Researches on Hierarchical Bare Bones Particle Swarm Optimization for Single-Objective Optimization Problems

GUO, Jia

(開始ページ / Start Page)
1
(終了ページ / End Page)
115
(発行年 / Year)
2020-03-24
(学位授与番号 / Degree Number)
32675甲第491号
(学位授与年月日 / Date of Granted)
2020-03-24
(学位名 / Degree Name)
博士(理学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)
(URL)
https://doi.org/10.15002/00022971

Researches on Hierarchical Bare Bones Particle Swarm Optimization for Single-Objective Optimization Problems

Jia Guo

Doctoral Dissertation Reviewed by Hosei University

Researches on Hierarchical Bare Bones Particle Swarm Optimization for Single-Objective Optimization Problems

Jia Guo

Abstract

In experiments and applications, optimization problems aim at finding the best solution from all possible solutions. According to the number of objective functions, optimization problems can be divided into single-objective problems and multi-objective problems. In this thesis, we focus on solutions for single-objective optimization problems. The purpose of this thesis is to clarify a means for realizing high search accuracy without parameter adjustment.

To achieve high accuracy results for single-objective optimization problems, there are four major points to note: the local search ability in unimodal problems, the global search ability in multimodal problems, diverse search patterns for different problems, and the convergence speed controlling. Population-based methods like the particle swarm optimization (PSO) algorithms are often used to solve single-objective optimization problems. However, the PSO is a parameter-needed method which means it needs to adjust parameters for better performances. The adjustment of parameters becomes an overhead when considering for engineering applications. Besides, the bare bones particle swarm optimization (BBPSO) algorithm is a parameter-free method but unable to change the search pattern according to different problems. Also, the convergence speed of the BBPSO is too fast to achieve high accuracy results. To cross the shortcoming of existing methods and present high accuracy results for single-objective optimization problems, seven different hierarchical strategies are combined with the BBPSO in this thesis. Four of the proposed

algorithms are designed with swarm division which are able to converge to the global optimum fast. The other three algorithms are designed with swarm reconstruction which are able to slow down the convergence and solve shifted or rotated problems. Moreover, no parameter adjustment is needed when controlling the convergence speed.

First of all, four algorithms with swarm division are proposed. In the pair-wise bare bones particle swarm optimization (PBBPSO) algorithm, the swarm splits into several search units. Two particle are placed in one unit to enhance the local search ability of the particle swarm.

To increase the global search ability, the dynamic allocation bare bones particle swarm optimization (DABBPSO) algorithm is proposed. Particles in DABBPSO are divided into two groups before evaluation according to their personal best position. One group is named as the core group (CG) and the other one is called the edge group (EG). The CG focuses on digging and trying to find the optimal point in the current local optimum. Conversely, the EG aims at exploring the research area and giving the whole swarm more chances to escape from the local optimum. The two groups work together to find the global optimum in the search area.

To solve the shifted of rotated problems, traditional methods usually need to increase the population size. However, the growth of population size may increase the computing time. To cross this shortcoming, a multilayer structure is used in the triple bare bones particle swarm optimization (TBBPSO) algorithm. The TBBPSO is able to present high accuracy results in shifted and rotated problems without the increasing of population size.

In real-world applications, optimization methods are required to solve different types of optimization problems. However, the original BBPSO can not change its search pattern according to different problems. To solve this problem, a bare bones particle swarm optimization algorithm with dynamic local search (DLS-BBPSO) is proposed. The dynamic local search strategy is able to provide different search patterns based on different questions.

In engineering applications, the optimization results can be improved by controlling the con-

vergence speed. Normally, traditional methods need parameter adjustment to control the convergence speed. It is difficult to adjust the parameters for every single problem. To solve this problem, three different reorganization strategies are combined with the BBPSO. In the bare bones particle swarm optimization algorithm with co-evaluation (BBPSO-C), a shadow swarm is used to increase the diversity of the original swarm. A dynamic grouping method is used to disperse both the shadow particle swarm and the original particle swarm. After the dispersion, an exchanging process will be held between the two swarms. The original swarm will be more concentrated and the shadow swarm will be more scattered. With the moving of particles between the two swarms, the BBPSO-C gains the ability to slow down the convergence without parameter adjustment.

With the improvement of technologies, it is possible to get high accuracy results with a long calculation. In the dynamic reconstruction bare bones particle swarm optimization (DRBBPSO) algorithm, a dynamic elite selection strategy is used to improve the diversity of the swarm. After elite selection, the swarm will be reconstructed by elite particles. According to experimental results, the DRBBPSO is able to provide high accuracy results after a long calculation.

To adapt to different types of optimization problems, a fission-fusion hybrid bare bones bare bones particle swarm optimization (FHBBPSO) is proposed. The FHBBPSO combines a fission strategy and a fusion strategy to sample new positions of the particles. The fission strategy aims at splitting the search space. Particles are assigned to different local groups to sample the corresponding regions. On the other side, the fusion strategy aims at narrowing the search space. Marginal groups will be gradually merged by the central groups until only one group is left. The two strategies work together for the theoretically best solution. The FHBBPSO shows perfect results on experiments with multiple optimization functions.

To conclude, the proposed hierarchical strategies provide each of the BBPSO-based algorithms variants with different search characteristics, which makes them able to realize high

search accuracy without parameter adjustment.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms	**Abbreviations**
AG	Assistant group
APSO	Adaptive particle swarm optimization
BBPSO	Bare bones particle swarm optimization
BBPSO-C	Bare bones particle swarm optimization with co-evaluation
CG	Core group
CLA-BBPSO	Cellular learning automata bare bones particle swarm optimization
CLPSO	Comprehensive learning particle swarm optimizer
DA	Dynamic allocation
DABBPSO	Dynamic allocation bare bones particle swarm optimization
DDS	Dynamic division strategy
DE	Differential evolution
DEPS	Dynamic elite particle selection
DG	Director group
DLS	dynamic local search
DLS-BBPSO	Bare bones particle swarm optimization algorithm with dynamic local search
DRBBPSO	Dynamic reconstruction bare bones particle swarm optimization

Acronyms	**Abbreviations**
EA	Evolutionary algorithms
EC	Evolutionary computation
EG	Edge group
ETI	Event-triggered impulsive
ETM	Event-triggered mechanism
FG	Follower Group
FHBBPSO	Fission-fusion hybrid bare bones bare bones particle swarm optimization
FIPS	Fully informed particle swarm
GA	Genetic Algorithm
GBDE	Gaussian bare-bones differential evolution
GO	Global Optimization
HO	Hierarchical operator
IP	Interaction probability
IPC	Impulsive control
LG	Leader Group
MG	Main Group
MHO	Mutated hierarchical operator
MTBBPSO	Mutated triple bare bones particle swarm optimization

Acronyms	Abbreviations
NDi-DE	Neighborhood and direction information based Differential evolution
OLPSO	Orthogonal learning particle swarm optimization
PBBPSO	Pair-wise bare bones particle swarm optimization
PSO	Particle swarm optimization
SE	Separate Exploration
SI	Swarm Intelligence
SMA-BBPSO	Bare bones particle swarm optimization with scale matrix adaptation
SOP	Single-objective Optimization Problem
TBBPSO	Triple bare bones particle swarm optimization
TOS	Theoretical optimal solution
WG	Worker Group

Chapter 1

Introduction

1.1 Research Background

During the last decade, the evolutionary computation (EC) started to receive significant attention in different fields[1]. Plenty of researchers engaged in theories and applications of the EC. Different types of evolutionary algorithms (EAs) including genetic algorithm (GA), differential evolution (DE), etc. have been developed for different requirements. As an important branch of the EC, the swarm intelligence (SI) has shown remarkable performances in the field of optimization.

Most of the SI algorithms are inspired from the natural creatures like the ant colony optimization (ACO) [2–4], the particle swarm optimization (PSO) [5], etc.. The PSO is inspired by bird flocking and fish schooling. Particles are used in the PSO to simulate the team behavior of fishes or birds. The PSO algorithms show significant performance in economic dispatch [6], power systems [7], and many other fields [8–10].

1.2 Research Motivation

1.2.1 The global optimization problems

The optimization problems appear everywhere in our lives and researches. For instance, if we want to travel to another city, we can choose to use a bus, a flight or a bicycle. Different choices will link to different fuel costs, time costs and danger levels. These problems can be summarized as a combination of optimization problems. These problems may be discontinuous or noise contained. In the research field, optimization problems exist in a wild range of subjects like mathematics, physical, chemistry, etc.. More precisely, the numerous global optimization (GO) problems can be described in Equation 1.1:

$$\begin{aligned} & f: \ X \to \mathbb{R} \\ & x^* \in X \\ & f(x^*) \leq f(X) \end{aligned} \tag{1.1}$$

where $X \subset \mathbb{R}^D$ is a nonempty compact set that contains all feasible solutions, D is the dimension of the problem, f is a real valued objective function, x^* is the theoretical optimal solution [11]. The purpose of an optimization algorithms is finding the x^*, even the objective functions maybe non-convex, multimodal, or badly scaled [12].

The GO is often described as a minimal problem because a maximum problem can be transfer to a minimal problem by Equation 1.2:

$$f(x) = (-1) * g(x) \tag{1.2}$$

where $g(x)$ is the original maximum problem, $f(x)$ is the target minimal problem. In experiments and applications, a GO problem is often described as a numerical optimization problem

which aims at finding the point with the smallest fitness value.

In this thesis, all functions we talk about are minimal numerical functions. The aim of our methods is finding a solution which can minimize the function. Hence, in this thesis, a best solution is a solution can minimize the function. In addition, all of the seven proposed methods are population-based algorithms. The optimization will be implemented by particles. For better explanation, in the following chapters, when we talk about " compare two particles" we mean compare the personal best position of the two particles. Moreover, a better particle is a particle with better position. A better position is a position make the functions reach a smaller value. The same definition applies to the word "best", "worse", and "worst".

1.2.2 Aim of this thesis

The single-objective optimization problem (SOP) is a basic and important part of GO problems. Most of GO problems can be treated as some combinations or variations of SOPs. Basically, SOPs contain unimodal functions and multimodal functions. A unimodal function contains only one local best in the feasible area while a multimodal functions contains more than one local best. A local best in a continuous function is a point with a reciprocal equals zero. Soling the unimodal problems and multimodal problems need different strategies. More complex, in real-world applications, the feature of problems are unknown. Hence, to sole SOPs, there four major sub-problems to cross.

(1) The convergence speed.

Different users have different needs with optimization problems. It is possible some of them need a fast response while others may prefer higher accuracy although it may take a longer time.

(2) The precision in unimodal problems.

In a population-based optimization algorithm, the optimization is implemented by search units. The search units sample the feasible area to find the global best solution. In the unimodal

problems, these units need to gather in a small area to increase the precision. However, some complex problems contain a wide range of gentle zones which might prevent the gathering of the units.

(3) The local minimal escape in multimodal problems.

The multimodal can be considered at a combination of several unimodal problems. Each local best will be a valley. If the search units are over concentrated, they might be trapped in some sub-valley and unable to reach the global best. If units disperse too much the precision will decline.

(4) The wide adaptability for multiple problems.

In most real-world applications, it is difficult to identify the types of a problem. Hence the algorithm need to have balanced ability on both unimodal problems and multimodal problems. In addition, in real applications, optimization problems might be complicated and noise-contained.

In the past few years, population-based methods like the particle swarm optimization (PSO) algorithm and the bare bones particle swarm optimization (BBOSO) algorithm are often used to solve single-objective optimization problems. However, the PSO is a parameter-needed method which means it needs to changes parameters for different problems. The BBPSO is a parameter-free method but lacks accuracy. Both of them are unable to completely solve the single-objective optimization problems. In this thesis, to completely solve these problems, seven different BBPSO-based algorithms are proposed. Several novel hierarchical evolutionary strategies are used in proposed methods to obtain better accuracy with fewer parameters.

In addition, in the application area, the needs of users become more diverse. For instance, applications on mobile devices need less storage space and faster calculation while applications in the research area need ultimate accuracy. However, existing methods are unable to serve these demands ideally. To solve these problems, four of the proposed algorithms are designed for fast calculation. Linear space complexity and linear time complexity are achieved. On the

other hand, three of the proposed algorithms are designed for ultimate accuracy. These methods can solve complex and hybrid problems. According to experimental results, proposed algorithms have better performance than existing methods.

1.3 The Overview of the Thesis

In summary, this thesis is organized as follows:

Chapter 1 presents the back ground and motivation of this thesis.

Chapter 2 introduces several evolutionary algorithms including the standard bare bones particle swarm optimization algorithms.

Chapter 3, 4, 5 and 6 propose four algorithms for fast convergence. These methods are composed by linear storage space and linear time complexity. Highlights and relationships of fast convergence algorithms are shown in Fig 1.3.

Chapter 3 proposes a pair-wise bare bones particle swarm optimization algorithm.

Chapter 4 proposes a dynamic allocation bare bones particle swarm optimization algorithm.

Chapter 5 proposes a triple bare bones particle swarm optimization algorithm.

Chapter 6 proposes a bare bones particle swarm optimization algorithm with dynamic local search.

Chapter 7, 8 and 9 propose three algorithms for slow convergence. These methods can be used for complex and hybrid problems. Highlights and relationships of slow convergence algorithms are shown in Fig 1.3.

Chapter 7 proposes a bare bones particle swarm optimization algorithm with co-evaluation.

Chapter 8 proposes a dynamic reconstruction bare bones particle swarm optimization algorithm.

Chapter 9 proposes a fission-fusion hybrid bare bones particle swarm optimization algo-

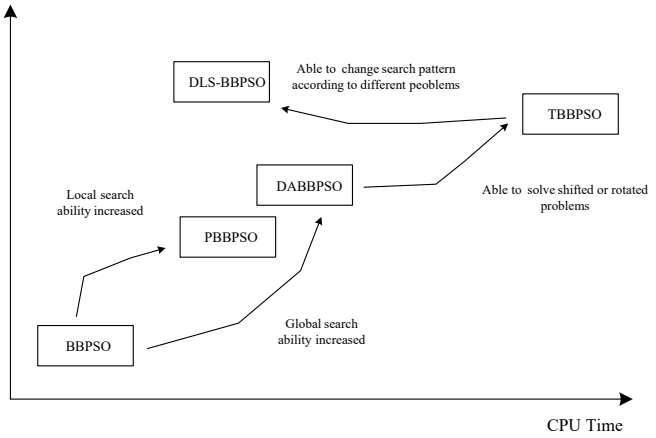

Figure 1.1: Highlights and relationships of fast convergence algorithms

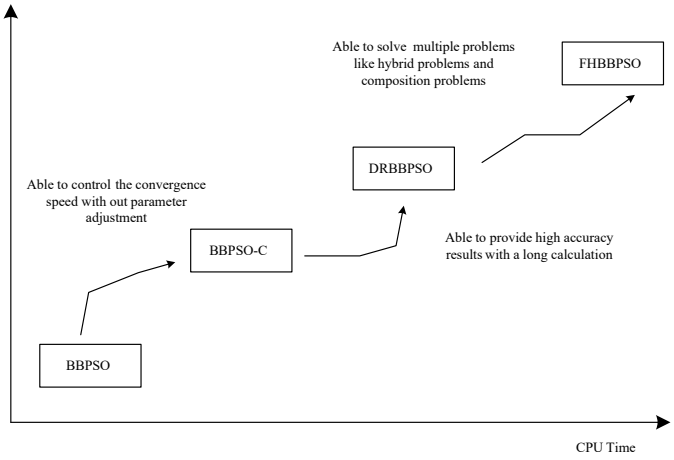

Figure 1.2: Highlights and relationships of slow convergence algorithms

rithm.

