1.0450E+03	8.2582E+02	1.1417E+03	5.0456E+02	1.7290E+03
	13th(Median)	1.1582E+03	9.1275E+02	1.2197E+03	9.6491E+02	1.7401E+03
1E+04	19th	1.1899E+03	9.3397E+02	1.2594E+03	1.1142E+03	1.7478E+03
	25th(Max)	1.2658E+03	1.0209E+03	1.3195E+03	1.2883E+03	1.8727E+03
	mean	9.9795E+02	8.9147E+02	1.0874E+03	8.8951E+02	1.7452E+03
	Std	3.3016E+02	7.0707E+01	2.9207E+02	3.0372E+02	2.9465E+01
	1st(Min)	3.0000E+02	7.5022E+02	4.2517E+02	2.0000E+02	1.7372E+03
	7th	1.0450E+03	8.0458E+02	1.1417E+03	5.0000E+02	1.7495E+03
	13th(Median)	1.1582E+03	9.1047E+02	1.1881E+03	5.0462E+02	1.7554E+03
	19th	1.1878E+03	9.3324E+02	1.2578E+03	1.0213E+03	1.7647E+03
	25th(Max)	1.2647E+03	1.0189E+03	1.2846E+03	1.2995E+03	1.8087E+03
	mean	9.9429E+02	8.8666E+02	1.0776E+03	7.2004E+02	1.7588E+03
Std	3.2749E+02	7.1176E+01	2.8681E+02	3.9639E+02	1.5416E+01	

Table 5.7: Best function error values achieved when FES = 1e+3, 1e+4, and 1e+5 for functions 21-25. The predefined error is 1e-1 for these four functions.

Table 5.8: Results compared with other evolutionary algorithms for functions 1-5

PSO-RDL	mean	2.5011E-14	1.7735E-13	9.6848E-02	2.4686E-07	2.0943E-07
	std.	2.8798E-14	2.0318E-13	3.3375E-01	7.0712E-07	7.2248E-07
PSO [50]	mean	0.0000E+00	0.0000E+00	2.2903E+05	0.0000E+00	2.6278E+03
	std.	0.0000E+00	0.0000E+00	4.0637E+05	0.0000E+00	4.8124E+02
DMS-PSO [51]	mean	0.0000E+00	1.2960E-13	7.0064E-09	1.8851E-03	1.1383E-06
	std.	0.0000E+00	1.5612E-13	2.6589E-09	1.8932E-03	2.1828E-06
SPC-PNX [52]	mean	8.8967E-09	9.6317E-09	1.0806E+05	9.3788E-09	9.1535E-09
	std.	9.3915E-10	3.2989E-10	8.7160E+04	6.3274E-10	6.3186E-10
DE [53]	mean	0.0000E+00	0.0000E+00	1.9400E-06	9.0900E-15	0.0000E+00
	std.	0.0000E+00	0.0000E+00	4.6300E-06	3.1500E-14	0.0000E+00
SaDE [54]	mean	0.0000E+00	1.0459E-13	1.6720E-05	1.4182E-05	1.2300E-02
	std.	0.0000E+00	5.1124E-13	3.1196E-05	7.0912E-05	1.4600E-02
restartCMA-ES [55]	mean	5.2000E-09	4.7000E-09	5.6000E-09	5.0200E-09	6.5800E-09
	std.	1.9400E-09	1.5600E-09	1.9300E-09	1.7100E-09	2.1700E-09

Table 5.9: Results compared with other evolutionary algorithms for functions 6-10

PSO-RDL	mean	9.5678E-01	5.7316E-02	2.0000E+01	1.2497E+01	3.8564E+01
	std.	1.7377E+00	4.6600E-02	2.1685E-04	8.1748E+00	1.7976E+01
PSO [50]	mean	1.5529E+01	1.7184E-01	2.0303E+01	2.7859E+00	1.8582E+01
	std.	3.0962E+01	9.9578E-02	8.1576E-02	1.5732E+00	7.7603E+00
DMS-PSO [51]	mean	6.8925E-08	4.5189E-02	2.0000E+01	0.0000E+00	3.6217E+00
	std.	3.1904E-07	3.2611E-02	5.5382E-09	0.0000E+00	8.5509E-01
SPC-PNX [52]	mean	1.8909E+01	8.2610E-02	2.0991E+01	4.0196E+00	7.3044E+00
	std.	3.9977E+01	6.2418E-02	5.7946E-02	2.2703E+00	5.2116E+00
DE [53]	mean	1.5900E-01	1.4600E-01	2.0400E+01	9.5500E-01	1.2500E+01
	std.	7.9700E-01	1.3800E-01	7.5800E-02	9.7300E-01	7.9600E+00
SaDE [54]	mean	1.1987E-08	1.9900E-02	2.0000E+01	0.0000E+00	4.9685E+00
	std.	1.9282E-08	1.0700E-02	5.3901E-08	0.0000E+00	1.6918E+00
restartCMA-ES [55]	mean	4.8700E-09	3.3100E-09	2.0000E+01	2.3900E-01	7.9600E-02
	std.	1.6600E-09	2.0200E-09	3.8900E-03	4.3400E-01	2.7500E-01

Table 5.10: Results compared with other evolutionary algorithms for functions 11-15

PSO-RDL	mean	5.5754E+00	1.3125E+02	8.8732E-01	3.7796E+00	2.7113E+02
	std.	1.4161E+00	4.5006E+02	4.0601E-01	3.4364E-01	1.5881E+02
PSO [50]	mean	4.1633E+00	3.3058E+02	6.2393E-01	3.0386E+00	
	std.	1.3470E+00	5.0884E+02	2.2529E-01	3.8894E-01	
DMS-PSO [51]	mean	4.6229E+00	2.4007E+00	3.6865E-01	2.3601E+00	4.8539E+00
	std.	5.8400E-01	4.3602E+00	5.6411E-02	3.3750E-01	1.3415E+01
SPC-PNX [52]	mean	1.9098E+00	2.5951E+02	8.3793E-01	3.0456E+00	2.5376E+02
	std.	1.1598E+00	4.8933E+02	2.6913E-01	4.3662E-01	1.5052E+02
DE [53]	mean	8.4700E-01	3.1700E+01	9.7700E-01	3.4500E+00	2.5900E+02
	std.	1.4000E+00	1.4200E+02	4.6700E-01	4.4000E-01	1.8300E+02
SaDE [54]	mean	4.8909E+00	4.5011E-07	2.2020E-01	2.9153E+00	3.2000E+01
	std.	6.6190E-01	8.5062E-07	4.1100E-02	2.0630E-01	1.1076E+02
restartCMA-ES [55]	mean	9.3400E-01	2.9300E+01	6.9600E-01	3.0100E+00	2.2800E+02
	std.	9.0000E-01	1.4200E+02	1.5000E-01	3.4900E-01	6.8000E+01

Table 5.11: Results compared with other evolutionary algorithms for functions 16-20

PSO-RDL	mean	2.1958E+02	2.2222E+02	1.0227E+03	9.8499E+02	9.5905E+02
	std.	1.7350E+02	1.0019E+02	1.1883E+02	1.0189E+02	1.0567E+02
DMS-PSO [51]	mean	9.4756E+01	1.1009E+02	7.6067E+02	7.1430E+02	8.2196E+02
	std.	1.0086E+01	4.3453E+00	1.8458E+02	2.0105E+02	4.5874E+01
SPC-PNX [52]	mean	1.0962E+02	1.1898E+02	4.3956E+02	3.8000E+02	4.4000E+02
	std.	9.8654E+00	1.0707E+01	2.2494E+02	1.8708E+02	2.2913E+02
DE [53]	mean	1.1300E+02	1.1500E+02	4.0000E+02	4.2000E+02	4.6000E+02
	std.	1.8000E+01	2.0100E+01	2.0400E+02	2.1800E+02	2.3800E+02
SaDE [54]	mean	1.0121E+02	1.1406E+02	7.1939E+02	7.0494E+02	7.1302E+02
	std.	6.1686E+00	9.9679E+00	2.0852E+02	1.9040E+02	2.0134E+02
restartCMA-ES [55]	mean	9.1300E-01	1.2300E+02	3.3200E+02	2.2600E+02	3.0000E+02
	std.	3.4900E+00	2.0900E+01	1.1200E+02	9.9300E+01	0.0000E+00

Table 5.12: Results compared with other evolutionary algorithms for functions 21-25

PSO-RDL	mean	9.9429E+02	8.8666E+02	1.0776E+03	7.2004E+02	1.7588E+03
	std.	3.2749E+02	7.1176E+01	2.8681E+02	3.9639E+02	1.5416E+01
DMS-PSO [51]	mean	5.3600E+02	6.9242E+02	7.3034E+02	2.2400E+02	3.6571E+02
	std.	2.1772E+02	1.5647E+02	1.6620E+02	8.3066E+01	1.5096E+02
SPC-PNX [52]	mean	6.8006E+02	7.4927E+02	5.7591E+02	2.0000E+02	4.0641E+02
	std.	2.6873E+02	9.3700E+01	8.2207E+01	0.0000E+00	2.3848E-01
DE [53]	mean	4.9200E+02	7.1800E+02	5.7200E+02	2.0000E+02	9.2300E+02
	std.	4.0000E+01	1.5800E+02	4.4800E+01	0.0000E+00	3.4000E-01
SaDE [54]	mean	4.6400E+02	7.3190E+02	6.6406E+02	2.0000E+02	3.7586E+02
	std.	1.5780E+02	9.1523E+01	1.5266E+02	0.0000E+00	3.1453E+00
restartCMA-ES [55]	mean	5.0000E+02	7.2900E+02	5.5900E+02	2.0000E+02	3.7400E+02
	std.	0.0000E+00	3.1800E-13	6.8600E+00	3.2400E-11	3.2200E+00