Chapter 10 presents the conclusion and future work of this thesis.

Chapter 2

Related Work

2.1 PSO and its variants

Particles in particle swarm optimization (PSO) [5] is a population-based optimization algorithm. The optimization is implemented by particles. PSO emulates the swarm behavior and the particles represent points in the search area. A particle represents a potential solution. In a numerical optimization problem, a potential solution is a set of coordinates. Also, particles are designed to have memories. For each particle, the best position it has ever been will be recorded as personal best position. From all particles, the best position discovers by the swarm will be recorded as the global best position. The moving trend of each particle is controlled by the velocity. More precisely, the velocity and next position of the ith particle is calculated by Equation 2.1:

$$\begin{aligned} v_{t+1}(i) &= w * v_t(i) + r_1 c_1 * (x_{best}(i) - x_t(i)) \\ &\quad + r_2 c_2 * (gbest - x_t(i)) \\ x_{t+1}(i) &= x_t(i) + c_3 v_{t+1} \end{aligned} \tag{2.1}$$

where $x_{best} = (x_{best}(1), x_{best}(2), ..., x_{best}(n))$ is a matrix for recording the best position each element has ever reached; *gbest* is the best position that all elements has ever reached; t is the iteration time; r_1 and r_2 are random number from 0 to 1; w, c_1, c_2 and c_3 are the parameters to control the convergence speed. And according to [13], c_1 and c_2 are usually set as 2.05; c_3 is usually set as 0.7298.

To increase the performance of PSO, a fully informed particle swarm (FIPS) is proposed by Mendes [14]. Particles in FIPS are affected by all of their neighborhoods rather than they are only affected by best neighborhoods in PSO. The FIPS has been applied in multimodal optimization problem and is improved in 2006 [15].

Liang proposes a comprehensive learning particle swarm optimizer (CLPSO), which uses a novel learning strategy whereby all other particles' historical best information is used to update a particle's velocity. The proposed strategy enables the diversity of the swarm to be preserved to discourage premature convergence [16].

2.2 BBPSO and its variants

In 2003, Kennedy [17] proposed the Bare Bones Particle Swarm Optimization (BBPSO), which is a simple version of PSO. The standard BBPSO is originally formulated as a means of studying the particle distribution of PSO. It cancels the velocity and uses a Gaussian distribution to sample the searching space. Particles generate with a normal distributed random number around the mean of personal best position and global best position on each dimension. During the iteration process, the personal and global best position keep detecting and exploring in the search area. Moreover, parameter-free means the algorithm can easily adapt to different problems. Hence, both varies BBPSO and numbers of methods based on it are proposed for real world applications. Omran [18] proposed a clustering method based on bare bones. The proposed algorithm finds

the centroids of a user specified the number of clusters, where each cluster groups together similar patterns. Moreover, the application of the proposed clustering algorithm to the problem of unsupervised classification and segmentation of images is investigated.

In bare bones particle swarm optimization (BBPSO), the next position of a particle is calculated by its personal best position and the global best position of the swarm. Because of the cancel of the velocity, BBPSO does not need any parameter. Moreover, in order to speed up convergence, Kennedy [17] introduce the interaction probability (IP) to BBPSO. The next position of the ith particle is calculated by Equation 2.2:

$$\begin{aligned} \mu &= \frac{p_i + gbest}{2} \\ \sigma &= |p_i - gbest| \\ x(t+1) &= \begin{cases} N(\mu, \sigma) \ \ (when \ \ \omega > 0.5) \\ \\ p_i \qquad\quad else \end{cases} \end{aligned} \tag{2.2}$$

where $P = (p_1, p_2, ..., p_n)$ is the best position of each particle; $gbest$ is the best position of the whole swarm; ω is a random number from 0 to 1.

To increase the accuracy during the optimization process, a bare bone particle swarm optimization with an integration of global and local learning strategies is proposed by Chen [19]. Moreover, Blackwell formulates the dynamic update rule of particle swarm optimization. This rule is expressed as a second-order stochastic difference equation. Also, general relations are derived for search focus, search spread, and swarm stability at stagnation. The relations are introduced to standard PSO, it's variant and BBPSO. Also, the simplicity of the Bare Bones swarm facilitates theoretical analysis, and a further no-collapse condition is derived, according to Blackwell's research [20].

In order to minimize the effects of the control parameters, in the research of Wang, a Gaussian

bare-bones differential evolution (GBDE) and its modified version are proposed [21]. The original differential evolution (DE) is a population-based stochastic optimization algorithm, which has already shown the noteworthy performance on both real-world and benchmark optimization problems. However, the DE is affected by several significant parameters in the algorithm. To solve this problem, the proposed algorithm presents some eliminating and dynamically adjusting strategies.

Krohling [22] introduces a jump strategy to bare bones particle swarm to solve the problem that BBPSO shows weak performance with functions with many local minima in high dimension area. The jump strategy is implemented based on the Gaussian or the Cauchy probability distribution. When there is no improvement of fitness value, the proposed will try to help the algorithm jump out of the current local wave. To verify the ability of the proposed algorithm, a set of well-known benchmark functions are used in the experiment. Simulation results show that the BBPSO with the jump strategy performs well in all functions investigated. Moreover, Chen [23] insists that in the BBPSO, if a particle is restricted to move to a new position only when the new position is better than its original position, the particle then retains the best position it ever found. Based on this observation, all personal best particles are no longer required. Hence a revised BBPSO is proposed by Chen. The proposed algorithm tends to eliminate personal best particle. This strategy makes the use of memory more efficient, especially when dealing with large scale problems or in microprocessor based applications.

Campos proposes a BBPSO with scale matrix adaptation (SMA-BBPSO) [24]. The proposed algorithm aims at solving the premature convergence problem of BBPSO with a strategy that selecting next position of a particle from a multivariate t-distribution with a rule for adaptation of its scale matrix. Also, to verify the searching ability of proposed algorithm, a set of well-known benchmark functions are used in the experiment. Moreover, a theoretical analysis is developed to explain how SMA-BBPSO works.

Richer points out that many foragers and wandering animals follow the Lévy distribution of steps. The Lévy distribution is introduced to both PSO and BBPSO algorithm. From the Lévy distribution BBPSO, long step and short step are combined to explore the searching area. Moreover, in the comparative experiment, proposed strategies show powerful ability for searching the global best position [25].

Vafashoar points out that BBPSO is highly prone to premature convergence and stagnation. To solve this problem, cellular learning automata bare bones particle swarm optimization (CLA-BBPSO), a new multi swarm version of BBPSO is proposed in his research. Different with standard BBPSO, several probability distributions are used in the proposed algorithm. Hence the diversity of particles increase. Moreover, a new approach for adaptive determination of covariance matrices, which are used in the updating distributions, has been proposed in the research. As the result that the searching ability of CLA-BBPSO has been confirmed by experiments, to improve the convergence speed can be the main future work [26].

In 2013, Cai *et al.* [27] proposed the neighborhood and direction information based DE (NDi-DE). The neighbor guided selection scheme for parents involved in mutation and the direction induced mutation strategies are used in the NDi-DE. The NDi-DE not only utilizes the information of neighboring individuals to exploit the regions of minima and accelerate convergence but also incorporates the direction information to prevent an individual from entering an undesired region and move to a promising area.

In 2016, Du *et al.* [28] proposed an event-triggered impulsive (ETI) control scheme for the DE. In the ETI scheme the event-triggered mechanism (ETM) and the impulsive control (IPC) are introduced to change the search performance of the population in a positive way. Also, both stabilizing and destabilizing impulses are used. Stabilizing impulses help the individuals with lower rankings instantly move to a desired state determined by the individuals with better fitness values. Destabilizing impulses randomly alter the positions of inferior individuals within

the range of the current population.

Chapter 3

A Pair-wise Bare Bones Particle Swarm Optimization Algorithm

In the BBPSO algorithm, the next position of every particle is calculated with the global best particle. Particles will get similar and gather together in a short generation times. All particles might be converged too early and unable to find the global best solution. To solve this problem, a pair-wise bare bones particle swarm optimization (PBBPSO) algorithm is proposed in this chapter. A separate iteration strategy is used in the pair-wise operator to enhance the diversity of the swarm. Each particle is designed to evolve with a partner particle. Different evolutionary strategies are used to different particles to slow down the diversity losing. The search accuracy of the swarm is increased by the pair-wise strategy. To verify the performance of the proposed algorithm, a set of well-known nonlinear benchmark functions are used in the experiment. In the control group, PSO and BBPSO are evaluated on the same functions. Finally, the experimental results and statistical analysis confirm the performance of PBBPSO.

3.1 Overview and Preliminaries

With the development of technologies, optimization problems grow high dimensional and multimodal. Traditional methods like the BBPSO are difficult to offer ideal results. To improve the performance of the BBPSO, researchers tried to introduce new methods or new strategies to BBPSO. The improvement of their results are exchanged by the calculation times. In addition, combining the BBPSO with inappropriate methods or harsh parameters may cause the algorithm difficult to apply to real-world applications. To solve these problems, the pair-wise strategy is proposed in this chapter.

3.2 Separate Exploration strategy for BBPSO

The separate exploration (SE) system is proposed in this section to keep the balance between exploration and exploitation during the optimization process. This system is inspired by human social class, which arranges different works for different classes. The integral swarm is divided into two different groups, the director group (DG) and the worker group (WG). The SE system is implemented by a random selection strategy. Each time two particles are placed into one team to compare with each other. The one who has a smaller fitness value will be in the DG while the other one will be placed in the WG. Moreover, this process is not static. This judgment will be carried out in each iteration. It is possible that one particle in the DG moves to WG in the next generation.

In order to maintain the stability of the algorithm, the next position of a particle in the DG is calculated by Equation 3.1:

$$
\begin{aligned}
\mu &= \frac{pbest(i) + gbest}{2} \\
\sigma &= |pbest(i) - gbest| \\
x_{t+1}(i) &= \begin{cases} N(\mu, \sigma) \;\; if(\omega > 0.5) \\ \\ pbest(i) \qquad else \end{cases}
\end{aligned}
\tag{3.1}
$$

where $Pbest = (pbest(1), pbest(2), ..., pbest(n))$ is the best position of each particle; $gbest$ is the best position of the whole swarm; ω is a random number from 0 to 1.

Also, if one particle gets a new global best position, its strategy in the next iteration will only depend on its current position. Specifically, the next position of a global best particle is calculated from Equation 3.2:

$$
x_{t+1}(i) = x_t(i) + \alpha * x_t(i) \tag{3.2}
$$

where α is a random number between -1 and 1. This means the particles that reach the global best position will swing with the current position as the center. When a particle reaches the top of its swarm, any other particles become useless reference objects. So it is reasonable to make the top particle swinging around itself.

Conversely, in order to jump out of current local minimal, the particles in the WG will move boldly with the Equation 3.3:

$$
\begin{aligned}
u &= pbest(director) \\
l &= |pbest(director) - pbest(worker)| \\
x_{t+1}(worker) &= N(u, l)
\end{aligned}
\tag{3.3}
$$

where $pbest = (pbest(1), pbest(2), ..., pbest(n))$ is a matrix for recording the personal best position of each particle; the $N(u, l)$ is a Gaussian distribution with a mean value u and a

standard deviation l.

3.3 Proposal of the Pair-wise BBPSO

On the basis of the SE system, the pair-wise bare bones particle swarm optimization (PBBPSO) is proposed in this section. The PBBPSO can solve both unimodal problems and multimodal problems without parameter adjustment.

3.3.1 Initialization

Initialization is the first step of the PBBPSO. We only need to know the number of particles N; the dimension of the problem D; the fitness function F, the max iteration times T, and the search range R. No parameter is needed to control the evolution. Then all particles are randomly spread in R. After that, first personal best positions, *pbest*, are calculated by F. Moreover, the global best position *gbest* can be obtained.

3.3.2 Pair-wise strategy

In order to balance the exploration and exploitation abilities of the particle swarm, a pair-wise strategy is introduced to BBPSO. The pair-wise strategy aims at slowing down the speed of swarm diversity losing. This is inspired by an old Chinese saying "never put all eggs in one basket." The saying means we may lose all eggs when accidents happen in the basket with all eggs. Applying this concept into optimization algorithms, every particle moving to the same global best position may cause the whole swarm losing diversity very fast, and increasing the possibility of fall into local minimal. The pair-wise strategy is proposed to ameliorate this situation. By offering different strategies to different groups, the pair-wise strategy aims at slowing down the diversity losing of the whole swarm.

Before iteration, two particles are randomly selected from the swarm. The one with a better position, which means it has a lower fitness value in a minimum problem, will be placed into the leader group (LG), and the other one will be in the follower group (FG). Then the two particles will update their position with the rules of their own group. This strategy will repeat until every particle is selected.

Particles in different group evolute with respective ways. Specifically, if the ith particle is in LG, the next position of the ith particle is selected by Equation 3.4:

$$\begin{aligned} u &= \frac{(pbest(i) + gbest)}{2} \\ l &= |pbest(i) - gbest| \\ x(i)_{new} &= N(u, l) \end{aligned} \tag{3.4}$$

where $pbest = (pbest(1), pbest(2), ..., pbest(n))$ is a matrix used for recording the best position each element has ever reached; $x(i)_{new}$ is the new position of the ith particle; $gbest$ is the best position that all element has ever reached; $N(u, l)$ is a Gaussian distribution with mean u and standard deviation l. This equation is inherited from the standard BBPSO.