Table 5.13: Problems solved in different evolutionary algorithms

Method	Unimodal Functions(5)	Basic Multimodal Functions (7)	Expanded Function (2)	Hybrid Composition Function (11)
PSO-RDL	1,2,4,5 (4)	6,7,12 (3)		*
PSO	1,2,4,5 (4)	6,7,12 (3)	*	*
SPC-PNX	1,2,4,5 (4)	6,7,11 (3)	*	*
Sa-DE	1,2,4 (3)	9,12 (2)	15	*
DE	1,2,3,4,5 (5)	6,9 (2)	*	*
DMS-PSO	1,2,3,5 (4)	6,7,9,12 (4)	15	*
LR-CMA-ES	1,2,3,4,5 (5)	6,7,12 (3)	*	*

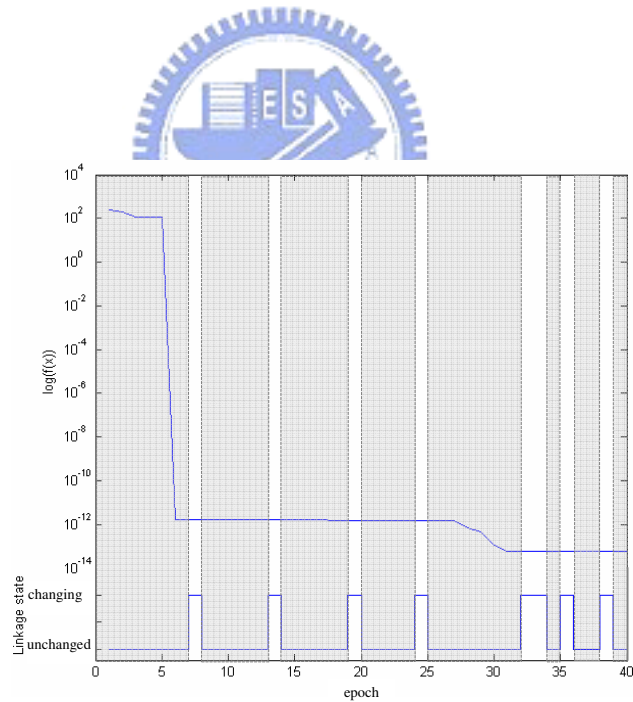


Figure 5.1: Fitness convergence and linkage dynamics of the Sphere function. A unimodal function which PSO-RDL solved successfully. The gray area in the figure represents the proper building blocks can improve the fitness and stay unchanged. Once the building blocks do not work well, the linkage configuration will change until the next suitable set is found.

runs. Functions 9, 10, and 11 are shifted Rastrigin’s function, shifted rotated Rastrigin’s function, and shifted rotated Weierstrass function, respectively, all of them have a huge number of local optima. The PSO-RDL has a relatively bad performance on the first two problems comparing with traditional PSO [50] and DMS-PSO [51]. Comparable results were obtained for Function 11. It may be because when the number of local optima is huge, the dissimilar individuals would likely to have similar fitness values. Although they could provide good building blocks, when different building blocks are combined to create new individuals, the offspring could have worse fitness values instead. As long as the building blocks cannot be identified correctly, the genetic operator cannot work well, either. Function 12 is Schwefel’s problem, and PSO-RDL achieves a 80% success rate. Functions 13 and 14 are extended functions, and the PSO-RDL produces comparable results in solving these two functions.

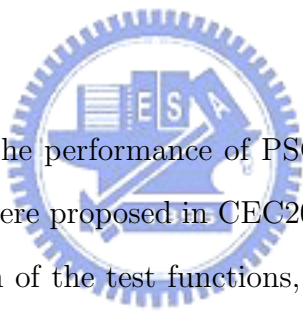
Function 15-25 are eleven composition functions. They are all built up with basic functions. They give a big challenge to any search algorithm. PSO-RDL has failed in all the experiment runs. These composition functions possess a huge number of local optima, thus made PSO-RDL performs worse due to the similar reason discussed for Functions 9 and 10. The failure of identifying building block makes the PSO-RDL also failed in the search process.

Observe the fitness convergence and linkage dynamics in Figures 5.1, 5.2, 5.3, and 5.4. The gray areas represent the time frames when a proper linkage configuration can assist the optimization process. When the current linkage groups are not suitable, i.e. the linkage configuration cannot assist the search, the linkage group composition will start to vibrate for some iterations until the next proper set of linkage groups is found. The phenomenon can explain the assumption that the building block’s composition is dynamically changed during the search process in the real-parameter optimization problem. Thus, it is reasonable that we hand over the linkage adaptation to the mechanism of natural selection. Moreover, Figure 7 shows the function with a large number of local optima and PSO-RDL failed. It is clearly that the linkage configuration keeps changing all the time. As discussed above, this phenomenon indicates that when the function has a large number of local optima, it is hard to recognize the building blocks because totally

different individuals may have similar fitness values. In such a case, different individual may provide their own good building blocks, but worse individuals may still be created by recombining these incompatible pieces of solutions.

Focusing on the time ratio of the linkage status (changing vs. unchanged), we can observe that for Figures 5.1 and 5.2, the linkage configuration stay unchanged most of the time. Correspondingly, the proposed algorithm provide good results on these two functions. On the contrary, the linkage configuration keeps changing in the Figures 5.3 and 5.4. Thus, our algorithm do not work very well on these two functions, although we mentioned that PSO-RDL can obtain comparable results on the shifted expanded Griewank's plus Rosenbrock's function. Because there does not exist a very efficient algorithm for this problem so far. Hence, we can conclude that when the linkage configuration changes too often, the algorithm will fail to solve the problem with a high probability.

5.5 Summary



In this chapter, we evaluate the performance of PSO-RDL by conducting the search on 25 numerical functions that were proposed in CEC2005 special session on real-parameter optimization. The description of the test functions, parameter setting and experimental results were given in the above context. We also discussed the strength and weakness of the PSO-RDL by analyzed the search result and linkage dynamics. From the results, it is considered that PSO-RDL can work well and produce a better performance than the traditional particle swarm optimizer. Further, from the observation of linkage dynamics, it is considered that dynamic linkage discovery and recombination operator do improve the performance of the particle swarm optimizer when the building blocks are successfully identified. A detailed discussion and summary of this research work will be provided in the next chapter.

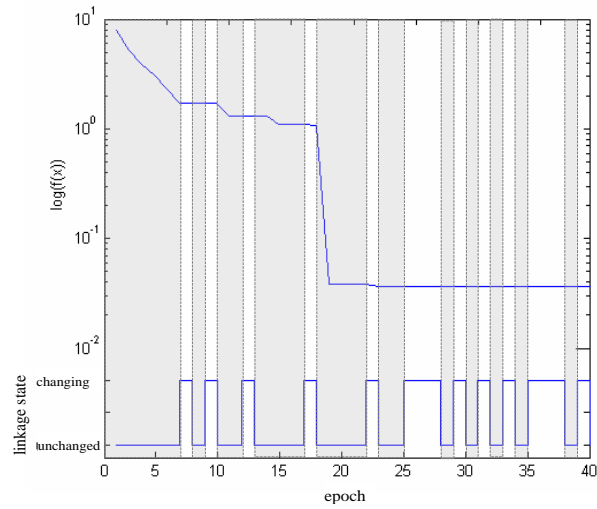


Figure 5.2: Fitness convergence and linkage dynamics of the Shifted Rotated Griewank's function. A multimodal function which PSO-RDL produced comparable results. The gray area in the figure represents the proper building blocks can improve the fitness and stay unchanged. Once the building blocks do not work well, the linkage configuration will change until the next suitable set is found.

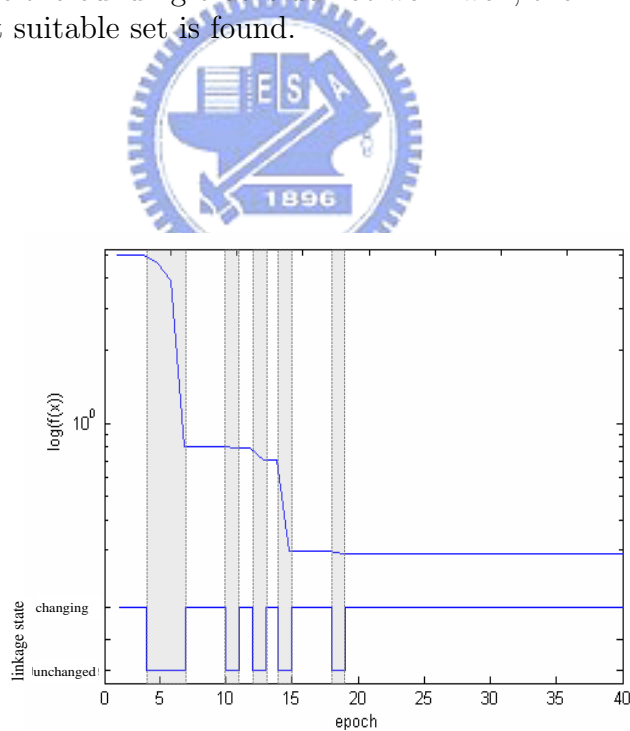


Figure 5.3: Fitness convergence and linkage dynamics of the Shifted Expanded Griewank's plus Rosenbrock's function. A multimodal function which PSO-RDL produced comparable results. The gray area in the figure represents the proper building blocks can improve the fitness and stay unchanged. Once the building blocks do not work well, the linkage configuration will change until the next suitable set is found.

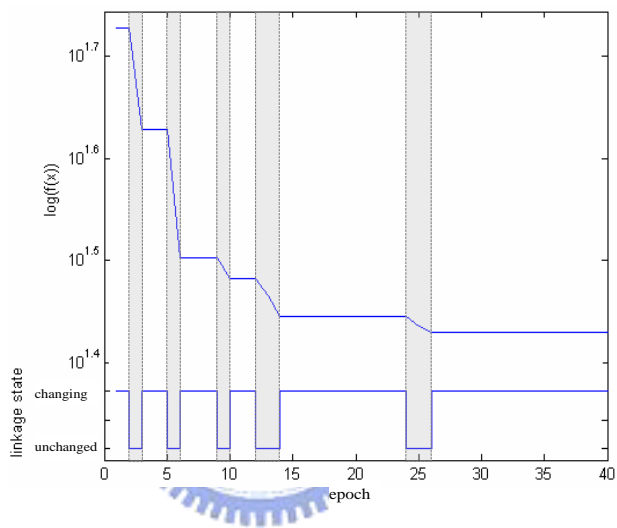


Figure 5.4: Fitness convergence and linkage dynamics of the Shifted Rastrigin's function. A multimodal function with large number of local optima and PSO-RDL failed to solve. The gray area in the figure represents the proper building blocks can improve the fitness and stay unchanged. Once the building blocks do not work well, the linkage configuration will change until the next suitable set is found.