Conversely, particles in FG aim at supporting the LG. If the ith particle is the corresponding leader of the jth particle, the next position of the jth particle is selected by Equation 3.5:

$$\begin{aligned} p &= \frac{(pbest(i) + pbest(j))}{2} \\ q &= |pbest(i) - pbest(j)| \\ x(j)_{new} &= N(p, q) \end{aligned} \tag{3.5}$$

where $Pbest = (pbest(1), pbest(2), ..., pbest(n))$ is a matrix used for recording the best position each particle has ever reached; $x(j)_{new}$ is the new position of the jth particle; $gbest$ is the best

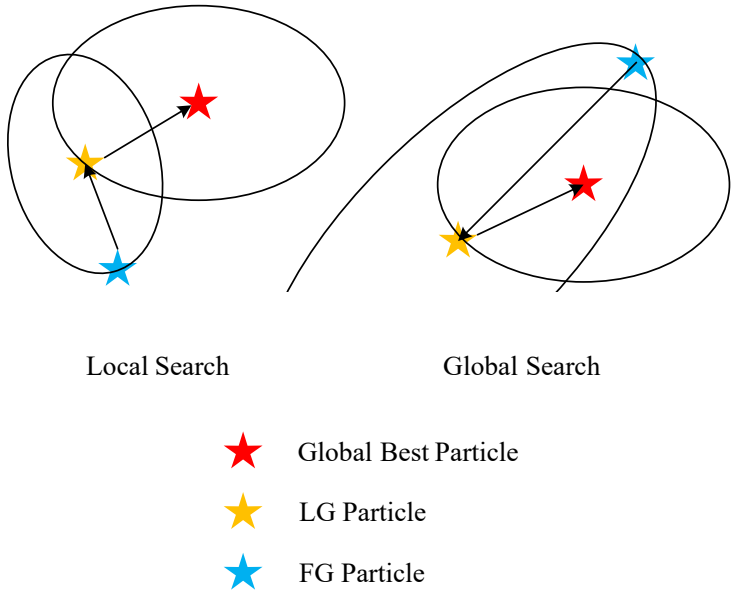

Figure 3.1: Possible situations of the pair-wise strategy

position that all element has ever reached; $N(p, q)$ is a Gaussian distribution with a mean p and a standard deviation q. The FG particles move toward to their leaders under this rule. Some possible situations of the pair-wise strategy are shown in Fig. 3.1.

According to Fig. 3.1, FG particles have an opportunity to implement both local and global search. This strategy can enhance the diversity of the swarm, also can do a deeply search around the current global best position. Moreover, this process is all random, and no parameter is needed, which means that the proposed system can be easily applied to different functions.

3.3.3 Pseudo code of PBBPSO

To show the working flow of PBBPSO, the pseudo code is given in Algorithm 1.

Algorithm 1 PBBPSO

Require: Max iteration time, T
Require: Dimension of the problem, D
Require: Fitness function, f
Require: Searching Range. R
Require: Number of particles n, n need to be an even
Require: Particle swarm $X = x_1, x_2, ...x_n$
Require: Personal best position $Pbest = p_1, p_2, ...p_n$
Require: Global best position $Gbest$
Ensure: all particles are in R

```
t = 1
while t <= T do
    while X ≠ ∅ do
        Randomly select and remove two particles, x_i and x_j, from X
        if f(x_i) < f(x_j) then
            Update x_i with Equation 3.4
            Update x_j with Equation 3.5
        else
            Update x_i with Equation 3.5
            Update x_j with Equation 3.4
        end if
    end while
    Update Pbest
    Update Gbest
    t = t + 1
end while
```

3.4 Experiments

3.4.1 Benchmark functions

In order to conduct a comprehensive evaluation of the optimization capabilities of PBBPSO, six famous benchmark functions including unimodal function and multimodal functions are selected for the experiment. These functions can be divided into three groups according to their properties:

1. Group 1, ($f_1 - f_2$), unimodal functions;
2. Group 2, ($f_3 - f_5$), multimodal functions without shifted or rotated operator;
3. Group 3, (f_6), shifted multimodal function.

Table 3.1: Benchmark Functions

Function	Search Range	Minimum	Reference
$f1$= Sphere Function	(100, 100)	0	[16][24]
$f_1(x) = \sum_{i=1}^{D} {x_i}^2$			
$f2$=Rosenbrock Function	$(-2.048, 2.048)$	0	[16][24]
$f_2(x) = \sum_{i=1}^{D-1} \left(100({x_i}^2 - x_{i+1})^2 + (x_i - 1)^2\right)$			
$f3$= Rastrigin Function	$(-5.12, 5.12)$	0	[16][24]
$f_3(x) = \sum_{i=1}^{D} ({x_i}^2 - 10\cos(2\pi x_i) + 10)$			
$f4$= Ackley Function	$(-32.768, 32.768)$	0	[16][24]
$f_4(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$			
$f5$= Griewank Function	$(-600, 600)$	0	[16][24]
$f_5(x) = \sum_{i=1}^{D} (\frac{{x_i}^2}{4000}) - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$			
$f6$= Shifted Rastrigin's Function	$(-5\ ,\ 5)$	-330	[24][29]
$f_6(x) = \sum_{i=1}^{D} ({z_i}^2 - 10\cos(2\pi z_i) + 10) + \mathit{shifted}, z = (x - o)$			
o is the shifted global optimum, $\mathit{shifted} = -330$			

These functions are considered in some early studies [16][29]. All of the 6 functions are minimum value problem; the summarize of all functions are shown in Table 3.1.

3.4.2 Results and discussion

To minimize the impact of accidental factors, the empirical error is calculated from 30 independent runs on each function with 1500 times iteration. The empirical error is defined as $\mid gbest - Minimun \mid$. Where *gbest* is the final global best value given by an algorithm, and $Minimun$ is the theoretical optimal solution of the function. Experimental results keep three

Table 3.2: Comparisons of empirical error between PSO, BBPSO, and BBPSOwj

Function	PSO[30]	BBPSO[17]	BBPSOwj[20]
	Mean	Mean	Mean
	SD	SD	SD
f_1	**1.13E−10**	2.48E−09	4.34E−03
	7.94E−11	3.96E−09	2.37E−02
f_2	5.26E+01	**3.25E+01**	6.50E+01
	4.20E+01	1.41E+01	4.22E+01
f_3	8.36E+01	4.96E+01	**1.11E+01**
	1.98E+01	1.27E+01	3.45E+00
f_4	1.90E+01	2.71E−01	**1.54E−01**
	3.59E+00	6.36E−01	3.55E−01
f_5	**2.10E−03**	1.35E−02	3.05E−02
	4.16E−03	1.86E−03	2.84E−02
f_6	6.70E+01	6.87E+01	**1.02E+01**
	1.54E+01	1.49E+01	2.86E+00

significant digits. Also, the dimension of each function is set as 30, the number of particles is set as 30.

In Table 3.2 and 3.3, the mean and standard division of the 30 independent results are displayed. Best results of each team are shown in boldface.

In group 1, PBBPSO gives excellent results with the two unrotated unimodal functions. The 3.37E−15 empirical error on f_1 has an obvious advantage than other algorithms. Because of the second rank algorithm, PSO only gives 1.13E−10. However, although PBBPSO still keeps

Table 3.3: Comparisons of empirical error between FIPS, Lévy BBPSO, and PBBPSO

Function	FIPS[14][15]	Lévy BBPSO[25]	PBBPSO
	Mean	Mean	Mean
	SD	SD	SD
f_1	1.03E−04	2.23E−10	**3.37E−15**
	6.95E−05	4.59E−10	4.11E−15
f_2	4.29E+01	6.01E+01	**2.85E+01**
	4.01E+01	8.55E+01	1.12E+01
f_3	7.78E+01	3.19E+01	**2.79E+01**
	1.63E+01	1.83E+01	6.36E+00
f_4	6.86E−01	1.27E+00	**4.47E−02**
	1.04E+00	8.70E−01	5.46E−01
f_5	5.53E−02	6.12E−01	**8.70E−03**
	8.15E−02	7.96E−03	8.50E−03
f_6	1.26E+02	**3.82E+01**	4.97E+01
	3.19E+01	6.18E+00	1.24E+01

its lead on f_2, the advantage decreases. The 2.85E+01 empirical error gets the 1st rank in all algorithm. But compares with FIPS, the second rank algorithm, the BBPSO gets a lead of 33.6%. These results are direct evidence that PBBPSO has good performance on unimodal functions. This can be attributed to the pair-wise system. When the minimum point is in the area that between FG particles and its leader, the crossover search enhance the probability of reaching the best point.

In group 2, PBBPSO gets one best and two second rank in all three unrotated multimodal functions. In f_3, PBBPSO is defeated by BBPSOwj, win the second place. The 2.79E+01 empirical error PBBPSO gives, is more than two times larger than BBPSOwj gives. However, the result of PBBPSO is 87.4% of the third algorithm. In f_4, PBBPSO beats all other algorithms. The second rank algorithm, BBPSOwj, gives a 1.54E−01 empirical error while PBBPSO gives 4.47E−02. In f_5, PBBPSO takes the third rank. The empirical error it gives is 8.70E−03, which is more than four times larger than the first rank algorithm. From the experimental results of group 2, it is able to confirm that PBBPSO can give better results than other algorithms on unrotated multimodal functions.

In group 3, PBBPSO gets the third rank. The empirical error of PBBPSO is more than four times larger than the empirical error of BBPSOwj.

To sum up, a ranking comparison is proposed to describe the searching ability of all algorithms in the experiment. Each function provides a rank value from 1 to 6 to all algorithms. The algorithm presents best results will get 1 and the algorithm presents worst results will get 6. The mean ranking value from all functions is calculated in the Table 3.4.

Table 3.4: Average ranking

Algorithm	Average Ranking
PBBPSO	1.86
BBPSOwj	3.00
PSO	3.71
BBPSO	3.57
Lévy BBPSO	4.29
FIPS	4.57

3.5 Summary

A pair-wise bare bones particle swarm optimization (PBBPSO), which can solve both unimodal and multimodal problems, is proposed in this chapter. The PBBPSO inherits advances from BBPSO such as simplicity and parameter-free. Moreover, it extends the searching concept from the original algorithm. In order to keep the diversity of the swarm and avoid premature convergence, the pair-wise strategy is used in the proposed algorithm. Particles are classified into two groups, the leader group and the follower group. Different iteration strategies are used to different groups to slow down the speed of diversity losing.

To verify the performance of PBBPSO, six famous benchmark functions are used in the experiment. Furthermore, severe variants of BBPSO and some other evolutionarily algorithms are also evaluated on the same functions as the control group. Finally, the experimental results and statistical analysis confirmed the performance of PBBPSO.

Chapter 4

A Dynamic Allocation Bare Bones Particle Swarm Optimization Algorithm

In **Chapter 3**, particles are divided into several different local groups. Each local group has a leader and a follower. Particles in a same local group has a close contact but there are no connection between the particles in different local groups. The isolation of local groups may reduce the search precision of the algorithm. To solve this problem, a dynamic allocation bare bones particle swarm optimization (DABBPSO) algorithm is proposed in this chapter. A pre-processing method is used in the DABBPSO to enhance the connection between particles.

4.1 Overview and Preliminaries

The bare bones particle swarm optimization algorithm is widely used in different areas. However, this algorithm may suffer from premature convergence by getting trapped in a local optimum when dealing with multimodal functions. To solve this problem, a dynamic allocation bare bones particle swarm optimization (DABBPSO) algorithm is proposed in this chapter. According to their personal best positions, particles are divided into two different local groups before evaluation. One group is named as core group (CG) and the other one is called the edge group (EG). The CG focuses on digging and trying to find the optimal point in the current local optimum. Conversely, the EG aims at exploring the research area and giving the whole swarm more chances to escape from the local optimum. The two groups work together to find the global optimum in the search area.

4.2 Proposal of the Dynamic Allocation

The dynamic allocation (DA) strategy is proposed in this section. This strategy is inspired by the construct of the human society. Different people are doing different jobs according to their talents or interests. This measure ensures the stability and robustness of human society. It is obvious that if everyone is assigned to do the same job our society will collapse; also, if everyone does the job he or she is not good at, our society may fall into chaos. The same logic applies to the swarm-based algorithms. The DA aims at balancing the exploration and exploitation of the swarm. All of the particles are divided into two groups by the DA. The DA considers that particles with better position have more chance to reach the global optimum while the particles with worse position have more chance to escape from current local optimum. Particularly, since all test functions in this research are minimal problems, a particle which has a smaller personal best value is considered to have a better position. Following this principle, the particles at the

better position will be in the CG and others will be in the EG. Before iteration, all particles will be ranked according to their personal best position. The top 50% particles will be in the CG. If the ith particle is in the CG, the next position of it will be selected by the Equation 4.1:

$$\begin{aligned} u &= \tfrac{(pbest(i)+gbest)}{2} \\ l &= |pbest(i) - gbest)| \\ x(i)_{new} &= N(u, l) \end{aligned} \tag{4.1}$$

where $Pbest = (pbest(1), pbest(2), ..., pbest(n))$ is a matrix used for recording the best position each particle has ever reached; $x(i)_{new}$ is the new position of the ith particle; $gbest$ stands for the global best position of the swarm; $N(u, l)$ is a Gaussian distribution. In addition, the rest particles will be put in the EG. All EG particles will find the next position by the Equation 4.2:

$$\begin{aligned} p &= \tfrac{(pbest(EG(j))+pbest(EG(j-1)))}{2} \\ q &= |pbest(EG(j)) - pbest(EG(j-1))| \\ x(j)_{new} &= N(p, q) \end{aligned} \tag{4.2}$$

where the $pbest(EG(i))$ stands for the personal best position of the EG particles; $gbest$ stands the global best position of the swarm; $N(p, q)$ is a Gaussian distribution.

Before iteration, an information initialization will be held. All particles will be randomly put into the searching area. Each particle can get its first personal best position and the swarm can get the first global best position. Then, the dynamic allocation system will disperse the swarm into two groups according to their personal best position. In each iteration, all particles will choose a new position under the direction of their team rules. To explain the working process of the DABBPSO more clearly, the pseudo code of the algorithm is given in Algorithm 2.

Algorithm 2 DABBPSO

Require: Max iteration time, T
Require: Fitness function, f
Require: Searching Range, R
Require: The number of all particles, n
Require: Particle swarm $X = x_1, x_2, ...x_n$
Require: Personal best position $Pbest = p_1, p_2, ...p_n$
Require: Global best position $Gbest$
Ensure: All particles are in R

$t = 1$
while $t <= T$ **do**
    Rank all the particles
    Put the top 50% particles into CG, put the rest into EG
    Choose a new position for CG particles according to Equation 4.1
    Choose a new position for EG particles according to Equation 4.2
    for i from $(1, n)$ **do**
        if $f(new\ x_i) < f(Pbest(i))$ **then**
            $X(i) = new\ x_i$
        end if
        if $f(new\ x_i) < f(Gbest)$ **then**
            $Gbest = new\ x_i$
        end if
    end for
    $t = t + 1$
end while

Table 4.1: Experimental environments

Operating system	Windows 10
Coding language	Matlab
CPU	Intel(R) Core(TM) i7
RAM	16 GB

4.3 Experiments

4.3.1 Experimental environments and benchmark functions

To verify the searching ability of the DABBPSO, a set of comprehensive experiments are proposed in this section. The experimental environments are displayed in Table 4.1.

Five famous benchmark functions are selected for experiment. The Rastrigin function is a typical example of a non-linear multimodal function. It is a non-convex function used as a performance test problem for optimization algorithms. The Ackley function is widely used for testing optimization algorithms. In its two-dimensional form, it is characterized by a nearly flat outer region, and a large hole at the center. The function poses a risk for optimization algorithms, to be trapped in one of its many local minima. The Griewank function has many widespread local minima, which are regularly distributed. The Sphere function and the Rosenbrock function are unimodal functions. More details for the test functions are displayed in Table 4.2.

To reduce the accidental error, each algorithm will have 30 independent runs on every function, and the empirical error is defined as $|gbest - TheoreticalMinimum|$. Specifically, all the algorithms have 1500 iteration times; and the dimension for all problems is 30. Experimental results keep three significant digits.

Table 4.2: Benchmark functions

Name	Optimal solution	optimal value
f_1 = Rastrigin	0.00^d	0
$f_1(x) = 10 * d + \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i))$		
$x \in (-5.12, 5.12)$		
f_2 = Ackely	0.00^d	0
$f_2(x) = -20 * \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$		
$x \in (-32.768, 32.768)$		
f_3 = Griewank	0.00^d	0
$f_3(x) = \sum_{i=1}^{D} (\frac{{x_i}^2}{4000}) - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$		
$x \in (-600, 600)$		
f_4 = Sphere	0.00^d	0
$f_4(x) = \sum_{i=1}^{D} {x_i}^2$		
$x \in (-100, 100)$		
f_5 = Rosenbrock	0.00^d	0
$f_5(x) = \sum_{i=1}^{D-1} \left(100({x_i}^2 - x_{i+1})^2 + (x_i - 1)^2\right)$		
$x \in (-2.048, 2.048)$		

4.3.2 Experimental results and discussion

The experimental results are presented in Table 4.3. The "Mean" in the table stands for the mean empirical error of the 30 test. The "SD" stands for the standard deviation of the 30 results. The best results of each function are shown in bold. In f_1, the mean empirical error of the DABBPSO is 24.80 % smaller than the result of the BBPSO, 52.57% smaller than the result of FIPS. In f_2, the mean empirical error of the DABBPSO is 81.51% smaller than the result of the BBPSO, 92.70% smaller than the result of FIPS. In f_3, the mean empirical error of the DABBPSO is 52.59% smaller than the result of the BBPSO, 87.16% smaller than the result of FIPS. In f_4, the empirical error of the DABBPSO is 99.99% smaller than the result of the BBPSO and the FIPS. In f_5, the empirical error of the DABBPSO is 41.72% smaller than the result of the FIPS.

4.4 Summary

A DABBPSO is proposed in this chapter. The DABBPSO is inspired by the social structure of humane society. In our society, different people do the different jobs according to their interest or talent. This division pattern activates the capacity of every single person. To simulate this process, the dynamic allocation strategy is proposed. In the proposed algorithm, particles with different personal best value are considered to have different capacity. To be specific, the particles which have higher positions in the personal best value ranking will be good at finding more accurately global best position while the rest are good at escaping from the current local optimum. Under this concept, the top particles are assigned into the CG while others are placed into the EG. The two groups work in different ways and their cooperation helps the algorithm to find the global best position. Moreover, like the human society, the group is not static. Particles may change their group in the next generation if they can get a better position than others.

Table 4.3: Comparisons of empirical error between PSO, BBPSO, and BBPSOwj

Function	FIPS[14][15]	BBPSO[17]	DABBPSO
	Mean	Mean	Mean
	SD	SD	SD
f_1	7.78E+01	4.96E+01	3.73E+01
	1.63E+01	1.27E+01	7.71E+00
f_2	6.86E−01	2.71E−01	5.01E−02
	1.04E+00	6.36E−01	2.74E−01
f_3	5.53E−02	1.35E−02	7.10E−03
	8.15E−02	1.86E−03	8.90E−03
f_4	1.03E−04	1.58E−06	8.88E−16
	6.95E−05	1.75E−06	2.53E−15
f_5	4.29E+01	9.47E+01	2.50E+01
	4.01E+01	8.55E+01	1.33E+00

To verify the search ability of the DABBPSO, five famous benchmark functions are used in the experiments. All of the experimental results confirmed the search ability of the DABBPSO.

Chapter 5

A Triple Bare Bones Particle Swarm Optimization Algorithm

In **Chapter** 3, the 2-particle evolutionary unit shows good performance on basic functions. But the optimization ability of PBBPSO is limited by its simple structure in each evolutionary unit. The precision of experimental results in the shifted and rotated functions are not good. Hence, to increase the precision of shifted and rotated unimodal problems, to increase the local minimal escape ability of shifted and rotated multimodal problems, a triple bare bones particle swarm optimization (TBBPSO) algorithm is proposed in this chapter. The size of the evolutionary unit is expanded and the structure becomes more complicated. Three particles are placed in one local group and a hierarchical method is used to classify particles in one local group. The experimental results confirm the search ability of the TBBPSO.

5.1 Overview and Preliminaries

The bare bones particle swarm optimization (BBPSO) is a population-based algorithm. The BBPSO is famous for easy coding and fast applying. The Gaussian distribution is used to control the behavior of the particles. However, every particle learning from a same particle may cause the premature convergence. To solve this problem, a new triple bare bones particle swarm optimization algorithm is proposed in this section. Three random particles are placed in one group and exchanging information during the iteration process. And a hierarchical method is used in every group. Therefore, the swarm gains an increase in diversity and more chances to escape from the local optimum. Moreover, a mutated structure for the local group is presented in this paper. To verify the ability of the proposed algorithms, a set of well-known benchmark functions are used in the experiment. Also, to make the experiment more persuasive, several evolutionary computation algorithms are applied to the same functions as the control group. The experimental results show that the proposed algorithms perform well in the test functions.

5.2 Proposal of the TBBPSO

The TBBPSO divides the swarm into numbers of local groups. Each group is composed by leader and teammates. The particle has the best position of the group will be the leader while others will be its teammates. The leader of each group is in charge of communicating with the global best particle and teammates are responsible for learning from their leaders.

In the standard BBPSO, the structure of the swarm is simple and direct. Each particle is treated equally. Every particle gets the information from the global best particle and chooses a new position by a Gaussian distribution. This is an effective strategy but convergence at a single particle may cause the premature convergence. To solve this problem, a hierarchical operator (HO) and mutated hierarchical operator (MHO) are proposed here.

5.2.1 The hierarchical operator

In the HO, every three particles are grouped in one team. The grouping process is all random. After that, the one who has a best position of the team will be the leader and the other two will be the teammates. The teammates and the leader will iterate under different rules. Teammates will gain information from their leader while the leader will gain information from the global best particle. The next position of a leader is selected by Equation 5.1:

$$\begin{aligned} \alpha &= \frac{(pbest(leader) + gbest)}{2} \\ \beta &= |pbest(leader) - gbest| \\ x(leader) &= N(\alpha, \beta) \end{aligned} \tag{5.1}$$

where the $pbest(leader)$ is the personal best position of a leader; the $gbest$ is the global best position of the whole swarm; $x(leader)$ is the next position we want to calculate, $N(\alpha, \beta)$ is the Gaussian distribution with a mean α and a standard deviation β.

On the other hand, the teammates will learn from their own leader. Their next position is selected by Equation 5.2:

$$\begin{aligned} \eta &= \frac{(pbest(leader) + pbest(teammate))}{2} \\ \theta &= |pbest(leader) - pbest(teammate)| \\ x(teammate) &= N(\eta, \theta) \end{aligned} \tag{5.2}$$

where the $pbest(teammate)$ is the personal best position of a teammate;the $pbest(leader)$ is the personal best position of a leader; $x(teammate)$ is the next position we want to calculate. By using this differentiated management, leaders and teammates perform their duties, seeking the global optimum in the search space.

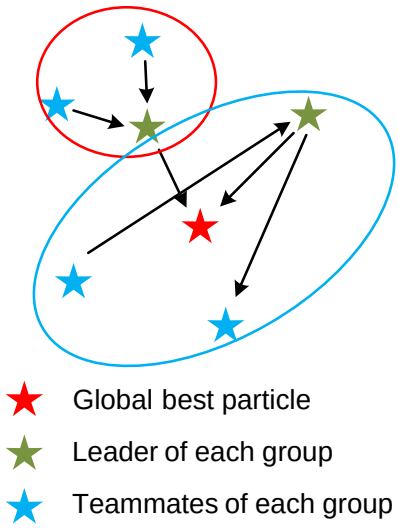

Figure 5.1: The lcoal structure of the TBBPSO

Generally speaking, the structure of HO is described in Fig 5.1. It can be found that the leader of each group communicate with the global best particle and the teammates gain the information from their leaders. This is essentially different from the BBPSO. The BBPSO has only one search center, the global best particle. But the HO has numbers of searching centers, a global best particle and plenty of leaders. This gives the swarm more chance to escape from a local optimum. Since the selection of teammates is random, it is possible that all members of a team gathered at a narrow region. For instance, in Fig 5.1, the local search is done by the group in the red circle. Also, like the group in the blue circle of the Fig 5.1, there is an opportunity that particles in a group are dispersed and will process a global search.

5.2.2 The local structure with mutation

It is easy to recognize that there is another structure for a 3-particle group. In this section, a mutated hierarchical operator (MHO) is used in the mutated triple bare bones particle swarm optimization (MTBBPSO).

In a specific group of HO, two teammates will learn from the same leader. This form likes

a miniature version of BBPSO. Hence there is a probability that both the HO and the BBPSO were caught in the same local optimum. So we change the structure of HO to avoid this situation. The structure of MHO is shown in 5.2. We considered that is maybe too general to make all teammates in a group to learn from their leader. So a simple level system is used in the MHO to make the algorithm more meticulous. Two particles are not treated equally in the group. The teammate with a worse position is ordered to learn from the better teammate. And the better teammate is ordered to learn from the leader. By organizing the structure like this, the deployment of a information exchange chain is completed. The passing down of the information from the global best particle gives the MHO more chance to jump out from a local optimum.

In addition, the iteration pattern of the MHO is similar with the HO. The only different is about the next position of the worst teammate. In particular, the worst teammate will iterate by Equation 5.3:

$$\begin{aligned} \xi &= \frac{(pbest(teammate1) + pbest(teammate2))}{2} \\ \omega &= |pbest(teammate1) - pbest(teammate2)| \\ x(worst) &= N(\xi, \omega) \end{aligned} \tag{5.3}$$

where the $pbest(teammate1)$ and the $pbest(teammate2)$ are the personal best position of the two teammates. By setting the teammates to two levels, particles gain more chance to explore the search space and more chance to escape from the local minimum. Also, the local search and the global search can be done by the simple strategy.

5.2.3 The process of the proposed algorithms

To explain the proposed algorithms more clearly, the pseudo code of the TBBPSO and MTBBPSO is given in Algorithm 3.

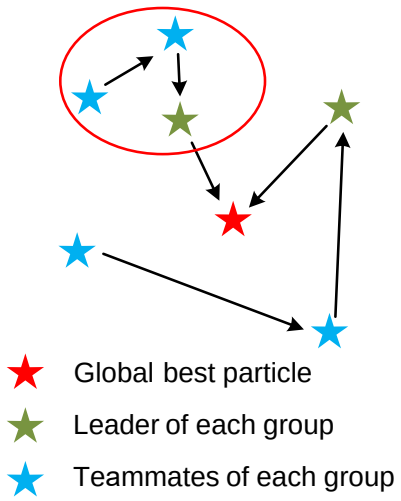

Figure 5.2: The local structure of the MTBBPSO

Algorithm 3 TBBPSO & MTBBPSO

Require: Max iteration time, T
Require: Fitness function, F
Require: Search Space, R
Require: Dimension of the function. D
Require: Particle swarm $X = x_1, x_2, ...x_n$
Require: Personal best position $Pbest = p_1, p_2, ...p_n$
Require: Global best position $Gbest$
Ensure: All particles are in R

$t = 1$
while $t <= T$ **do**
  while $X \neq \varnothing$ **do**
    Select 3 particles from X
    Sort them by personal best position
    For instance: $Pbest_i < Pbest_j < Pbest_k$
    Choose a new position for x_i with Equation 5.1
    Choose a new position for x_j with Equation 5.2
    if The algorithm is TBBPSO **then**
      Choose a new position for x_k with Equation 5.2
    end if
    if The algorithm is MTBBPSO **then**
      Choose a new position for x_k with Equation 5.3
    end if
  end while
  Update $Pbest$
  Update $Gbest$
  $t = t + 1$
end while

5.3 Experiments

5.3.1 Experimental method

To verify the ability of TBBPSO and MTBBPSO, an experiment is proposed in this section. A set of comprehensive benchmark functions are used in the experiment. They are selected from the CEC 2005 test functions [31] and more details are shown in Table 5.1. All of the 6 functions are minimum value problem and have been shifted or rotated. They are divided into two groups according to their features:

(1) Group 1, high dimension, unimodal functions with only one global optimum in the search region, $f_1 - f_3$;

(2) Group 2, high dimension, complex multimodal functions with several local minimum in the search region , $f_4 - f_6$;

In order to reduce the error caused by the accident , the empirical error is calculated from 30 independent runs while each algorithm has 300,000 iteration times. The empirical error is defined as $\mid result - TOS \mid$, where the $result$ is the global best value given by an algorithm after its final iteration, the TOS is the theoretical optimal solution (TOS) of the function. The dimension of all functions is 30. For all the test functions, the TOS are shifted to a new value different from the zero point. Moreover, the adaptive PSO (APSO) [32], the comprehensive learning PSO (CLPSO) [16] and the orthogonal learning PSO (OLPSO) [33] are used in the control group. According to the original references, the size of population is set to 20 for APSO, 40 for CLPSO and OLPSO, 30 for the rest 2 algorithms. More details about the algorithms can be found in [34]. Experimental results keep three significant digits.