Chapter 6

Real-world Applications

In this chapter, we proposed the algorithm to real-world applications. We focus on solving the economic dispatch (ED) problem which is an significant topic in the power system. There are lots of studies working on the ED problem including optimization and equation modeling. Due to the importance of the ED problem, here in this study, we try to solve this problem with the proposed efficient search algorithm. The following topics will be covered in this chapter:

- Economic Dispatch Problem: Briefly introduce the purpose and formulations of the ED problem.
- Our solution: The PSO-RDL is applied to solve the economic dispatch problem, and the constraint handling technique is described as well.
- Experimental results: The result of 3-generators, 40-generators economic dispatch problems and comparison with other search algorithm are listed in this section.



6.1 Economic Dispatch Problem

With the development of modern power systems, the economic dispatch problem has received an increasing attention. Economic dispatch is essential for real-time control of power system operation. It consists of allocating the total generation required among the available thermal generating units, assuming that a thermal unit commitment is previously determined. The objective aims to minimize the fuel cost subject to the physical and operational constraints.

The economic dispatch problem is to find the optimal combination of power generations that minimizes the total generation cost while satisfying equality and inequality constraints. To model the economic dispatch problem, a simplified cost function of each generator which is represented as a quadratic function is as follows[56]:

$$C = \sum_{j \in J} F_j(P_j) \quad (6.1)$$

$$F_j(P_j) = a_j + b_j P_j + c_j P_j^2 \quad (6.2)$$

where

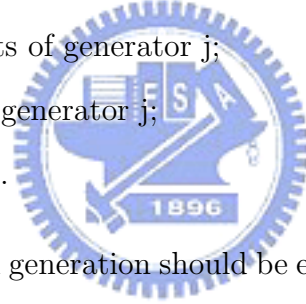
C : total generation cost;

F_j : cost function of generator j ;

a_j, b_j, c_j : cost coefficients of generator j ;

P_j : electrical output of generator j ;

J : set for all generators.



In the real world, the total generation should be equal to the total system demand plus the transmission network loss. However, in this study, the network loss is not considered for simplicity as in many studies. Thus the constraints of the problem include two main parts. The first part is the equality constraint. The total system demand must be equal to the summation of each generator's output, i.e.,

$$\sum_{j \in J} P_j = D, \quad (6.3)$$

where D is the total system demand.

Secondly, the generation output of each unit should be between its minimum and maximum limits. This introduces the inequality constraints as follows:

$$P_{jmin} \leq P_j \leq P_{jmax}, \quad (6.4)$$

where P_{jmin}, P_{jmax} is the minimum, maximum output of generator j .

In reality, the objective function of economic dispatch problem is more complicated due to the valve-point effects and change of fuels. Therefore, the nonsmooth cost functions were considered. The inclusion of valve-point loading effects makes the modeling of the incremental fuel cost function of the generators more practical. This increase the non-linearity as well as number of local optima in the solution space. Also make search algorithm trap in the local optimal easily. The incremental fuel cost function of the generating units with valve-point loadings are represented as follows[57]:

$$F_j(P_j) = a_j + b_j P_j + c_j P_j^2 + |e_j * \sin(f_j * (P_{jmin} - P_j))| \quad (6.5)$$

where e_j and f_j are the coefficients of generator j reflecting valve-point effects.

In this study, we focus on solving the economic dispatch problem with valve point effect, which is model as 6.1. We applied the proposed method as a optimization tool, and the equality and inequality constraints in this problem is handled through repair and penalty mechanisms. A detail description will be given in the following section.

6.2 Our Solution

According to the importance of the ED problem, there were many studies in the past. Especially, evolutionary algorithms like genetic algorithm[57, 58, 59, 60, 61], evolutionary programming[62, 63, 64, 65], particle swarm optimization[66, 67, 68, 69], have been adapted on this problem successfully. Here in this study, we proposed the algorithm with constraint handling techniques including repair and penalty mechanism to the ED problem.

The most important topic for solving the ED problems may be the equality and inequality constraints. These constraints divide the entire solution space into complicated areas. Such a condition leads the search algorithm to be inefficient. To address this problem, we here propose a constraint handling technique based on both of the repair and the penalty concepts. According to the repair concept, the infeasible solution is somehow fixed to be a feasible one. We do this by firstly rearranging a sequence number in a random way. Each number in the sequence represents a generator in the solution. Thus, the result sequence means the order we process the generator. With this order, we

check the equality constraint, i.e., the sum of the generator's output has to be equal to the total demand. If the equality constraint is not satisfied, the output of current generator would be modified according to the following function:

$$G_i = \min(\max((DM - \sum_{j=1, j \neq i}^n G_j), Low_bound(G_i)), Up_bound(G_i)), \quad (6.6)$$

where DM means the total demand output, $Low_bound(G_i)$ and Up_bound mean the low bound and up bound of G_i , i.e., the inequality constraint of G_i .

The above repair process is conducted with a probability, and is repeated until the current solution becomes feasible. On the other hand, to preserve the diversity of the population and the optima that appear close to the boundaries, we do not repair all the infeasible solutions. Instead, we use a simple penalty function to deal with those infeasible solutions. The penalty function was designed as follows:

$$fitness(P_i) = fitness(P_i) + abs(DM - \sum_{j=1}^n P_{ij})^3 \quad (6.7)$$

The proposed algorithm incorporate the constraint handling technique make it able to solve the ED problem efficiently. With the two constraint handling techniques, the diversity of the population is preserved. Furthermore, the recombination process in the original framework make it possible to discover different potential solutions which are divided by the equality and inequality constraints in the ED problem. To verify our approach, numerical experiments were conducted, and the results are given in the next section.

6.3 Experiments

In this section, the test problems will be described in detail. The propose method do work well in the test problem and produces good results. A complete experimental results will be shown and discussed.

6.3.1 Test Problems

In the experiment, we focus on solving the economic dispatch problem with nonsmooth functions considering the valve-point effects. The nonsmooth functions were described

Table 6.1: Units data for test case I(3-Unit System) with valve-point loading. a, b, c, e, and f are cost coefficients in the fuel cost function $F_j(P_j) = a_j + b_j P_j + c_j P_j^2 + |e_j * \sin(f_j * (P_{jmin} - P_j))|$

Generator	P_{min} (MW)	P_{max} (MW)	a	b	c	e	f
1	100	600	0.001562	7.92	561	300	0.0315
2	50	200	0.00482	7.97	78	150	0.063
3	100	400	0.00194	7.85	310	200	0.042

in the previous section and modeled as Equation 6.1. The PSO-RDL is applied to two ED problems, one with 3 generators and another with 40 generators. The input data for 3-generator system are given in [57] and those for 40-generator is given in [63]. The detail parameters include each generator output range and related coefficients in both systems are given in Table 6.1 and Table 6.2. Here, the total demand for the 3-unit and 40-unit systems are set as 850 MW and 10500 MW, respectively. It is proved that for 3-unit system, the glob optimum solution is 8234.07 [70]. As for the 40-unit system, the optimum has not been found yet and the best solution reported until now is 122252.265 [68]. The parameter setting in our algorithm is as same as Table 5.2. The probability threshold used to decide the infeasible solution should be repaired or not is set to 0.4.

6.3.2 Experimental Results

The above experiments were done to evaluate the performance of PSO-RDL on the real-world problem. We have done 100 trials for each problem. The obtained results for the 3-unit system are given in Table 6.3 and the results were compared with those of IEP[71], EP[64] and MPSO[68]. It shows that PSO-RDL has successfully found the global optimum solution presented in [70]. In the case of the 40-unit system, the results are compared with those from other methods in [63] such as classical EP(CEP), fast EP(FEP), modified FEP(MEFP), improved FEP(IFEP), and also the results from MPSO in [68]. The obtained best value from PSO-RDL is 121468.820, which is better than the previous best result 122252.265 in [68]. The best solution obtained in each method is shown in Table 6.4. The generation outputs and the corresponding costs of the best solution are provided in Table 6.5. To compare the result of PSO-RDL with other various methods in a statistical manner, we here shows the each range of cost among 100 trials in Table

6.6. Furthermore, to compare the performance of PSO-RDL with MPSO [68], we also run the t-Test for the 40-unit system experimental results. Since we don't have the actual data set for MPSO, we do the t-Test with two kinds of data. First, we use the MPSO data set contains forty-seven 122252.265 which is the optimum reported in MPSO [68] and fifty-three 122750 which is average of 122500 to 123000. The t-Test results is listed in Table 6.7. Secondly, we use the MPSO data set contains forty-seven 122252.265 which is the optimum reported in MPSO [68] and fifty-three 122750 which is best value in the range from 122500 to 123000. The t-Test results is listed in Table 6.8.

From the experimental results, it is obvious that our algorithm performs well for these two ED problems. Especially for the 40-unit system, we improve the known best solution to 121468.82. From the Table 6.6, 6.7, 6.8, it can be observed that our algorithm is statistically outperformed MPSO[68]. The ED problem is a highly constraint optimization problem, and we use two constraint handling techniques including the repair techniques and the penalty function. These two mechanisms are easy to implement and incorporate well with the proposed algorithm. From this application, we can find that for the constrained optimization problems, the proposed algorithm can also still performs well as in the unconstrained optimization problems.