Table 5.1: Benchmark Functions

Function
f_1 = Shifted Sphere Function $f_1(x) = \sum_{i=1}^{D} {z_i}^2 + \mathit{shifted},\ z = x - o$ $x \in (-100, 100)$, $\mathit{shifted} = -450$, o is the shifted global optimum, global minimum=-450
f_2 = Shifted Schwefel's Problem 1.2 $f_2 = \sum_{i=1}^{D} (\sum_{j=1}^{i} z_i)^2 + \mathit{shifted},\ z = x - o$ $x \in (-100, 100)$, $\mathit{shifted} = -450$, o is the shifted global optimum, global minimum=-450
f_3 = Shifted Rotated High Conditioned Elliptic Function $f_3(x) = \sum_{i=1}^{d} (10^6)^{\frac{i-1}{d-1}} z_i^2 + \mathit{shifted},\ z = (x - o) * M$, M: orthogonal matrix $x \in (-100, 100)$, $\mathit{shifted} = -450$, o is the shifted global optimum, global minimum=-450
f_4 = Shifted Rotated Griewank's Function without Bounds $f_4(x) = \sum_{i=1}^{d} \frac{z_i^2}{4000} - \prod_{i}^{d} \cos(\frac{z_i}{\sqrt{i}}) + 1 + \mathit{shifted},\ z = (x - \mathit{optimum}) * M$ $x \in (0, 600)$, $\mathit{shifted} = -180$, o is the shifted global optimum, global minimum=-180 M': linear transformation matrix, condition number=3, M=M'(1+0.3 $\lvert N(0,1) \rvert$)
f_5 = Shifted Rotated Rastrigin's Function $f_5(x) = \sum_{i=1}^{D} (\frac{{z_i}^2}{4000}) - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + \mathit{shifted}, z = (x - o) * M$ $x \in (-5, 5)$, $\mathit{shifted} = -330$, o is the shifted global optimum M: linear transformation matrix, condition number=2, global minimum= -330
f_6 = Shifted Rotated Weierstrass Function $f_6(x) = \sum_{i=1}^{D} (\sum_{k=0}^{20} [a^k cos(2\pi b^k (z_i + 0.5))]) - D \sum_{k=0}^{20} [a^k cos(2\pi b^k * 0.5)] + \mathit{shifted}$ $x \in (-0.5, 0.5), a = 0.5, b = 3, z = (x - o) * M, o$ is the shifted global optimum M: linear transformation matrix, condition number=5, shifted=90, global minimum= 90

Table 5.2: Comparisons of empirical error between BBPSO, FIPS, BBPSOwj, TBBPSO and MTBBPSO

Function	Mean/SD	f_1	f_2	f_3
APSO [32][34]	Mean	7.01E−14	9.97E−13	3.96E+05
	SD	2.45E−14	1.79E−12	1.59E+05
CLPSO [16][34]	Mean	5.68E−14	8.79E+02	1.67E+07
	SD	**0E+00**	1.79E+02	4.66E+06
OLPSO [33][34]	Mean	**0E+00**	1.50E+01	1.46E+07
	SD	**0E+00**	1.23E+01	5.33E+06
TBBPSO	Mean	**0E+00**	**0E+00**	**1.19E+05**
	SD	**0E+00**	**0E+00**	**9.11E+04**
MTBBPSO	Mean	**0E+00**	**0E+00**	1.35E+05
	SD	**0E+00**	**0E+00**	2.81E+05

5.3.2 Experimental results and discussion

The experimental results are shown in the Table 5.2 and 5.3. In the Table 5.2 and 5.3, the "mean" stands for the mean empirical error of 30 independent runs; the "SD" stands for the standard deviation of the 30 results. Moreover, the best results of each team are shown in **boldface**. For easy and accurate discussing, we define that if two results have a gap more than ten times, we will describe as "significant" or "outstanding". If the difference between two results is smaller than 20%, we will consider that they are "similar". Also, if one algorithm reaches the TOS while others not, we will consider the algorithm has a significant advantage.

On the f_1, the OLPSO, the TBBPSO and the MTBBPSO find the exact global best point in

Table 5.3: Comparisons of empirical error between BBPSO, FIPS, BBPSOwj, TBBPSO and MTBBPSO

Function	Mean/SD	f_4	f_5	f_6
APSO [32][34]	Mean	4.70E+03	1.50E+02	2.78E+01
	SD	2.34E−04	6.25E+01	3.16E+00
CLPSO [16][34]	Mean	4.70E+03	1.14E+02	2.70E+01
	SD	6.78E−12	**1.50E+01**	**1.71E+00**
OLPSO [33][34]	Mean	4.70E+03	1.10E+02	**2.55E+01**
	SD	**1.61E−12**	3.12E+01	2.95E+00
TBBPSO	Mean	**1.77E−02**	**7.50E+01**	3.17E+01
	SD	1.71E−02	4.26E+01	4.17E+00
MTBBPSO	Mean	2.29E−02	9.38E+01	2.83E+01
	SD	3.19E−02	3.36E+01	5.51E+01

all of the 30 independent runs. The APSO gives an empirical error of 7.01E−14 while CLPSO gives 5.68E−14. On the f_2, both of the proposed algorithms re The swarm converge at the theoretical optimal solution in all 30 independent runs. On the other hand, the APSO gives 7.01E−14 on f_1 and 9.97E−13 on f_2. The result is very close to the theoretical optimal solution but we still can say that the TBBPSO and the MTBBPSO have a significant advantage over the other 3 algorithms on f_1 and f_2. On the f_3, the TBBPSO offers the best result among all the test functions while the MTBBPSO takes the second place. The results of the TBBPSO and the MTBBPSO are similar but the TBBPSO has an advantage of 11.85%. In addition, the results of the TBBPSO has an advantage of 28.74% over the results of the CLPSO.

On the f_4, the results of the TBBPSO and the MTBBPSO have a significant advantage over the results of other 3 algorithms. The TBBPSO takes the first place among all the algorithms. It has an advantage of 13.97% over the MTBBPSO. On the f_5, the TBBPSO gives the best result among all the algorithms. Compared with the OLPSO, the result of TBBPSO has an advantage of 31.82%. The MTBBPSO gets the second position in the competition. On the f_6, all of the algorithms perform similarly. The result of CLPSO has an advantage of 4.59% over the result of the MTBBPSO. The results indicate that the swarm of the TBBPSO failed to escape some local optimum. So is reasonable to consider that the hierarchical construct works not well on this function.

Generally speaking, the performance of the TBBPSO and the MTBBPSO is better than the other algorithms. In the unimodal group, the TBBPSO and the MTBBPSO beat all the other algorithms. It is reasonable to conjecture that the constructs of the proposed algorithms can effectively explore the searching space. On the other hand, the proposed algorithms keep leading in the multimodal group but the advantage is not as obvious as in the unimodal group. Also, the results of all algorithms on f_6 are similar. We can consider that constructs of the TBBPSO and the MTBBPSO are lack at accurate on this kind functions. More importantly, the time

complexity of the two proposed algorithms is o(n). So improving the accuracy of the algorithms and exploiting their applying area should be an important and possible future work.

5.4 Summary

A triple bare bones particle swarm optimization (TBBPSO) algorithm and a mutated triple bare bones particle swarm optimization (MTBBPSO) algorithm are proposed in this chapter. Particles are separated into different groups to increase the diversity of the swarm. Specifically, particles in a group play different roles. The leader of a group is responsible for exchanging information with the global best particle while the teammates are focus on learning from their leaders. Moreover, the structure is not frozen. A teammate can be a leader in the next iteration if it can get a better position than the original leader. On the other side, the mutated version has one leader, one senior teammate and one normal teammate in one group.

To verify the performance of the TBBPSO and the MTBBPSO, both unimodal and multimodal benchmark functions are used in the experiments. To make the experiments more persuasive, several evolutionary algorithms are also applied to the same function. The experimental results confirm the abilities of proposed algorithm on the nonlinear functions.

Since the iteration time and the population number are the important parameters in the proposed algorithms, one of the interesting future work is trying to eliminate these human setting numbers. Also, we are interested in introducing the TBBPSO to real-world functions.

Chapter 6

A Bare Bones Particle Swarm Optimization Algorithm with Dynamic Local Search

In previous chapters, proposed algorithms are working with static local structures. The number of particles in each local group is changeless. This makes the algorithms unable to solve different types of problems. To solve this problem, a bare bones particle swarm optimization algorithm with dynamic local search (DLS-BBPSO), a complete parameter-free method is proposed in this chapter. Particles are divided into several local groups while the number of local groups and the number of particle in each local group are dynamic.

6.1 Overview and Preliminaries

Swarm intelligence algorithms are wildly used in different areas. The bare bones particle swarm optimization (BBPSO) is one of them. In the BBPSO, the next position of a particle is chosen from the Gaussian distribution. However, all particles learning from the only global best particle may cause the premature convergence and rapid diversity-losing. Thus, a BBPSO with dynamic local search (DLS-BBPSO) is proposed to solve these problems. Also, because the blind setting of local groups may cause an unpredictable increase in the time complexity, a dynamic strategy is used in the process of local group creation to avoid this situation.

6.2 Proposal of the bare bones particle swarm optimization with dynamic local search

In this section, the bare bones particle swarm optimization algorithm with dynamic local search (DLS-BBPSO) will be presented. As it is introduced, the DLS-BBPSO is a parameter free algorithm for single objective problems. It means that the proposed algorithm can adapt different functions without human intervention and adjustment. Unlike other variations of BBPSO, no additional calculations are introduced during the iteration process in the DLS-BBPSO. In the following subsections, each step of the DLS-BBPSO will be presented.

6.2.1 Initialization

Initialization is the first step of the DLS-BBPSO. As a parameter-free algorithm, only the data connected to test functions are needed before an experiment. In particular, necessary messages are: the number of particles N; the dimension of the problem D; the fitness function F; the exploring area, R; the max iteration times T.

After all information is inputted, all particles will be randomly spread in R. Then the first version of personal best positions, $pbest$, can be calculated by F. Meanwhile, the global best position, $gbest$, can be obtained. It can be found that no additional parameter is needed during the initialization process, which means the DLS-BBPSO can be easily applied to different problems. This is a great advantage when compared with parameter-needed algorithm in the real-world problems.

6.2.2 Dynamic local search system

In order to increase the diversity of the swarm during the iteration process, a dynamic local search (DLS) system is introduced to the standard BBPSO. In the DLS, particles are classified into different groups. Each group has only one leader and several teammates. Since all of the test functions are minimum problems, a particle with smaller fitness value is set to have a better position. At the beginning, the DLS will select a random particle to be the first " current leader", then it will select another random particle to compare with the "current leader". If the position of the selected particle is better than the "current leader", the selected particle will be a new "current leader". Otherwise, it will be a teammate of the "current leader". The DLS system will keep doing this until all particles are selected. It can be seen that the number of groups and the number of teammates of each group is not static. The process of the grouping is all random and no parameter is needed. This is the reason that the method named as "dynamic local search".

6.2.3 Evaluation

As a variation of standard bare bones particle swarm algorithm, DLS-BBPSO inheres the Gaussian distribution from BBPSO. The selecting mode is used both in and out groups. In particular, the next position of each leader will be selected by Gaussian distribution with a mean $(leader + gbest)/2$ and a standard deviation $|leader - gbest|$. More details are in the Equation

6.1:

$$\begin{aligned} u &= \frac{(pbest(leader) + gbest)}{2} \\ l &= |pbest(leader) - gbest| \\ x_t(i) &= N(u, l) \end{aligned} \tag{6.1}$$

where $pbest(leader)$ is the personal best position best position of each leader; $gbest$ is the best position that all element has ever reached; $N(u, l)$ is a Gaussian distribution with a mean u and a standard deviation l. This equation is inherited from standard BBPSO.

Conversely, teammates in groups will not move to the global best position like the BBPSO. The next position of each teammate will be selected by the Gaussian distribution with a mean $(leader + teammate)/2$ and a standard deviation $|leader - teammate|$. More details are in the Equation 6.2:

$$\begin{aligned} u &= \frac{(p(leader) + p(teammate))}{2} \\ l &= |p(leader) - p(teammate)| \\ x_t(i) &= N(u, l) \end{aligned} \tag{6.2}$$

where $p(teammate)$ is the personal best position of a teammate particle and $p(leader)$ is the personal best position of the teammate's corresponding leader.

Specifically, the comparison of the evolution pattern between the DLS-BBPSO and the standard BBPSO is shown in Figure 6.1. It can be observed that all particles learn from the only global best particle in BBPSO while only the leader of each group will learn from the global best in DLS-BBPSO. With the teammates' learning from their leaders, the swarm gains more diversity and more chance to escape from the local minimum. To explain the DLS-BBPSO more clearly, the pseudo code is given in Algorithm 4.

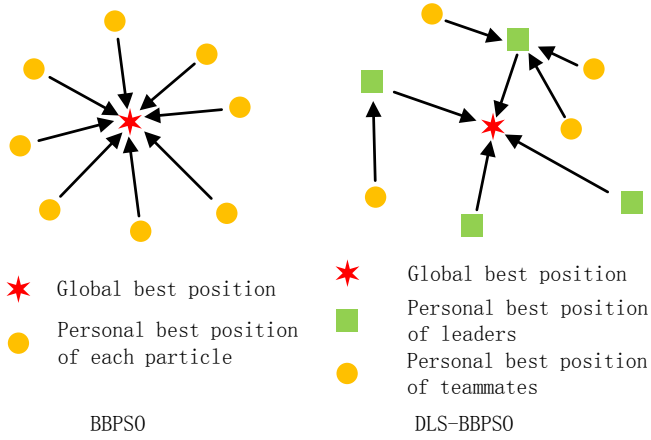

Figure 6.1: The comparison between BBPSO and DLS-BBPSO

Algorithm 4 DLS-BBPSO

Require: Fitness function, F
Require: Search Space, R
Require: Max iteration time, T
Require: Dimension of the function, D
Require: Particle swarm, $X = (x_1, x_2, ...x_n)$
Require: Personal best position, $Pbest = (p_1, p_2, ..., p_n)$
Require: Global best position, $Gbest$

$t = 1$
while $t <= T$ **do**
  From X, select and remove a random particle x_k as the first "current leader"
  while $X \neq \varnothing$ **do**
    From X, select and remove a random particle x_i from X
    if $p(x_i) < p(current\_leader))$ **then**
      Choose a new position for "current leader" with Equation 6.1
      Make $p(x_i)$ to be a new "current leader"
    else
      Make $p(x_i)$ to be a teammate of the "current leader"
      Choose a new position for x_i with Equation 6.2
    end if
  end while
  Update $Pbest$
  Update $Gbest$
  Destroy the construction of local groups
  Put all particles into X
  $t = t + 1$
end while

Table 6.1: Benchmark Functions

Function	Search Range	Referance
f_1 = SPHERE FUNCTION	(100, 100)	[24][16]
f_2 = ROSENBROCK FUNCTION	$(-2.048, 2.048)$	[24][16]
f_3 = RASTRIGIN FUNCTION	$(-5.12, 5.12)$	[24][16]
f_4 = ACKLEY FUNCTION	$(-32.768, 32.768)$	[24][16]
f_5 = GRIEWANK FUNCTION	$(-600, 600)$	[24][16]
f_6 = WEIERSTRASS FUNCTION	$(-0.5, 0.5)$	[24][16]

6.3 Experiments

6.3.1 Experimental method

To verify the search ability of DLS-BBPSO, a set of comprehensive benchmark functions are chosen for the experiment. They are divided into two groups according to their properties:

(1) $f_1 - f_2$, unimodal functions with only one global optimum in the search area;

(2) $f_3 - f_6$, complex multimodal functions with several local minimum in the search area.