Table 6.2: Units data for test case II(40-Unit System) with valve-point loading. a, b, c, e, and f are cost coefficients in the fuel cost function $F_j(P_j) = a_j + b_j P_j + c_j P_j^2 + |e_j * \sin(f_j * (P_{jmin} - P_j))|$

Generator	P_{min} (MW)	P_{max} (MW)	a	b	c	e	f
1	36	114	0.0069	6.73	94.705	100	0.084
2	36	114	0.0069	6.73	94.705	100	0.084
3	60	120	0.2028	7.07	309.54	100	0.084
4	80	190	0.00942	8.18	369.03	150	0.063
5	47	97	0.0114	5.35	148.89	120	0.077
6	68	140	0.01142	8.05	222.33	100	0.084
7	110	300	0.00357	8.03	287.71	200	0.042
8	135	300	0.00492	6.99	391.98	200	0.042
9	135	300	0.00573	6.6	455.76	200	0.042
10	130	300	0.00605	12.9	722.82	200	0.042
11	94	375	0.00515	12.9	635.2	200	0.042
12	94	375	0.00569	12.8	654.69	200	0.042
13	125	500	0.00421	12.5	913.4	300	0.035
14	125	500	0.00752	8.84	1760.4	300	0.035
15	125	500	0.00708	9.15	1728.3	300	0.035
16	125	500	0.00708	9.15	1728.3	300	0.035
17	220	500	0.00313	7.97	647.85	300	0.035
18	220	500	0.00313	7.95	649.69	300	0.035
19	242	550	0.00313	7.97	647.83	300	0.035
20	242	550	0.00313	7.97	647.81	300	0.035
21	254	550	0.00298	6.63	785.96	300	0.035
22	254	550	0.00298	6.63	785.96	300	0.035
23	254	550	0.00284	6.66	794.53	300	0.035
24	254	550	0.00284	6.66	794.53	300	0.035
25	254	550	0.00277	7.1	801.32	300	0.035
26	254	550	0.00277	7.1	801.32	300	0.035
27	10	150	0.52124	3.33	1055.1	120	0.077
28	10	150	0.52124	3.33	1055.1	120	0.077
29	10	150	0.52124	3.33	1055.1	120	0.077
30	47	97	0.0114	5.35	148.89	120	0.077
31	60	190	0.0016	6.43	222.92	150	0.063
32	60	190	0.0016	6.43	222.92	150	0.063
33	60	190	0.0016	6.43	222.92	150	0.063
34	90	200	0.0001	8.95	107.87	20	0.042
35	90	200	0.0001	8.62	116.58	200	0.042
36	90	200	0.0001	8.62	116.58	200	0.042
37	25	110	0.0161	5.88	307.45	80	0.098
38	25	110	0.0161	5.88	307.45	80	0.098
39	25	110	0.0161	5.88	307.45	80	0.098
40	242	550	0.00313	7.97	647.83	300	0.035

Table 6.3: Comparison of simulation results of each method considering valve-point effect (3-unit system)

Unit	GA	IEP (pop=20)	EP	MPSO (par=20)	PSO-RDL (par=20)
1	300	300.23	300.26	300.27	300.267
2	400	400	400	400	400
3	150	149.77	149.74	149.73	149.733
TP	850	850	850	850	850
TC	8237.6	8234.09	8234.07	8234.07	8234.07



Table 6.4: Comparison of simulation results of each method considering valve-point effect (40-unit system)

	CEP	FEP	MFEP	IFEP	MPSO	PSO-RDL
Minimum cost	123488.3	122679.7	122647.6	122624.35	122252.3	121468.82

Table 6.5: Generation output of each generator and the corresponding cost in 40-unit system

Unit	Pmin(MW)	Pmax(MW)	Generation	Cost
1	36	114	112.2886	949.880767
2	36	114	111.0704	929.604348
3	60	120	97.49443	1192.38418
4	80	190	179.7531	2143.97098
5	47	97	88.89745	724.712068
6	68	140	140	1596.46432
7	110	300	300	3216.42404
8	135	300	284.7229	2782.07788
9	135	300	284.777	2801.46883
10	130	300	130	2502.065
11	94	375	94.00612	1893.44177
12	94	375	94.03925	1909.04089
13	125	500	214.77	3792.32437
14	125	500	394.2823	6414.93466
15	125	500	304.5313	5171.47843
16	125	500	394.2847	6436.72027
17	220	500	489.2827	5296.78245
18	220	500	489.3102	5289.42926
19	242	550	511.2908	5541.17665
20	242	550	511.2941	5541.22862
21	254	550	523.2818	5071.33798
22	254	550	523.398	5073.69255
23	254	550	523.3437	5058.51899
24	254	550	523.3715	5059.07705
25	254	550	523.2815	5275.13221
26	254	550	523.28	5275.10232
27	10	150	10.00005	1140.52506
28	10	150	10.00442	1140.62574
29	10	150	10.01797	1140.93732
30	47	97	92.60281	785.447407
31	60	190	190	1643.99125
32	60	190	190	1643.99125
33	60	190	190	1643.99125
34	90	200	200	2101.01703
35	90	200	200	2043.72703
36	90	200	200	2043.72703
37	25	110	110	1220.16612
38	25	110	110	1220.16612
39	25	110	110	1220.16612
40	242	550	511.3228	5541.87129
Total Generation & Total Cost			10500	121468.82

Table 6.6: Comparison of method on relative frequency of convergence in the ranges of cost

Range of Cost [k\$]												
Evaluation Method	127.0	126.5	126.0	125.5	125.0	124.5	124.0	123.5	123.0	122.5	122.0	121.5
	-	-	-	-	-	-	-	-	-	-	-	-
	126.5	126.0	125.5	125.0	124.5	124.0	123.5	123.0	122.5	122.0	121.5	121.0
CEP	10	4	-	16	22	42	4	2	-	-	-	-
FEP	6	-	4	2	10	20	26	24	6	-	-	-
MFEP	-	-	-	-	-	14	26	50	10	-	-	-
IFEP	-	-	2	-	4	4	18	50	22	-	-	-
MPSO	-	-	-	-	-	-	-	-	53	47	-	-
PSO-RDL	-	-	-	-	-	-	-	6	8	36	49	1

Table 6.7: t-Test for the results of PSO-RDL and MPSO under condition 1, where the PSO-RDL data set contains the actual results, and the MPSO data set contains forty-seven 122252.265 and fifty-three 122750.

Method	PSO-RDL	MPSO
mean	122083.5084	122516.0646
t value	7.12311	
p value	1.91171E-11	

Table 6.8: t-Test for the results of PSO-RDL and MPSO under condition 2, where the PSO-RDL data set contains the actual results, and the MPSO data set contains forty-seven 122252.265 and fifty-three 122500.

Method	PSO-RDL	MPSO
mean	122083.5084	122516.06455
t value	9.13251	
p value	7.91529E-17	

Chapter 7

Conclusions

7.1 Summary

In this project, we studied the particle swarm optimization and genetic linkage problems in genetic algorithms. After survey on the hybridization of particle swarm optimizer and genetic algorithms, we decided to introduce the genetic linkage concept, which is an important topic in genetic algorithms, to particle swarm optimizer. To address the genetic linkage problem in real-parameter optimization problems, we develop the dynamic linkage discovery technique. Further, in order to make good use of building blocks information, we also design a recombination operator. By combining these mechanisms, we proposed a new efficient search algorithm and have the experiments on real-parameter test functions. Finally, we applied PSO-RDL on the economic dispatch problem, which is an essential problem in the power control system.

Chapter 2 briefly introduced the particle swarm optimization algorithms, including the historical background, working principles of PSO. The initial and modified global version PSO were described, and the local variant can be made through small changes. In order to understand how parameters impact on the PSO, we also make a short discussion of the parameters control in PSO. Finally, the recent advances of PSO that related to our research work were done that help us understand the problems and issues in particle swarm optimization.

Chapter 3 explains the genetic linkage in genetic algorithm and shows the impacts on genetic algorithms through identifying genetic linkage. Various kinds of genetic linkage learning technique are also discussed in this chapter. Under the assumption of building

blocks hypothesis, it is obviously that addressing genetic linkage problem is an important issued when using genetic algorithms.

Chapter 4 describes about the dynamic linkage discovery technique and recombination operator are also given. Finally, the complete algorithm flow is introduced. In this algorithm, we introduce the genetic linkage concept to particle swarm optimization. Based on the natural selection concept, a dynamic linkage discovery technique is designed. Furthermore, in order to make good use of building blocks, a recombination operator is incorporated.

In Chapter 5, the performance of PSO-RDL is evaluated by simulated the search on 25 numerical functions that proposed in CEC2005 special session on real-parameter optimization. The description of the test functions, parameter setting and experimental results were given in this chapter. We also discussed the strength and weakness of the PSO-RDL by analyzed the search result and linkage dynamics. From the results, it is considered that the PSO-RDL algorithm can work well and produce a better performance than the traditional particle swarm optimizer.

Chapter 6 applied PSO-RDL on the economic dispatch problem in the power system. The economic dispatch problem is essential to power controls and include many constraints. We evaluate the performance by solving both the 3-unit and 40-unit systems. Comparing to other advance evolutionary algorithms, PSO-RDL do performs well on economic dispatch problems.

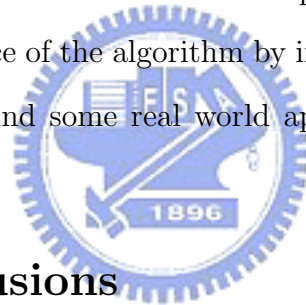
7.2 Future Work

In this paper, we proposed a new framework by introducing the recombination mechanism with the dynamic linkage discovery technique to PSO. From the experiments, the proposed algorithm can provide a good performance on a carefully designed benchmark function set. The following is a list of research directions for future consideration suggested by the author:

- From the experimental results, we can observe that the dynamic linkage discovery technique and recombination operator do improve the performance of the particle

swarm optimizer. Hence, we can try to apply the dynamic linkage discovery technique with recombination operator to other evolutionary optimization algorithms.

- Since we have successfully improve the performance of particle swarm optimizer by introducing the linkage concept and genetic operator, it shows the possibility that the linkage problem exists in the real-parameter optimization problems. Moreover, from the observation of the linkage dynamics of the experiments, the linkage configuration may dynamically change through the search process. Based on these observations, we can develop other linkage discover techniques for real-parameter optimization problems and improves the performance of the algorithm through addressing the genetic linkage problems.
- According to the analysis and observation of the experimental results, we have understood the strength and weakness of the proposed method. Thus, we should enhance the performance of the algorithm by improving the shortcomings. Furthermore, we should also find some real world applications with different features as advanced experiments.



7.3 Main Conclusions

In this paper, we first surveyed on the recent studies. We recognize the importance of the linkage concept of GA and that the correct combination of GA and PSO can lead to the further algorithmic advance. We then introduced the dynamic linkage discovery technique into PSO by incorporating the recombination operator to work on the identified building blocks. We adopted the benchmark functions defined in CEC2005 to evaluate the performance of the proposed algorithm. The experimental results indicated that the proposed algorithm can provide a good performance on the benchmark functions of different characteristics.

Furthermore, the work on PSO-RDL gives us two observations. First, in the literature, it is rarely discussed about the building blocks in real-parameter optimization problems. This work may shed light on the existence of building blocks in real-parameter optimization problems. Secondly, if building blocks do exist, then why these building blocks

cannot be detected by the linkage detection techniques previously proposed in the literature? According to the information obtained in this study, perhaps in a real-parameter optimization problem, the configuration of building blocks dynamically changes along with the search stage. Thus, those traditional, static linkage detection techniques fail to accomplish the task.

In this study, we introduce recombination with dynamic linkage discovery to PSO and consider the integration as a promising research direction. By combining the strength of different optimization models, we create the PSO-RDL algorithm with intriguing features and properties. We will continue to work on understanding and analyzing the real number optimization problem in order to design better evolutionary algorithms in the future.



Appendix A

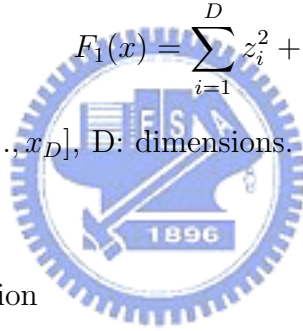
CEC'05 25 Real-Parameter Functions

Unimodal Problems:

1. Shifted Sphere Function

$$F_1(x) = \sum_{i=1}^D z_i^2 + f_bias_1 \quad (A.1)$$

$z = x - o$, $x = [x_1, x_2, \dots, x_D]$, D: dimensions. $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum.



2. Shifted Schwefels Function

$$F_2(x) = \sum_{i=1}^D \left(\sum_{j=1}^i z_j \right)^2 + f_bias_2 \quad (A.2)$$

$z = x - o$, $x = [x_1, x_2, \dots, x_D]$, D: dimensions. $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum.

3. Shifted Rotated High Conditioned Elliptic Function

$$F_3(x) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_bias_3 \quad (A.3)$$

$z = (x - o) * M$, $x = [x_1, x_2, \dots, x_D]$, D: dimensions. $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum. M: orthogonal matrix

4. Shifted Schwefels Problem 1.2 with Noise in Fitness

$$F_4(x) = \sum_{i=1}^D \left(\sum_{j=1}^i z_j \right)^2 * (1 + 0.4 |N(0, 1)|) + f_bias_4 \quad (A.4)$$

$z = x - o$, $x = [x_1, x_2, \dots, x_D]$, D: dimensions. $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum.

5. Schwefels Problem 2.6 with Global Optimum on Bounds

$$f(x) = \max \{ |x_1 + 2x_2 - 7|, |2x_1 + x_2 - 5| \}, i = 1, \dots, n, x^* = [1, 3], f(x^*) = 0$$

Extend to D Dimensions:

$$F_5(x) = \max\{|A_i x - B_i|\} + f_bias_5 \quad (A.5)$$

$$, i = 1, \dots, D, x = [x_1, x_2, \dots, x_D]$$

D:dimensions

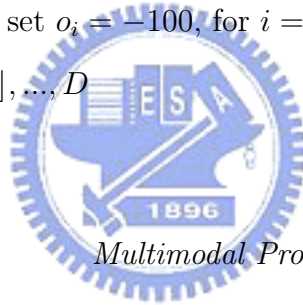
A is a $D * D$ matrix, a_{ij} are integer random number in the range $[-500, 500]$,

$\det(A) \neq 0$, A_i is the i^{th} row of A.

$B_i = A_i * o$, o is a $D * 1$ vector, o_i are random number in the range $[-100, 100]$

After load the data file, set $o_i = -100$, for $i = 1, 2, \dots, \lceil D/4 \rceil$,

$o_i = 100$, for $i = \lfloor 3D/4 \rfloor, \dots, D$



Multimodal Problems:

6. Shifted Rosenbrocks Function

$$F_6(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_bias_6 \quad (A.6)$$

$$z = x - o + 1, x = [x_1, x_2, \dots, x_D],$$

D: dimensions. $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum.

7. Shifted Rotated Griewanks Function without Bounds

$$F_7(x) = \sum_{i=1}^D \frac{z_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_bias_7 \quad (A.7)$$

$$z = (x - o) * M, x = [x_1, x_2, \dots, x_D],$$

D: dimensions. $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum.

M' : linear transformation matrix, condition number=3

$$M = M'(1 + 0.3|N(0, 1)|)$$

8. Shifted Rotated Ackleys Function with Global Optimum on Bounds

$$F_8(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D z_i^2}) - \exp\left(\frac{1}{D} \cos(2\pi z_i)\right) + 20 + e + f_bias_8 \quad (\text{A.8})$$

$$z = (x - o) * M, \quad x = [x_1, x_2, \dots, x_D],$$

D: dimensions. $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum.

After load the data file, set $o_{2j-1} = -32o_{2j}$ are randomly distributed in the search range, for $j = 1, 2, \dots, \lfloor D/2 \rfloor$

M : linear transformation matrix, condition number=100

9. Shifted Rastrigins Function

$$F_9(x) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10) + f_bias_9 \quad (\text{A.9})$$

$z = x - o, \quad x = [x_1, x_2, \dots, x_D]$, D: dimensions. $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum.

10. Shifted Rotated Rastrigins Function

$$F_{10}(x) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10) + f_bias_{10} \quad (\text{A.10})$$

$$z = (x - o) * M, \quad x = [x_1, x_2, \dots, x_D],$$

D: dimensions. $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum.

M : linear transformation matrix, condition number=2

11. Shifted Rotated Weierstrass Function

$$F_{11}(x) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (z_i + 0.5))] \right) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k * 0.5)] + f_bias_{11} \quad (\text{A.11})$$

$$a = 0.5, \quad b = 3, \quad k_{\max} = 20, \quad z = (x - o) * M, \quad x = [x_1, x_2, \dots, x_D],$$

D: dimensions. $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum.

M : linear transformation matrix, condition number=5

12. Schwefels Problem 2.13

$$F_{12}(x) = \sum_{i=1}^D (A_i - B_i(x))^2 + f_bias_{12} \quad (A.12)$$

$$A_i = \sum_{j=1}^D (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j), B_i(x) = \sum_{j=1}^D (a_{ij} \sin x_j + b_{ij} \cos x_j)$$

D:dimensions

A, B are two $D * D$ matrix, $a_{ij}, b_{i,j}$ are integer random numbers in the range $[-100, 100]$, $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_D]$, α_j are random numbers in the range $[-\pi, \pi]$

Expanded Problems

Using a 2-D functions $F(x, y)$ as a starting functions, corresponding expanded function is :

$$EF(x_1, x_2, \dots, x_D) = F(x_1, x_2) + F(x_2, x_3) + \dots + F(x_{D-1}, x_D) + F(x_D, x_1)$$

13. Shifted Expanded Griewank's plus Rosenbrock's Function

$$F8:\text{Griewank's Function: } F8(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$F2:\text{Rosenbrock's Function: } F2(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$

$$F8F2(x_1, x_2, \dots, x_D) = F8(F2(x_1, x_2)) + F8(F2(x_2, x_3)) + \dots$$

$$+ F8(F2(x_{D-1}, x_D)) + F8(F2(x_D, x_1))$$

Shift to

$$F_{13}(x) = F8(F2(z_1, z_2)) + F8(F2(z_2, z_3)) + \dots + F8(F2(z_{D-1}, z_D))$$

$$+ F8(F2(z_D, z_1)) + f_bias_{13} \quad (A.13)$$

$$z = x - o + 1, x = [x_1, x_2, \dots, x_D]$$

D: dimensions $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum

14. Shifted Rotated Expanded Scaffer's F6 Function

$$F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2+y^2})-0.5)}{(1+0.001(x^2+y^2))^2}$$

Expanded to

$$F_{14}(X) = EF(z_1, z_2, \dots, z_D) = F(z_1, z_2) + F(z_2, z_3) + \dots + F(z_{D-1}, z_D) + F(z_D, z_1) + f_bias_{14} \quad (\text{A.14})$$

$$z = (x - o) * M, \quad x = [x_1, x_2, \dots, x_D],$$

D: dimensions. $o = [o_1, o_2, \dots, o_D]$: the shifted global optimum.

M: linear transformation matrix, condition number=3

Composition Problems

$F(x)$: new composition function

$f_i(x)$: i^{th} basic function used to construct the composition function

n : number of basic functions

D : dimensions

M_i : linear transformation matrix for each $f_i(x)$

o_i : new shifted optimum position for each $f_i(x)$

$$F(x) = \sum_{i=1}^n \{w_i * [f'_i((x - o_i)/\lambda_i * M_i) + bias_i]\} + f_bias$$

w_i : weight value for each $f_i(x)$, calculated as below:

$$w_i = \exp\left(-\frac{\sum_{k=1}^D (x_k - o_{ik})^2}{2D\sigma_i^2}\right)$$

$$w_i = \begin{cases} w_i & w_i = \max(w_i) \\ w_i * (1 - \max(w_i))^{10} & w_i \neq \max(w_i) \end{cases}$$

then normalize the weight $w_i = w_i / \sum_{i=1}^n w_i$

σ_i : used to control each $f_i(x)$'s coverage range, a small σ_i give a narrow range for that $f_i(x)$

λ_i : used to stretch compress the function, $\lambda_i > 1$ means stretch, $\lambda_i < 1$

means compress

o_i define the global and local optima's position, $bias_i$ define which optimum is global optimum. Using o_i , $bias_i$, a global optimum can be placed anywhere.