All of the 6 functions are minimum value problem. Also, the summarize of all functions are shown in Table 6.1. Meanwhile, a control group is set to increase the persuasive of the experiment. The setting of the group is considered in an earlier research [24]. The population of each function is 30. Experimental results keep three significant digits.

6.3.2 Experimental results and discussion

In order to minimize accidental errors, the empirical error is calculated from 30 independent runs while each algorithm has 1500 iteration times. The empirical error is defined as $\mid gbest - Optimum \mid$, where the *gbest* is the global best value given by an algorithm after its last iteration,

Table 6.2: Comparisons of empirical error between PSO, BBPSO and FIPS

	PSO[30]		BBPSO[17]		FIPS[14][15]	
Function	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
f_1	**1.13E−10**	**7.94E−11**	1.58E−06	1.75E−06	1.03E−04	6.95E−05
f_2	5.26E+01	4.20E+01	9.47E+01	8.55E+01	**4.29E+01**	**4.01E+01**
f_3	8.36E+01	1.98E+01	8.28E+01	1.83E+01	**7.78E+01**	**1.63E+01**
f_4	1.90E+01	3.59E+00	**5.08E−01**	**8.70E−01**	6.86E−01	1.04E+00
f_5	**2.10E−03**	**4.16E−03**	5.23E−03	7.96E−03	5.53E−02	8.15E−02
f_6	**3.19E−03**	**4.95E−04**	5.23E−03	3.18E−03	8.01E−02	7.05E−02

and the *Optimum* is the theoretical optimal solution of the function. Experimental results are shown in Table 6.2 and 6.3. In Table 6.2 and 6.3, the experimental results are displayed, where "mean" stands for the mean empirical error of 30 independent runs; "Std. Dev." stands for the standard deviation of the 30 results. And the best results of each team are shown in **boldface**.

The experimental results show that DLS-BBPSO has significant performance on the chosen functions. In the unimodal function group, the DLS-BBPSO wins a first rank and a second rank. Specifically speaking, the DLS-BBPSO gives a mean empirical error 2.14E−41 on f_1. Meanwhile, the champion on f_1, SMA-BBPSO gives a mean empirical error 2.71E−154 and the third rank algorithm PSO gives 4.34E−03. It is reasonable to believe that although the DLS-BBPSO is better than other algorithms in the test, it is still behind the SMA-BBPSO so far. Otherwise, on the f_2, the result of SMA-BBPSO is 2.78% larger than DLS-BBPSO's while the 3rd rank function FIPS gives a 53.76% larger result. Hence it can be conjectured that the search ability of DLS-BBPSO on unimodal functions needs to be improved.

On the other hand, in the multimodal group, the DLS-BBPSO gives excellent results. It wins on all of the four functions in the group and finds the exactly right position on three. Specifically

Table 6.3: Comparisons of empirical error between BBPSOwj, SMA-BBPSO and DLS-BBPSO

	BBPSOwj[20]		SMA-BBPSO[24]		DLS-BBPSO	
Function	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
f_1	4.34E−03	2.37E−02	**2.42E−154**	**2.71E−154**	2.14E−41	1.17E−40
f_2	6.50E+01	4.22E+01	2.87E+01	**1.37E−02**	**2.79E+01**	8.08E−01
f_3	1.11E+01	3.45E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_4	1.54E−01	3.55E−01	2.22E−15	1.81E−15	**1.72E−15**	**1.53E−15**
f_5	3.05E−02	2.84E−02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_6	5.21E−01	1.22E−01	4.80E−12	2.39E−12	**0.00E+00**	**0.00E+00**

speaking, the DLS-BBPSO gives a final result 0 on the f_3, while SMA-BBPSO gives the same answer. But the 3rd rank algorithm, BBPSOwj only gives 1.11E+01. It can be assumed that the DLS-BBPSO and the SMA-BBPSO successfully escape from the local minimum while others do not. On the f_4, the result SMA-BBPSO gives is 29.07% larger than DLS-BBPSO gives, while the BBPSOwj gives a 106.40% larger empirical error. The results of this team prove the DLS-BBPSO has better performance than other chosen algorithms but still has much room to improve. Moreover, DLS-BBPSO finds the right point on both the f_5 and the f_6 while the SMA-BBPSO only succeeds in f_5. It is a good evidence to prove that the dynamic local search system has a very strong ability to escape from the local minimum. The random selecting and dynamic grouping strategy give the swarm a powerful weapon to escape from the local minimum.

6.4 Summary

A bare bones particle swarm optimization with dynamic local search (DLS-BBPSO) is proposed in this chapter. The DLS-BBPSO algorithm inherits the Gaussian distribution from the BBPSO.

Apart from this, a dynamic local grouping system is used to increase the diversity of the swarm. Furthermore, no parameter is needed in DLS-BBPSO. Hence it can be fast applied to different functions.

To verify the performance of the DLS-BBPSO, both unimodal and multimodal benchmark functions are used in the experiment. Meanwhile, the standard PSO and several variants of the BBPSO are running on the same functions to contrast. Finally, the results confirmed the searching ability of the DLS-BBPSO.

Chapter 7

A Bare Bones Particle Swarm Optimization Algorithm with Co-evaluation

In Chapter 6, the DLS-BBPSO divides particles into several local groups to explore the search area. Each local group will focus on its corresponding area. However, since the division will be implemented in every iteration, the connection between different local groups are cut off. The DLS-BBPSO is troubled by this defect when dealing with shifted and rotated functions. To solve this problem, a bare bones particle swarm optimization algorithm with co-evaluation (BBPSO-C) algorithm is proposed. A shadow particle swarm and an original particle swarm work together for the global optimum.

7.1 Overview and Preliminaries

A novel bare bones particle swarm optimization algorithm with co-evaluation (BBPSO-C) is proposed in this chapter. A shadow particle swarm is used to enhance the global search ability of the proposed algorithm. A dynamic grouping method is used to disperse both the shadow particle swarm and the original particle swarm. After the dispersion, an exchanging process will be held between the two swarms. The original swarm will be more concentrated and the shadow swarm will be more scattered. In addition, particles in different swarms are not static. The grouping and exchanging process will be implemented every iteration. With the moving of particles between the two particle swarms, the BBPSO-C gains the ability on both the global search and the local search. To verify the search ability of the proposed method, a set of comperhensive benchmark functions are used in the experiments. Finally, the experimental results confirmed the optimization ability of the BBPSO-C.

7.2 The bare bones particle swarm optimization algorithm with co-evaluation

The bare bones particle swarm optimization algorithm with co-evaluation (BBPSO-C) is proposed in this section. A shadow swarm is used in the BBPSO-C to enhance the search ability. The shadow swarm will assist the original swarm during iterations. Particles are flowing between the two swarms. Generalized speaking, particles which are more dispersed will move to the shadow swarm and the particles which are more central will move to the original swarm. The BBPSO-C is consisted of two major steps: the dynamic classification and the particle exchanging. They will be introduced in the following subsection.

7.2.1 Dynamic classification

A dynamic classification method is used to classify both the original particle swarm and the shadow particle swarm. The classification is implemented inside a particle swarm to keep it individual. Each time two random particles are selected from the swarm. A comparison will hold between the two particles. The particle with better personal best position will be assigned to the main group (MG) while the other one will be assigned to the assistant group (AG). The next position of the particle in the AG will be selection from the Equation 7.1:

$$\begin{aligned} \alpha &= \frac{(pbest(MG) + pbest(AG))}{2} \\ \beta &= |pbest(MG) - pbest(AG)| \\ x(AG) &= N(\alpha, \beta) \end{aligned} \tag{7.1}$$

where the $Pbest = (pbest(1), pbest(2), pbest(n))$ is the personal best position of each particle; the $N(\alpha, \beta)$ is a Gaussian distribution; the x(i) is the next position of the particle. On the other hand, the next position of a particle in the MG is selected from the Equation 7.2:

$$\begin{aligned} \gamma &= \frac{(pbest(MG) + gbest)}{2} \\ \delta &= |pbest(MG) - gbest| \\ x(MG) &= N(\gamma, \delta) \end{aligned} \tag{7.2}$$

where the $Pbest = (pbest(1), pbest(2), pbest(n))$ is the personal best position of each particle; the $N(\gamma, \delta)$ is a Gaussian distribution; the $x(i)$ is the next position of the particle; the $gbest$ is the global best position of the swarm. The pseudo code of this process in the original swarm is shown in Algorithm 5. The same method will be used in the shadow swarm.

Since the selection of the two particles is random, it is uncertain that whether the two particles are very near or very far. If the two particles are close enough, the iterative pattern between the

Algorithm 5 BBPSO-C: grouping, original swarm

Require: Max iteration time, T
Require: Particle swarm $X = x_1, x_2, ...x_n$
Require: Personal best position $Pbest = p_1, p_2, ...p_n$
Require: Global best position $Gbest$

```
t = 1
while t <= T do
    while X ≠ ∅ do
        Randomly select two particles, x_i and x_j, from X
        if f(x_i) < f(x_j) then
            Update x_i with Equation 7.2
            Put x_i into the MG
            Update x_j with Equation 7.1
            Put x_j into the AG
        else
            Update x_i with Equation 7.1
            Put x_i into the AG
            Update x_j with Equation 7.2
            Put x_j into the MG
        end if
    end while
    Update Pbest, Gbest
    t = t + 1
end while
```

two particles should be a local search. Otherwise, if the two particles are wide apart, the iterative pattern should be a global search. The possible iterative pattern of the dynamic classification is shown in Fig. 7.1.

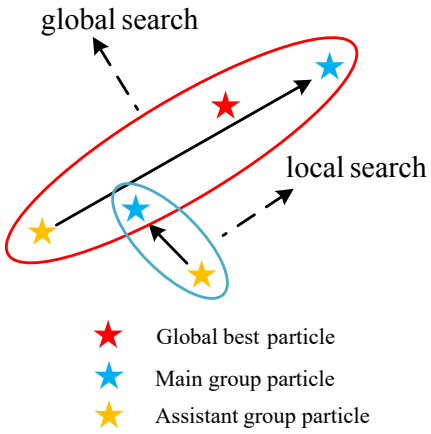

Figure 7.1: The possible iterative pattern of the dynamic classification

7.2.2 Particle exchanging

After the dynamic classification, each swarm has one MG and one AG. The AG of the original swarm and the MG of the shadow swarm will change their positions in the next iteration. The exchanging pattern is shown in Fig. 7.2.

After the exchanging, the original swarm gains stronger ability on locating the local minimum with the two MGs. Meanwhile, the shadow swarm gains stronger ability on escaping from a local minimum with the two AGs.

Also, the dynamic classification will be implemented in the next iteration. The construction of current AG and MG will be destroyed and reconstructed every iteration. The exchange of the particles between the two swarms increases both the local search and the global search ability of the proposed algorithm.

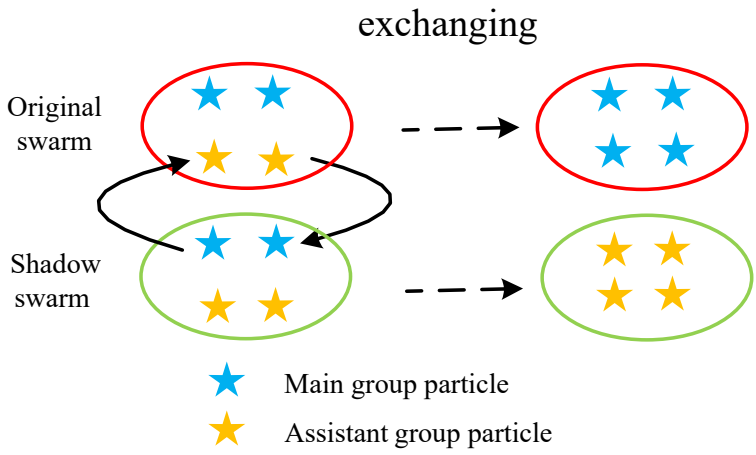

Figure 7.2: The exchanging pattern

7.3 Experiments and discussion

7.3.1 Experimental methods

To verify the performance of the BBPSO-C, both unimodal and multimodal benchmark functions are used in the experiment. The details of the benchmark functions are shown in Table 7.1. All the test functions are in two major groups, the unimodal group ($f_1 - f_4$) and the multimodal group ($f_5 - f_8$). In the unimodal group, the f_1 is a simple unimodal function. The f_2 to f_4 are collected by Suganthan [29]. The f_2 is a shifted unimodal function; the f_3 is a shifted and rotated unimodal function; the f_4 is a scalable and noise in fitness unimodal function. In the multimodal group, the f_5 is a widely used test function, the f_6 to f_8 are collected by Suganthan [29]. The f_6 is a shifted separable function; the f_7 is a shifted and rotated scalable function; the f_8 is a shifted, rotated function which continuous but differentiable on a set of points.

Also, the BBPSO [5], the PBBPSO [35] and the DLS-BBPSO [36] are used as the control group. The dimension of all functions is 30. The population of each algorithm is 30. The iteration time of each test is 1500. Also, to reduce the accidental error, all results are the average value from 30 independent runs. Experimental results keep three significant digits.

7.3.2 Experimental results and discussion

The experimental results are shown in Table 7.2. The empirical error is defined as $|result - TOS|$, where the $result$ is the global best value given by an algorithm after its final iteration, the TOS is the theoretical optimal solution (TOS) of the function. To make a clear competition, we rank the result form the best to the worst. The best result gets one point and the worst one get 4. The average rank result is shown at the bottom of Table 7.2.