If $f_i(x)$ are different functions, different functions have different properties and height, in order to get a better mixture, estimate a biggest function value f_{maxi} for 10 functions $f_i(x)$, then normalize each basic functions to similar heights as below:

$$f'_i(x) = C * f_i(x) / |f_{maxi}|, C \text{ is a predefined constant.}$$

$$|f_{maxi}| \text{ is estimated using } |f_{maxi}| = f_i((x'/\lambda_i) * M_i), x' = [5, 5, \dots, 5]$$

In the following composition functions, Number of basic functions $n = 10$.

D : dimensions, $o:n * D$ matrix, defines $f_i(x)$'s global optimal positions.

$bias = [0, 100, 200, 300, 400, 500, 600, 700, 800, 900]$. Hence, the first function $f_1(x)$ always the function with the global optimum.

$$C = 2000$$

15. Hybrid Composition Function

$f_{1-2}(x)$: Rastrigin's Function

$$f_i(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$$

$f_{3-4}(x)$: Weierstrass Function

$$f_i(x) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k * 0.5)]$$

$$a = 0.5, b = 3, k_{max} = 20$$

$f_{5-6}(x)$: Griewank's Function

$$f_i(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$f_{7-8}(x)$: Ackley's Function

$$f_i(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{D} \cos(2\pi x_i)) + 20 + e$$

$f_{9-10}(x)$: Sphere Function

$$f_i(x) = \sum_{i=1}^D x_i^2$$

$\sigma_i = 1$ for $i = 1, 2, \dots, D$

$\lambda = [1, 1, 10, 10, 5/60, 5/60, 5/32, 5/32, 5/100, 5/100]$

M_i are all identity matrices

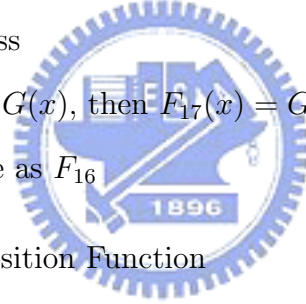
16. Rotated Version of Hybrid Composition Function F_{15}

Except M_i are different linear transformation matrixes with condition number of 2, all other setting are the same as F_{15}

17. F_{16} with Noise in Fitness

Let $(F_{16} - f_bias_{16})$ be $G(x)$, then $F_{17}(x) = G(x) * (1 + 0.2|N(0, 1)|) + f_bias_{17}$

All setting are the same as F_{16}



18. Rotated Hybrid Composition Function

$f_{1-2}(x)$: Ackley's Function

$$f_i(x) = -20 \exp(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}) - \exp(\frac{1}{D} \cos(2\pi x_i)) + 20 + e$$

$f_{3-4}(x)$: Rastrigin's Function

$$f_i(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$$

$f_{5-6}(x)$: Sphere Function

$$f_i(x) = \sum_{i=1}^D x_i^2$$

$f_{7-8}(x)$: Weierstrass Function

$$f_i(x) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k * 0.5)] \quad a = 0.5, b = 3, k_{\max} = 20$$

$f_{9-10}(x)$: Griewank's Function

$$f_i(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$\sigma = [1, 2, 1.5, 1.5, 1, 1, 1.5, 1.5, 2, 2,];$$

$$\lambda = [2 * 5/32; 5/32; 2 * 1; 1; 2 * 5/100; 5/100; 2 * 10; 10; 2 * 5/60; 5/60]$$

M_i are all rotation matrices. Condition numbers are [2 3 2 3 2 3 20 30 200 300]

$$o_{10} = [0,0,\dots,0]$$

19. Rotated Hybrid Composition Function with narrow basin global optimum

All settings are the same as F_{18} except

$$\sigma = [0.1, 2, 1.5, 1.5, 1, 1, 1.5, 1.5, 2, 2]$$

$$\lambda = [0.1 * 5/32; 5/32; 2 * 1; 1; 2 * 5/100; 5/100; 2 * 10; 10; 2 * 5/60; 5/60]$$

20. Rotated Hybrid Composition Function with Global Optimum on the Bounds

All settings are the same as F_{18} except after load the data file, set $o_{1(2j)} = 5$, for

$$j = 1, 2, \dots, \lfloor D/2 \rfloor$$

21. Rotated Hybrid Composition function

$f_{1-2}(x)$: *Rotated Expanded Scaffer's F6 Function*

$$F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2+y^2})-0.5)}{(1+0.001(x^2+y^2))^2}$$

$$f_i(x) = F(x_1, x_2) + F(x_2, x_3) + \dots + F(x_{D-1}, x_D) + F(x_D, x_1)$$

$f_{3-4}(x)$: Rastrigin's Function

$$f_i(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$$

$f_{5-6}(x)$: F8F2 Function

$$F8(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$F2(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$

$$f_i(x) = F8(F2(x_1, x_2)) + F8(F2(x_2, x_3)) + \dots + F8(F2(x_{D-1}, x_D)) + F8(F2(x_D, x_1))$$

$f_{7-8}(x)$: Weierstrass Function

$$f_i(x) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k * 0.5)]$$

$a = 0.5, b = 3, k_{\max} = 20$

$f_{9-10}(x)$: Griewank's Function

$$f_i(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$\sigma = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2];$$

$$\lambda = [5 * 5/100; 5/100; 5 * 1; 1; 5 * 1; 1; 5 * 10; 10; 5 * 5/200; 5/200]$$

M_i are all orthogonal matrix

22. Rotated Hybrid Composition Function with High Condition Number Matrix

All settings are the same as F_{21} except M_i 's condition numbers are [10 20 50 100 200 1000 2000 3000 4000 5000]

23. Non-Continuous Rotated Hybrid Composition Function

All settings are the same as F_{21}

$$\text{Except } x_j = \begin{cases} x_j & |x_j - o_{1j}| < 1/2 \\ \text{round}(2x_j)/2 & |x_j - o_{1j}| \geq 1/2 \end{cases} \text{ for } j = 1, 2, \dots, D$$

$$\text{round}(x) = \begin{cases} a - 1 & \text{if } x \leq 0 \& b \geq 0.5 \\ a & \text{if } b < 0.5 \\ a + 1 & \text{if } x > 0 \& b \geq 0.5 \end{cases}$$

where a is x 's integral part and b is x 's decimal part

All "round" operators in this document use the same schedule.

24. Rotated hybrid Composition Function

$f_1(x)$: Weierstrass Function

$$f_i(x) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k * 0.5)]$$

$a = 0.5, b = 3, k_{\max} = 20$

$f_2(x)$: Rotated Expanded Scaffer's F6 Function

$$F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2+y^2})-0.5)}{(1+0.001(x^2+y^2))^2}$$

$$f_i(x) = F(x_1, x_2) + F(x_2, x_3) + \dots + F(x_{D-1}, x_D) + F(x_D, x_1)$$

$f_3(x)$: F8F2 Function

$$F8(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$F2(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$

$$f_i(x) = F8(F2(x_1, x_2)) + F8(F2(x_2, x_3)) + \dots + F8(F2(x_{D-1}, x_D)) + F8(F2(x_D, x_1))$$

$f_4(x)$: Ackley's Function

$$f_i(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}\right) - \exp\left(\frac{1}{D} \cos(2\pi x_i)\right) + 20 + e$$

$f_5(x)$: Rastrigin's Function

$$f_i(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$$

$f_6(x)$: Griewank's Function

$$f_i(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$f_7(x)$: Non-Continuous Expanded Scaffer's F6 Function

$$F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2}$$

$$f(X) = F(y_1, y_2) + F(y_2, y_3) + \dots + F(y_{D-1}, y_D) + F(y_D, y_1)$$

$$y_j = \begin{cases} y_j & |y_j - o_{1j}| < 1/2 \\ \text{round}(2y_j)/2 & |y_j - o_{1j}| \geq 1/2 \end{cases} \quad \text{for } j = 1, 2, \dots, D$$

$f_8(x)$: Non-Continuous Rastrigin's Function

$$f(x) = \sum_{i=1}^D (y_i^2 - 10 \cos(2\pi y_i) + 10)$$

$$y_j = \begin{cases} y_j & |y_j - o_{1j}| < 1/2 \\ \text{round}(2y_j)/2 & |y_j - o_{1j}| \geq 1/2 \end{cases} \quad \text{for } j = 1, 2, \dots, D$$

$f_9(x)$: High Conditioned Elliptic Function

$$f(x) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} x_i^2$$

$f_{10}(x)$: Sphere Function with Noise in Fitness

$$f_i(x) = \left(\sum_{i=1}^D x_i^2\right)(1 + 0.1|N(0, 1)|)$$

$\sigma_i = 2$, for $i = 1, 2, \dots, D$

$\lambda = [10; 5/20; 1; 5/32; 1; 5/100; 5/50; 1; 5/100; 5/100]$

M_i are all rotation matrices, condition numbers are [100 50 30 10 5 5 4 3 2 2];

F_{25} : Rotated Hybrid Composition Function without bounds

All settings are same as F_{24} except no exact search range set for this test function.



Appendix B

Global Optimum for CEC'05 25 Real-Parameter Functions



Table B.1: The individual of global optimum for each function in the benchmark

Function	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	$f(x)$
1	-39.312	58.9	-46.322	-74.651	-16.8	-80.544	-10.594	24.969	89.838	9.1119	-450
2	35.627	-82.912	-10.642	-83.582	83.155	47.048	-89.436	-27.422	76.145	-39.06	-450
3	-32.201	64.978	-38.3	-23.258	-54.009	86.629	-6.3009	-49.364	5.3499	52.242	-450
4	35.627	-82.912	-10.642	-83.582	83.155	47.048	-89.436	-27.422	76.145	-39.06	-450
5	-100	-100	-100	8.3897	7.7182	-8.3147	100	100	100	100	-310
6	81.023	-48.395	19.232	-2.5231	70.434	47.177	-7.8358	-86.669	57.853	-9.9533	390
7	-276.27	-11.911	-578.79	-287.65	-84.386	-228.68	-458.15	-202.21	-105.86	-96.49	-180
8	-32	14.977	-32	9.5566	-32	-17.19	-32	0.8511	-32	10.793	-140
9	1.9005	-1.5644	-0.9788	-2.2536	2.499	-3.2853	0.9759	-3.6661	0.0985	-3.2465	-330
10	1.9005	-1.5644	-0.9788	-2.2536	2.499	-3.2853	0.9759	-3.6661	0.0985	-3.2465	-330
11	-0.1367	0.1186	-0.0968	0.0237	-0.2933	-0.0478	0.3518	0.3579	-0.0586	-0.0375	90
12	-1.99236	-1.10056	-0.86187	-1.18194	-0.54732	0.504645	0.974223	-1.24602	1.831887	2.351454	-460
13	0.2471	-0.8497	0.5629	0.2673	0.6874	0.1395	-0.2764	0.4172	-0.5007	-0.5204	-130
14	-73.603	-23.55	-21.774	18.134	32.724	34.413	80.946	48.147	19.151	10.103	-300
15	3.3253	-1.2835	1.8984	-0.4095	0.0881	2.758	0.9776	-1.809	-2.4957	2.7367	120
16	3.3253	-1.2835	1.8984	-0.4095	0.0881	2.758	0.9776	-1.809	-2.4957	2.7367	120
17	3.3253	-1.2835	1.8984	-0.4095	0.0881	2.758	0.9776	-1.809	-2.4957	2.7367	120
18	1.5953	2.644	1.8047	0.9389	-3.0486	-1.1571	3.5582	2.4246	-0.3767	4.4637	10
19	1.5953	2.644	1.8047	0.9389	-3.0486	-1.1571	3.5582	2.4246	-0.3767	4.4637	10
20	1.5953	5	1.8047	5	-3.0486	5	3.5582	5	-0.3767	5	10
21	1.2141	-0.01	1.8864	-4.1124	2.0627	1.1535	4.0653	-1.0213	1.1986	-4.1795	360
22	1.2141	-0.01	1.8864	-4.1124	2.0627	1.1535	4.0653	-1.0213	1.1986	-4.1795	360
23	1.2141	-0.01	1.8864	-4.1124	2.0627	1.1535	4.0653	-1.0213	1.1986	-4.1795	360
24	-2.933	-3.034	-4.3583	-4.2851	-3.505	-1.4618	-3.7277	-2.1445	-3.1317	-2.5637	260
25	-2.933	-3.034	-4.3583	-4.2851	-3.505	-1.4618	-3.7277	-2.1445	-3.1317	-2.5637	260

Appendix C

Solutions found by PSO-RDL for CEC'05 25 Real-Parameter Functions



Table C.1: The solution found by PSO-RDL for each function in the benchmark

Function	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	$f(x)$
1	-39.312	58.9	-46.322	-74.651	-16.8	-80.544	-10.594	24.969	89.838	9.1119	-450
2	35.627	-82.912	-10.642	-83.582	83.155	47.048	-89.436	-27.422	76.145	-39.06	-450
3	-32.2019	64.97619	-38.3039	-23.2536	-54.0095	86.62329	-6.29976	-49.3608	5.349092	52.24255	-449.999619
4	35.6267	-82.9123	-10.6423	-83.5815	83.1552	47.048	-89.4359	-27.4219	76.1448	-39.0595	-450
5	-100	-100	-100	8.3897	7.7182	-8.3147	100	100	100	100	-310
6	81.0232	-48.395	19.2316	-2.5231	70.4338	47.1774	-7.8358	-86.6693	57.85319	-9.95332	390
7	-276.268	-11.911	-578.788	-287.649	-84.3858	-228.675	-458.152	-202.215	-105.864	-96.4898	-180
8	-19.9608	14.0162	2.69311	0.762849	19.63525	2.959513	16.62638	9.89564	-1.99917	-22.1767	-139.999619
9	0.905541	-1.5644	-0.9788	-2.2536	2.499	-2.29034	0.9759	-3.6661	0.0985	-3.2465	-330
10	2.684208	-0.34563	-1.18601	-1.62344	2.684879	-2.58328	-0.29892	-2.39323	0.146698	-2.28207	-330
11	0.100636	-0.38906	-0.37413	0.074387	0.293118	0.014596	0.238655	0.192788	-0.24594	0.351385	92.828684
12	-1.48581	-0.55785	-1.48574	-1.35094	-0.36165	0.147824	0.482965	-1.52963	1.457278	-4.00696	-460
13	-0.2376	-1.58396	-0.60608	-1.03873	-0.54839	-1.1371	-0.86921	-0.17334	-0.90296	-0.81092	-129.661766
14	-13.543	-2.63707	-22.2218	-31.764	3.149738	-17.0834	82.81796	-4.17712	6.051408	-46.8081	-296.864114
15	4.329885	-1.28442	1.898128	-0.40647	-1.90249	2.762148	1.97788	-2.80085	-2.50322	2.734012	220.6482
16	5	0.881179	2.130428	0.425533	-0.88378	2.441627	2.055183	-3.69536	-2.5206	2.605947	247.8216
17	4.918916	0.125656	1.973826	-0.36212	-0.19562	1.728074	2.748412	-3.3316	-2.18652	3.307306	238.6596
18	3.334302	-0.873	1.3635	4.259698	2.576901	0.8656	-0.3686	-3.9734	4.291299	2.750802	810
19	3.063854	-1.64763	-5	-4.9465	-3.71656	-4.93662	-4.98062	-3.76369	4.768376	0.558586	705.264
20	3.334296	-0.873	1.363498	4.259704	2.576897	0.865603	-0.3686	-3.9734	4.291302	2.750797	810
21	-2.6639	-2.1802	-3.1792	2.0201	2.4322	1.5009	-0.7393	-0.2911	-1.4312	2.7447	660
22	-2.56932	-4.14768	-4.93465	1.165214	1.717028	2.404797	-2.21793	0.663713	-2.00378	3.883255	1110.2249
23	0.695732	-4.02173	-0.31046	4.403578	0.058289	3.167627	3.032266	-2.43707	3.600158	0.84034	785.1725
24	0.7973	-2.6471	4.2808	-3.1835	1.7367	-2.3262	-0.8345	-2.9156	3.6308	3.8112	460
25	2	2.02289	2	2.067772	2	2	2.004609	2.042546	5	4.999993	1997.223

Bibliography

- [1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 1942-1948, 1995.
- [2] J. Kennedy and R. C. Eberhart, "A new optimizer using particle swarm theory," in *Proceeding of the Sixth Int. Symposium on Micromachine and Human Science*, Nagoya, Japan, 1995, pp. 39–43.
- [3] J. H. Holland, *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.
- [4] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.
- [5] ———, *The design of innovation: Lessons from and for competent genetic algorithms*. Kluwer Academic Publishers, 2002.
- [6] D. E. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: Motivation, analysis, and first result," *Complex Systems*, vol. 3, pp. 493–530, 1989.
- [7] D. E. Goldberg, K. Deb, and B. Korb, "Messy genetic algorithms revisited: Studies in mixed size and scale," *Complex Systems*, vol. 4, pp. 415–444, 1990.
- [8] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," *Evolutionary Programming of Lecture Notes in Computer Science*, vol. 1447, pp. 611–616, 1998.
- [9] P. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," *Evolutionary Programming*, vol. 1447 of Lecture Notes in Computer Science, pp. 601–610, 1998.

- [10] M. Lvbjerg, T. K. Rasmussen, and T. Krink, “Hybrid particle swarm optimiser with breeding and subpopulations,” in *Proceeding of Genetic and Evolutionary Computation Conference*, 2001.
- [11] M. Settles and T. Soule, “Breeding swarms: A GA/PSO hybrid,” in *Proceeding of Genetic and Evolutionary Computation Conference*, Washington, DC, USA, 2005, pp. 161–168.
- [12] D. Devicharan and C. K. Mohan, “Particle swarm optimization with adaptive linkage learning,” in *Congress on Evolutionary Computation*, Portland, Oregon, 2004, pp. 530–535.
- [13] F. Heppner and U. Grenander, “A stochastic nonlinear model for coordinated bird flocks.” *American Association for the Advancement of Science*, 1990.
- [14] C. W. Reynolds, “Flocks, herds and schools: a distributed behavioral model,” *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.
- [15] E. O. Wilson, *Sociobiology: The new synthesis*. Belknap Press, Cambridge, MA, 1975.
- [16] J. Kennedy, “The particle swarm: Social adaptation of knowledge,” in *IEEE International Conference on Evolutionary Computation*, 1997, pp. 303–308.
- [17] Y. Shi and R. Eberhart, “A modified particle swarm optimization,” in *IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, 1998, pp. 69–73.
- [18] Y. Shi and R. C. Eberhart, “Parameter selection in particle swarm optimization,” *Evolutionary Programming of Lecture Notes in Computer Science*, vol. 1447, pp. 591–600, 1998.
- [19] A. Carlisle and G. Dozier, “An off-the-shelf pso,” in *The Particle Swarm Optimization Workshop*, 2001, pp. 1–6.