In the unimodal group, the BBPSO-C gets two first and two second rank. On f_1, the result of the BBPSO is more than 1.00E+3 times larger than the BBPSO. On f_2, the BBPSO-C gets

Table 7.1: Benchmark Functions

f_1 = Sphere Function

$f_1(x) = \sum_{i=1}^{D} {z_i}^2,$

$z \in (-100, 100)$, global minimum=0

---

f_2 = Shifted Schwefel's Problem 1.2

$f_2(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} z_i)^2 + \mathit{shifted},\ z = x - o$

$x \in (-100, 100)$, $\mathit{shifted} = -450$, global minimum= $-450, o$ is the shifted global optimum

---

f_3 = Shifted Rotated High Conditioned Elliptic Function

$f_3(x) = \sum_{i=1}^{d} (10^6)^{\frac{i-1}{d-1}} z_i^2 + \mathit{shifted},\ z = (x - o) * M,\ M$ is the orthogonal matrix

$x \in (-100, 100)$, $\mathit{shifted} = -450$, global minimum=$-450, o$ is the shifted global optimum

---

f_4 = Shifted Schwefel's Problem 1.2 with Noise in Fitness

$f_4(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} z_i)^2 * (1 + 0.4\,|N(0,1)|) + \mathit{shifted},\ z = x - o$

$x \in (-100, 100)$, $\mathit{shifted} = -450$, global minimum= $-450, o$ is the shifted global optimum

---

f_5 = Ackley Function

$f_5(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$

$x \in (-32.768, 32.768)$, global minimum= 0

---

f_6 = Shifted Rastrigin's Function

$f_6(x) = \sum_{i=1}^{D} ({z_i}^2 - 10\cos(2\pi z_i) + 10) + \mathit{shifted}, z = (x - o)$

$z \in (-5, 5)$, $\mathit{shifted} = -330$, global minimum= -330, o is the shifted global optimum,

---

f_7 = Shifted Rotated Rastrigin's Function

$f_7(x) = \sum_{i=1}^{D} (\frac{{z_i}^2}{4000}) - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + \mathit{shifted}, z = (x - o) * M$

$x \in (-5, 5)$, $\mathit{shifted} = -330, o$ is the shifted global optimum

M: linear transformation matrix, condition number= 2, global minimum= -330

---

f_8 = Shifted Rotated Weierstrass Function

$f_8(x) = \sum_{i=1}^{D} (\sum_{k=0}^{20} [a^k cos(2\pi b^k (z_i + 0.5))]) - D \sum_{k=0}^{20} [a^k cos(2\pi b^k * 0.5)] + \mathit{shifted}$

$x \in (-0.5, 0.5), a = 0.5, b = 3, z = (x - o) * M, o$ is the shifted global optimum

M: linear transformation matrix, condition number= 5, $\mathit{shifted}$= 90, global minimum= 90

Table 7.2: Comparisons of empirical error between BBPSO, PBBPSO, DLS-BBPSO and BBPSO-C

Function	BBPSO[17]	PBBPSO[35]	DLS-BBPSO[36]	BBPSO-C
	Mean	Mean	Mean	Mean
	Std. Dev.	Std. Dev.	Std. Dev.	Std. Dev.
f_1	**2.04E−21**	1.67E−15	8.64E−15	3.14E−18
	3.70E−21	3.67E−15	1.98E−14	5.26E−18
f_2	1.97E+04	1.10E+03	5.45E+03	**5.09E+02**
	1.29E+04	7.08E+02	4.69E+03	2.25E+02
f_3	2.94E+08	1.07E+07	4.47E+07	**7.32E+06**
	1.13E+08	8.91E+06	3.66E+07	3.48E+06
f_4	2.13E+06	**1.11E+04**	1.54E+04	1.32E+04
	9.25E+05	4.12E+03	5.96E+03	7.47E+03
f_5	3.27E−02	1.54E−02	8.86E−02	**2.91E−10**
	7.89E−02	4.74E−02	3.40E−02	3.47E−10
f_6	7.12E+01	5.29E+01	**3.89E+01**	4.57E+01
	2.21E+01	1.61E+01	7.56E+00	1.21E+01
f_7	2.07E+02	1.70E+02	2.15E+02	**1.40E+02**
	1.90E+02	6.21E+01	5.40E+01	6.69E+01
f_8	**3.85E+01**	4.06E+01	4.11E+01	4.04E+01
	4.89E+00	2.06E+00	1.32E+00	1.20E+00
Average Rank	3	2.25	3.25	1.5

the first rank. The result is 53.73% smaller than the result of the PBBPSO. On f_3, the result of the BBPSO-C is 83.62% smaller than the result of the DLS-BBPSO. Om f_4, the result of the BBPSO-C is 18.92% times larger than the first rank, the PBBPSO. In the multimodal group, the BBPSO-C gets two first and two second rank. On f_5, the result of the BBPSO-C is 2.91E−10 while other results are around 2.00E−02. It means the BBPSO-S performances much better than other algorithms. On f_6, the BBPSO-C is the second rank. The result is 17.48% larger than the result of the DLS-BBPSO. On f_7, the BBPSO-C gets the first. The result is 17.65% smaller than the result of the PBBPSO. On f_8, the result of the BBPSO-C is 4.93% larger than the result of the BBPSO. To sum up, the performance of the BBPSO-C is best among all the algorithms. The average rank is 1.5. It means that the proposed co-evolution method can improve the search ability of the original BBPSO on both unimodal and multimodal functions.

7.4 Summary

A bare bones particle swarm optimization algorithm with co-evaluation (BBPSO-C) is proposed in this section. A shadow particle swarm is used in the BBPSO-C to keep the diversity of the original swarm. Particles in each swarm are divided into two groups: the main group and the assistant group. Particles in the main group are more concentrated and particles in the assistant group are more scattered. After that, the assistant group of the original swam and the main group of the shadow swarm will change their positions. The original swarm with two main groups will have a stronger ability in locating the local optimum. The shadow swarm will have a stronger ability on escaping from a local optimum. In the next iteration, the particle grouping and exchanging process will be implemented again.The two swarms work together to find the global optimal position in the search area. To verify the performance of the proposed method, a set of comprehensive benchmark functions are used in the experiments. Finally, the experimental

results confirmed the optimizing ability of the BBPSO-C.

Chapter 8

A Dynamic Reconstruction Bare Bones Particle Swarm Optimization Algorithm

As the advancement of the hardware, its reasonable to increase the iteration time for better optimization results. However, in previous chapters, proposed algorithms are difficult to work with long iterations. Particles will converge in advance and field to find better results. In order to solve this problem, a dynamic reconstruction bare bones particle swarm optimization algorithm (DRBBPSO) is proposed in this chapter. A swarm reconstruction strategy is used to keep the diversity of the swarm. In each iteration, several particles will be selected from the original swarm as the elite particles. After a few times of generation, the original swarm will be replaced by the elite particles.

8.1 Overview and Preliminaries

The bare bones particle swarm optimization algorithm is a useful method for the optimization problems. Each individual particle has been given memories to recorded its personal best position. The best of all personal best positions is recorded as the global best position by the particle swarm. A Gaussian distribution is used to control the behavior of the particles according to the personal and the global best position. However, this iterative pattern weak at multimodal problems. Particles are easy to be trapped in the local minimums. To cross this shortcoming, the dynamic reconstruction bare bones particle swarm optimization algorithm (DRBBPSO) is proposed in this section. The dynamic reconstruction strategy is used to enhance the global search ability of the particle swarm. Numbers of elite particles are selected to reconstruct the particle swarm. To verify the performance of the proposed algorithm, a set of comprehensive benchmark functions are used in the experiments. Also, several swarm-based algorithms including the standard bare bone particle swarm optimization algorithm are used in the control group. The experimental results confirmed the searching ability of the DRBBPSO.

8.2 Proposal of the dynamic reconstruction Method for BBPSO

The performance of the original BBPSO is limited by its iterative pattern. All particle moving to the same global best particle makes the swarm very easy to be trapped by the local minimum. To cross the shortcomings, the dynamic reconstruction bare bones particle swarm optimization algorithm (DRBBPSO) is proposed in this section. The core concept of the DRBBPSO is selecting elite particles from each iteration. After the number of selected particle equals the number of the original particle swam, the former swam will be replaced by the elite particle

swarm. Hence, the DRBBPSO is combined by the two major steps: the elite selection and the swarm reconstruction. The introduction of each step is in the subsection.

8.2.1 The elite selection

A dynamic elite particle selection (DEPS) method is used in the DRBBPSO. The DEPS works by observing the iterative process of the particle swarm and copying the elite particles to an additional container. Different with some regular selecting method, no ranking is needed during the selecting process. Each time two random particles are selected from the swarm. We consider the two particles formed a local group. A comparison will be hold between the two particles. The one with better personal best position will be selected as an elite particle. The other one will be assigned as a retinue of the elite. The elite and its retinue have different iterative rules in further iteration. Specifically, the elite particles will keep learning from the global best particle. The next position of an elite particle is selected by the Equation 8.1:

$$\begin{aligned} \alpha &= \frac{(pbest(elite) + gbest)}{2} \\ \beta &= |pbest(elite) - gbest)| \\ elite_{new} &= N(\alpha, \beta) \end{aligned} \tag{8.1}$$

where $pbest = (pbest(1), pbest(2), ..., pbest(n))$ is a matrix used for recording the best position each particle has ever reached; g_{best} is the best position that all particles have ever reached; $N(\alpha, \beta)$ is a Gaussian distribution with a mean value α and standard deviation β.

Conversely, the retinue particles aim at searching around their corresponding elites. The next position of a retinue particle is randomly selected from Equation 8.2:

$$retinue_(new) = N(\frac{p(r) + p(e)}{2}, |p(r) - p(e)|) \tag{8.2}$$

where $p(r)$ is the personal best position of a retinue particle, $p(e)$ is the personal best position of the corresponding elite of the retinue particle.

In this iterative pattern, the elite particles exchange information with the global best particle and try to get over it. On the other hand, the retinue particles explore the area around their elite. This strategy remains the diversity of the swarm. A low rank particle still has the chance to be an elite and vice versa. Moreover, the composition of each local group is dynamic. It is possible that the selected particles are very close and this search should be a local search. Also the two selected particles maybe very far and the search should be a global search. Since the selection is random, the construction of each local group is dynamic. One particle may be selected with different particles in different iteration. This feature increases the diversity of the particle swarm and gives the swarm a strong ability to anti the local minimum. The construction of particle swarm are shown in Fig 8.1.

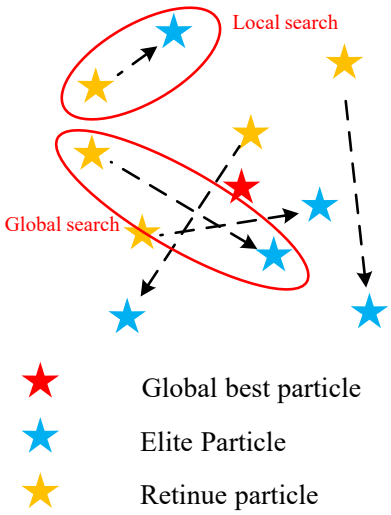

Figure 8.1: Swarm construction of the elite selection

After one round of iteration, the copy of elite particles will be saved to the container and the iteration will keep going. This selection pattern remains the original features of the particle swarm without interference. The selection will continue until the container is full. So if a retinue

wins the comparison in the next round, it will be selected as an elite by the DEPS.

8.2.2 The swarm reconstruction

The container is set to have a same size with the original particle swarm of the DRBBPSO. After two round of the elite selection, the container is full of elite particles. Then we will replace the original particle swarm by the elite particle swarm. It can be seen that one particle with an unsatisfactory personal best position also has an opportunity to be chosen as an elite. This situation helps keeping the diversity of the swarm. Moreover, since some particles have the opportunity to be selected as the elite twice, the second round of elite particles will multiple a distortion coefficient d. The d is a random number between 0.9 and 1.1.

After the reconstruction of the particle swarm, we will empty the container. The swarm will iterative with the same rule and the DEPS will keep selecting container gets full again. To show the working flow of the DRBBPSO, the pseudo code is given by Algorithm 6:

Algorithm 6 DRBBPSO

Require: Max iteration time, T
Require: Dimension of the problem, D
Require: Fitness function, f
Require: Searching Range, R
Require: Particle swarm, $X = x_1, x_2, ...x_n$
Require: The particle number of the particle swarm, N
Require: Personal best position, $Pbest = p_1, p_2, ...p_n$
Require: Global best position, $Gbest$
Require: The elite particle swarm, ES
Require: The particle number of the elite particle swarm, EN
Require: Distortion coefficient d
Ensure: all particles are in R

$t = 0, EN = 0$
while $t < T$ **do**
  while $EN < N$ **do**
    while $X \neq \varnothing$ **do**
      From X, randomly select and remove two particles, x_i and x_j
      if $p(x_i) < p(x_j)$ **then**
        Update x_i with Equation 8.1
        Update x_j with Equation 8.2
        Copy x_i to the ES
        EN=$EN + 1$
      else
        Update x_i with Equation 8.2
        Update x_j with Equation 8.1
        Copy x_j to the ES
        EN=$EN + 1$
      end if
    end while
    while $X \neq \varnothing$ **do**
      From X, randomly select and remove two particles, x_i and x_j
      if $p(x_i) < p(x_j)$ **then**
        Update x_i with Equation 8.1
        Update x_j with Equation 8.2
        Copy $d * x_i$ to the ES
        EN=$EN + 1$
      else
        Update x_i with Equation 8.2
        Update x_j with Equation 8.1
        Copy $d * x_j$ to the ES
        EN=$EN + 1$
      end if
    end while
  end while
  $X = ES$, ES = empty, $EN = 0$
  Update $Pbest$
  Update $Gbest$
  $t = t + 1$
end while

8.3 Experiments

8.3.1 Benchmark functions

To verify the performance of the DRBBPSO, a set of comprehensive benchmark functions are selected in the experiments [16][24]. Both unimodal and multimodal functions are chosen. All of the 5 functions are minimum value problem; the details of all functions are shown in the Table 8.1.

To minimize the impact of accidental factors, the empirical error is calculated from 30 independent runs. The empirical error is defined as $\mid gbest - Minimun \mid$. Where $gbest$ is the final global best value given by an algorithm, and $Minimun$ is the theoretical optimal solution of the test function. Also, the dimension of each function is set as 30. Moreover, the comprehensive learning PSO (CLPSO) [16], the orthogonal learning PSO (OLPSO) [33], the BBPSO [17] and the DLS-BBPSO [36] are used in the control group. According to Li [34], the size of population is set to 40 for CLPSO and OLPSO, 30 for the rest bare bones PSO algorithms. Experimental results keep three significant digits.

8.3.2 Results of the experiments with different iteration times

To verify the convergence speed of the DRBBPSO, the standard BBPSO is used in the compare to the proposed algorithm. The two algorithms run with the five benchmark functions, the results of 100 times iteration, 200 times iteration and 500 times iteration are recoded in the Table 8.3.2 and 8.3. The results are the mean values of 30 independent runs. The dimension of the problem is 30, population of this test is 30. The best result of each comparison is shown in **Bold**.