- [20] J. Kennedy, “Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance,” in *IEEE Congress on Evolutionary Computation*, 1999, pp. 1931–1938.
- [21] R. C. Eberhart and X. Hu, “Multiobjective optimization using dynamic neighborhood particle swarm optimization,” in *IEEE Congress on Evolutionary Computation*, Hawaii, 2002, pp. 1677–1681.
- [22] J. J. Liang and P. N. Suganthan, “Dynamic multi-swarm particle optimizer,” in *IEEE International Swarm Intelligence Symposium*, 2005, pp. 124–129.
- [23] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, “Particle swarm optimization algorithm with novel learning strategies,” in *International Conference on Systems, Man and Cybernetics*, The Netherlands, 2004.
- [24] J. J. Liang, P. N. Suganthan, A. K. Qin, and S. Baskar, “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions,” *To appear in IEEE Transaction on Evolutionary Computation*, 2006.
- [25] J. Robinson, S. Sinton, and Y. Rahmat-Samii, “Particle swarm, genetic algorithm, and their hybrids: Optimization of a profiled corrugated horn antenna,” in *IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting*, San Antonio TX, 2002.
- [26] X. H. Shi, Y. H. Lu, C. G. Zhou, H. P. Lee, W. Z. Lin, and Y. C. Liang, “Hybrid evolutionary algorithms based on PSO and GA,” in *IEEE Congress on Evolutionary Computation*, 2003, pp. 2393–2399.
- [27] Y.-p. Chen, “Extending the scalability of linkage learning genetic algorithms: Theory and practice,” Illinois Genetic Algorithms Laboratory (IlligAL) Department of General Engineering University of Illinois at Urbana-Champaign, Tech. Rep. 2004018, 2004.
- [28] D. E. Goldberg, “Genetic algorithms and walsh functions: Part i, a gentle introduction,” *Complex Systems*, vol. 3, no. 2, pp. 129–152, 1989.

- [29] —, “Genetic algorithms and walsh functions: Part ii, deception and its analysis,” *Complex Systems*, vol. 3, no. 2, pp. 153–171, 1989.
- [30] D. Thierens, “Analysis and design of genetic algorithms,” Ph.D. dissertation, Katholieke Universiteit Leuven, Leuven, Belgium, 1995.
- [31] D. E. Goldberg, K. Deb, and D. Thierens, “Toward a better understanding of mixing in genetic algorithms,” *Journal of the Society of Instrument and Control Engineers*, vol. 32, no. 1, pp. 10–16, 1993.
- [32] D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik, “Rapid accurate optimization of difficult problems using fast messy genetic algorithms,” in *Proc. of the Fifth Int. Conf. on Genetic Algorithms*, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 56–64.
- [33] H. Kargupta, “The gene expression messy genetic algorithm,” in *International Conference on Evolutionary Computation*, 1996, pp. 814–819.
- [34] M. Munetomo and D. E. Goldberg, “Linkage identification by non-monotonicity detection for overlapping functions,” *Evolutionary Computation*, vol. 7, no. 4, pp. 377–398, 1999.
- [35] T.-L. Yu, D. E. Goldberg, Y. Ali, and Y.-p. Chen, “A genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm,” in *Artificial Neural Networks in Engineering*, 2003, pp. 327–332.
- [36] G. R. Harik and D. E. Goldberg, “Learning linkage,” in *Foundations of Genetic Algorithms 4*, R. K. Belew and M. D. Vose, Eds. San Francisco, CA: Morgan Kaufmann, 1997, pp. 247–262.
- [37] Y.-p. Chen and D. E. Goldberg, “Introducing start expression genes to the linkage learning genetic algorithm,” 2002.

- [38] S. Baluja, “Population-based incremental learning: A method of integrating genetic search based function optimization and competitive learning,” Carnegie Mellon University, Tech. Rep. CMU-CS-94-163, 1994.
- [39] M. Pelikan and H. Mühlenbein, “The bivariate marginal distribution algorithm,” in *Advances in Soft Computing - Engineering Design and Manufacturing*, R. Roy, T. Furuhashi, and P. K. Chawdhry, Eds. London: Springer-Verlag, 1999, pp. 521–535.
- [40] G. Harik, “Linkage learning via probabilistic modeling in the ECGA,” 1999.
- [41] P. Bosman and D. Thierens, “Linkage informatino processing in distribution estimation algorithms,” in *Genetic and Evolutionary Computation Conference*, 1999, pp. 60–67.
- [42] M. Pelikan and D. E. Goldberg, “Escaping hierarchical traps with competent genetic algorithms,” in *Genetic and Evolutionary Computation Conference*, 2001, pp. 511–518.
- [43] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, “Linkage learning, estimation distribution, and bayesian networks,” *Evolutionary Computation*, vol. 8, no. 3, pp. 314–341, 2000.
- [44] K. Deb and D. E. Goldberg, “Analyzing deception in trap functions,” *Foundations of Genetic Algorithms 2*, pp. 93–108, 1994.
- [45] M. Tezuka, M. Munetomo, and K. Akama, “Linkage identification by nonlinearity check for real-coded genetic algorithms,” in *Genetic and Evolutionary Computation Conference*, 2004, pp. 222–233.
- [46] G. Syswerda, “Simulated crossover in genetic algorithms,” in *Foundations of Genetic Algorithms 2*, L. D. Whitley, Ed. Morgan Kaufmann, 1993, pp. 239–255.
- [47] A. E. Eiben, P.-E. Raué, and Z. Ruttkay, “Genetic algorithms with multi-parent recombination,” in *Parallel Problem Solving from Nature – PPSN III*, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Berlin: Springer, 1994, pp. 78–87.

- [48] J. Smith and T. C. Fogarty, “An adaptive poly-parental recombination strategy,” in *Proceedings of AISB-95 Workshop on Evolutionary computing*, 1995, pp. 48–61.
- [49] 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,” Natural Computin Laboratory Department of Computer Science National Chiao Tung University, Tech. Rep. NCL-TR-2005001, May 2005.
- [50] M. F. Tasgetiren, Y.-C. Liang, G. Gencyilmaz, and I. Eker, “Global optimization of continuous functions using particle swarm optimization,” 2005.
- [51] J. J. Liang and P. N. Suganthan, “Dynamic multi-swarm particle swarm optimizer with local search,” in *IEEE Congress on Evolutionary Computation*, vol. 1, 2005, pp. 522–528.
- [52] P. J. Ballester, J. Stephenson, J. N. Carter, and K. Gallagher, “Real-parameter optimization performance study on the cec-2005 benchmark with spc-pnx,” in *IEEE Congress on Evolutionary Computation*, 2005, pp. 498–505.
- [53] J. Ronkkonen, S. Kukkonen, and K. V. Price, “Real-parameter optimization with differential evolution,” in *IEEE Congress on Evolutionary Computation*, 2005, pp. 506–513.
- [54] A. K. Qin and P. N. Suganthan, “Self-adaptive differential evolution algorithm for numerical optimization,” in *IEEE Congress on Evolutionary Computation*, 2005, pp. 1785–1791.
- [55] A. Auger and N. Hansen, “A restart cma evolutionary strategy with increasing population size,” in *IEEE Congress on Evolutionary Computation*, 2005, pp. 1769–1776.
- [56] J. W. Allen and F. W. Bruce, *Power Generation, Operation, and Control*. New York: Wiley, 1984.

- [57] D. C. Walters and G. B. Sheble', "Genetic algorithm solution of economic dispatch with valve point loading," *IEEE Transaction on Power Systems*, vol. 8, no. 3, pp. 1325–1332, August 1993.
- [58] G. B. Sheble and K. Brittig, "Refined genetic algorithm - economic-dispatch example," *IEEE Transactions on Power System*, vol. 10, no. 1, pp. 117–124, 1995.
- [59] P. H. Chen and H. C. Chang, "Large-scale economic-dispatch by genetic algorithm," *IEEE Transactions on Power System*, vol. 10, no. 4, pp. 1919–1926, 1995.
- [60] S. Baskar, P. Subbaraj, and M. V. C. Rao, "Hybrid real coded genetic algorithm solution to economic dispatch problem," *Computers and Electrical Engineering*, vol. 29, no. 3, pp. 407–419, 2003.
- [61] T. Yalcinoz, H. Altun, and M. Uzam, "Economic dispatch solution using a genetic algorithm based on arithmetic crossover," in *IEEE Power Tech Conference*, 2001.
- [62] J. T, J. K, J. DN, and R. T, "Evolutionary programming techniques for different kinds of economic dispatch problems," *Electric Power Systems Research*, vol. 73, no. 2, pp. 169–176, 2005.
- [63] N. Shinha, R. Chakrabarti, and P. K. Chattopadhyay, "Evolutionary programming technique for economic load dispatch," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 1, pp. 83–94, 2003.
- [64] H. T. Yang, P. C. Yang, and C. L. Huang, "Evolutionary programming based economic dispatch for units with non-smooth fuel cost functions," *IEEE Transactions on Power System*, vol. 11, no. 1, pp. 112–117, 1996.
- [65] K. P. Wong and J. Yuryevich, "Evolutionary-programming-based algorithm for environmentally-constrained economic dispatch," *IEEE Transactions on Power System*, vol. 13, no. 2, pp. 301–306, 1998.
- [66] Z. L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *IEEE Transactions on Power System*, vol. 18, no. 3, pp. 1187–1195, 2003.

- [67] T. A. A. Victoire and A. E. Jeyakumar, "Hybrid pso-sqp for economic dispatch with valve-point effect," *Electric Power Systems Research*, vol. 71, no. 1, pp. 51–59, 2004.
- [68] J. B. Park, K. S. Lee, J. R. Shin, and K. Y. Lee, "A particle swarm optimization for economic dispatch with nonsmooth cost functions," *IEEE Transactions on Power System*, vol. 20, no. 1, pp. 34–42, 2005.
- [69] T. A. A. Victoire and A. E. Jeyakumar, "Discussion of "particle swarm optimization to solving the economic dispatch considering the generator constraints",," *IEEE Transactions on Power System*, vol. 19, no. 4, pp. 2121–2122, 2004.
- [70] W. M. Lin, F. S. Cheng, and M. T. Tsay, "An improved tabu search for economic dispatch with multiple minima," *IEEE Transactions on Power System*, vol. 17, no. 1, pp. 108–112, 2002.
- [71] Y.-M. Park, J. R. Won, and J. B. Park, "New approach to economic load dispatch based on improved evolutionary programming," *Eng. Intell. Syst. Elect. Eng. Commun*, vol. 6, no. 2, pp. 103–110, June 1998.