It can be seen that the result of the DRBBPSO is 13.17% larger than the result of the BBPSO after 100 times iteration in f_1. But the DRBBPSO gives better result at 200 iterations and keep leading when the iteration time reaches 500. On f_2, the results of the DRBBPSO is better than

Table 8.1: Benchmark Functions

Function
f_1 = Shifted Sphere Function $f_1(x) = \sum_{i=1}^{D} {z_i}^2 + \mathit{shifted},\ z = x - o$ $x \in (-100, 100), o$ is the shifted global optimum, *shifted*$= -450$, global optimum$= -450$
f_2 =Shifted Schwefel's Function 1.2 $f_2(x) = \sum_{i=1}^{d} (\sum_{j=1}^{i} z_i^2)^2 + \mathit{shifted},\ z = x - o$ $x \in (-100, 100), o$ is the shifted global optimum, *shifted*$= -450$, global minimum$= -450$
f_3 = Shifted Rotated High Conditioned Elliptic Function $f_3(x) = \sum_{i=1}^{d} (10^6)^{\frac{i-1}{d-1}} z_i^2 + \mathit{shifted},\ z = (x - o) * M$ $x \in (-100, 100), o$ is the shifted global optimum, *shifted*$= -450$, global minimum$= -450$ M: orthogonal matrix
f_4 = Shifted Rotated Griewank's Function without Bounds $f_4(x) = \sum_{i=1}^{d} \frac{z_i^2}{4000} - \prod_{i}^{d} \cos(\frac{z_i}{\sqrt{i}}) + 1 + \mathit{shifted},\ z = (x - o) * M$ $x \in (0, 600), o$ is the shifted global optimum, *shifted* $= -180$, global minimum$= -180$ M': linear transformation matrix, condition number=3, $M = M'(1 + 0.3\,\lvert N(0,1)\rvert)$
f_5 = Shifted Rotated Ackley's Function with Global Optimum on Bounds $f_5(x) = -20\exp(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} z_i^2} - \exp(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi z_i)) + 20 + e + \mathit{shifted},$ $x \in (-32, 32), z = (x - o) * M, o$ is the shifted global optimum, *shifted*$= -140$, global minimum$= -140$ M: linear transformation matrix, condition number$= 100$

the result of the BBPSO after 100 iterations, 200 iterations and 500 iterations. On f_3, the result of the BBPSO is more than 4 time larger than the result of the DRBBPSO after 100 iterations, more than two time after 200 iterations and 80.84% larger when the iteration time reaches 500. We can consider that the DRBBPSO has better performance on the f_2 and the f_3 than the BBPSO.

On f_4, the empirical error of the BPPSO is more than 5 times larger than the DRBBPSO after 100 iterations.However, the disadvantage is reduced when the iteration time reaches 200. Finally the BBPSO gives better result than the DRBBPSO when the iteration time reaches 500. The advantage is 3.18%. It can be speculated that at the beginning, the DRBBPSO has stronger search ability than the BBPSO at the first 100 iterations. Then the searching of the DRBBPSO is slow down. Finally it was exceeded by the BBPSO.

On f_5, the empirical error of the BPPSO is 0.47% larger than the DRBBPSO after 100 iterations. After that the two algorithms give the same results on 200 iterations and 500 iterations. Hence we consider that the DRBBPSO and the BBPSO are trapped in some local minimum after 100 iteration. Since the two algorithms give the same results, it is reasonable that the bare bones strategy can not get over of this problem.

Since the DRBBPSO gives better results than the BBPSO on 3 functions, we can say that the DRBBPSO has stronger ability on fast search than the BBPSO. However, we can also find out that on f_5, the DRBBPSO may losing diversity too fast and unable to escape from the local minimum.

Table 8.2: Comparisons of empirical error between the BBPSO and the DRBBPSO, on different iteration times

	BBPSO		
Function	100 Iterations	200 Iterations	500 Iterations
f_1	**2.43E+03**	5.56E+01	4.97E−04
f_2	4.79E+04	3.42E+04	9.67E+03
f_3	2.09E+08	8.42E+07	3.02E+07
f_4	5.59E+02	7.08E+01	**1.52E+00**
f_5	2.12E+01	**2.11E+01**	**2.11E+01**

Table 8.3: Comparisons of empirical error between the BBPSO and the DRBBPSO, on different iteration times

	DRBBPSO		
Function	100 Iterations	200 Iterations	500 Iterations
f_1	2.75E+01	**4.33E+00**	**6.20E−03**
f_2	**1.78E+04**	**8.36E+03**	**2.54E+03**
f_3	**4.55E+07**	**3.95E+07**	**1.61E+07**
f_4	**1.07E+02**	**1.92E+0**1	1.57E+00
f_5	**2.11E+01**	**2.11E+01**	**2.11E+01**

Table 8.4: Comparisons of empirical error between the OLPSO and the CLPSO on 30 dimensional test

	OLPSO		CLPSO	
Function	Mean	Std. Dev.	Mean	Std. Dev.
f_1	**0.00E+00**	**0.00E+00**	5.68E−14	**0.00E+00**
f_2	**1.50E+01**	**1.23E+01**	8.79E+02	1.79E+02
f_3	1.46E+07	5.33E+06	1.67E+07	**4.66E+06**
f_4	4.70E+03	1.61E−12	4.70E+03	**6.78E−12**
f_5	**2.09E+01**	6.90E−02	2.09E+01	5.46E−02

8.3.3 Results of the 30 dimensional test

To make a comprehensive assessment of the search ability of the DRBBPSO, the OLPSO, the CLPSO and the DLS-BBPSO are running on the five benchmark functions as the control group. The iteration time of all the functions is 300,000 in this round. The results are shown in the Table 8.4 and 8.5. In the Table 8.4 and 8.5, the *mean* means the mean value of the 30 independent empirical errors, the *Std.Dev.* means the standard deviation of the 30 results. Experimental results keep three significant digits. The best result are shown in **Bold**.

On f_1, every algorithm reaches the global optimum except the CLPSO. On f_2, the OLPSO gives the best result. The result of the DRBBPSO is about 39 times larger than the best one. On f_3, the DRBBPSO gives the best result. The result of the OLPSO is 57.67% larger than the best one. On f_4, the DRBBPSO gives the best result. The empirical error is small than the 1% compared with the OLPSO and the CLPSO. On f_5 the results of all algorithms nearly have no difference. The OLPSO and the CLPSO have the same results and the DLS-BBPSO and the DRBBPSO have the same result. The PSO algorithms have a 0.4% advantage.

To sum up, the DRBBPSO gets 3 first rang and 2 second rank in the experiment. It is

Table 8.5: Comparisons of empirical error between the DLS-BBPSO and the DRBBPSO on 30 dimensional test

	DLS-BBPSO		DRBBPSO	
Function	Mean	Std. Dev.	Mean	Std. Dev.
f_1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_2	5.22E+03	3.41E+03	5.88E+02	8.07E+02
f_3	6.34E+07	6.89E+07	**9.26E+06**	6.78E+06
f_4	6.54E+01	3.97E+01	**4.73E−01**	2.68E−01
f_5	2.10E+01	5.06E−02	2.10E+01	**4.91E−02**

reasonable to believe that the proposed method have better performance than the algorithms in the control group. However, the result of the DRBBPSO on f_5 is unsatisfactory. The particle swarm is trapped since 100 iterations. And the result moves from 2.11E+01 to 2.10E+01 after 300,000 iterations. In addition, the BBPSO and the DLS-BBPSO are also trapped by the f_5, it is reasonable to conjecture that the bare bones strategy makes the swarm very easy to be trapped. Hence, trying to break the limit of the bare bones strategy, improving the global search ability for the multimodal functions and balancing the local search and the global search should be a main area of the future work.

8.4 Summary

A dynamic reconstruction bare bones particle swarm optimization (DRBBPSO) algorithm, which can solve both unimodal and multimodal problems, is proposed in this paper. A dynamic elite particle selection (DEPS) method is used in the DRBBPSO. The DEPS is used for elite particle selecting during the iterative process. It just copy the elite particles and do no interference until the number of the elite particles equals the number of particles in the original particle swarm.

Then the algorithm will replace the original particle swarm by the elite particle swarm. This strategy enhance the local search ability by the reconstruction of the particle swarm. Also, the DEPS ensures that the particles with unsatisfactory personal best position also has a chance to be selected as an elite. It keeps the diversity of the elite particle swarm. To verify the search ability of the DRBBPSO, several well-known benchmark functions are used in the experiments. Also, to enhance the persuasion of the experiments, several swarm-based evolutionarily algorithms are also evaluated on the same functions as the control group. Finally, the experimental results and the statistical analysis confirmed the search ability of DRBBPSO in the selected functions.

Chapter 9

A Fission-Fusion Hybrid Bare Bones Particle Swarm Optimization Algorithm

As the society progresses, optimization problems become more complex and diverse. It is necessary for optimization algorithms to obtain the ability to solve different types of optimization problems. Also, in some extreme cases, the algorithms may not have a chance to do the training or parameter adjustment. In response to these needs, a fission-fusion hybrid bare bones particle swarm optimization (FHBBPSO) algorithm is proposed in this chapter. The FHBBPSO is a parameter-free algorithm which is able to solve multiple types of problems. A fission method and a fusion method work together to find the global best solution of the single-objective optimization problems.

9.1 Overview and Preliminaries

The particle swarm optimization [5] algorithm is a powerful method for single-objective problems. Particles simulate the team behavior of fish and birds. The concept of memory and velocity are introduced to particles. A particle can record its personal best position and use it in the future generation. Plenty of researchers engaged in this field since its introduction. The PSO has become one of the most popular optimization techniques and has been successfully applied in different areas like image processing [37], motion system [38], supply chains [39] and so on.

In the original PSO algorithm, particles are moving in the search space. The moving distance and angle are controlled by a velocity item. The velocity is effected by the current position of this particle, the historical best position of this particle, the current position of the global best particle and the parameters. Under this pattern, previous works are needed to adjust the parameters. Also, it is maybe necessary to chance the parameters when the PSO is dealing with complicated problems. In 2003, the bare bones particle swarm (BBPSO) is proposed by Kennedy [17]. The velocity item is canceled from the original PSO algorithm. The next position of a particle is selected by a Gaussian distribution. Particles are moving to the global best particle in each iteration. This evolutionary pattern insures the fast convergence and no parameters are needed during the iteration. The BBPSO and its variants are widely used in different areas like path planning [40], clustering [41] and so on.

Differential evolution (DE), proposed by Storn and Price [42], is another powerful population-based evolutionary algorithm for global numerical optimization. It employs three main operators: the mutation, the crossover and the selection. The mutation gives each individual a chance to get a sudden change. The crossover ensures that information can be shared between different individuals. The selection makes the better individuals have more chance to spread their information in further generation. The DE and its variants are widely used in different areas like

image processing [43] , clustering [44] and so on.

However, the population based algorithms are still troubled in balancing the exploration and exploitation. The population may easily be trapped in a local minimum if it obtains a strong local search ability. On the other hand, a population may be difficult to find a precise local best point if it own a strong breadth search ability. Trying to solve this problem, the fission-fusion hybrid bare bones particle swarm optimization algorithm is proposed in this section. The rest of this section is organized as follows: in section 9.2, the processes and advantages of the FHBBPSO are introduced; in section 9.3, the details of the verifiable experiments and discussion are displayed; in section 9.4, the summary of this chapter is presented.

9.2 Proposal of the FHBBPSO

9.2.1 Motivation and definition

There plenty of existing evolutionary methods for single-objective problems. But most of them improve their ability by adding supernumerary calculation like feather selection, stage control, multi-method combination and so on. These methods usually need a lot of previous work to determine their parameters. Also, additional methods may slow down the algorithms while dealing with complex functions. To solve these problems, a fission-fusion bare bones particle swarm optimizer (FHBBPSO) is proposed in this section. The FHBBPSO is designed to evolve without parameters. Two different evolutionary methods are proposed to control the behavior of the particles. During the iteration process, particle are divided into several local groups and search their corresponding ares. A local group contains a leader and several teammates. The details of how these methods work will be presented in later subsections.

9.2.2 The structure of local groups

Before iteration, an initialization process will be done. As a parameter-free method, no previous knowledge or human intervention is needed. All of the particles will be randomly spread in the search space. Then, the personal best position of each particle and the global best position of the particle swarm can be calculated. With the personal best and the global best position, the proposed method is ready to work.

In the FHBBPSO, the basic arithmetic unit is the local group. The working process of the FHBBPSO is controlled by the number of local groups. Every particle will only belong to one local group. In a single local group, particles are playing two characters: the leader and the teammate. A leader is the best particle in the local group. The leader is designed to get information from the global best particle and do the global search. The next position of a leader is selected from the Equation 9.1:

$$\begin{aligned} \alpha &= \frac{(pbest(leader) + gbest)}{2} \\ \beta &= |pbest(leader) - gbest)| \\ leader_{new} &= N(\alpha, \beta) \end{aligned} \tag{9.1}$$

where the $pbest(leader)$ is the personal best position of the leader, the $gbest$ is the global best position of the swarm, the $N(\alpha, \beta)$ is a Gaussian distribution with a mean value α and a standard deviation β.

On the other side, the teammates are the rest particles in the local group except the leader. The teammates are designed to do the local search inside the local group. Each teammate will get information from its corresponding leader. The next position of a teammate is selected from

the Equation 9.2:

$$\begin{aligned} \gamma &= \frac{(pbest(teammate) + pbest(leader))}{2} \\ \delta &= |pbest(teammate) - pbest(leader))| \\ teammate_{new} &= N(\gamma, \delta) \end{aligned} \tag{9.2}$$

where the $pbest(teammate)$ is the personal best position of the teammate, the $pbest(leader)$ is the personal best position of the corresponding leader of the teammate, the $N(\gamma, \delta)$ is a Gaussian distribution with a mean value γ and a standard deviation δ.

In the FHBBPSO, the character of a single particle is not static. It is possible that a leader particle becomes a teammate in the future iteration and vise versa.

9.2.3 The fission

An exhaustive unsupervised dynamic division strategy (DDS) is used in the fission process. After the initialization, we consider that all of the particles are in a same local group and the leader of this group is the global best particle. For convenience, we name this swarm as the original swarm. The DDS aims at separating the original swarm to several local groups. It works by the random selection. First, a random particle will be taken out from the original swarm as the first "current" leader. Then, another random particle will be taken out to compare with the "current" leader. If the selected particle is worse than the "current" leader, it will be a teammate of the "current" leader. Otherwise, it will replace the previous one and become the new "current" leader. After every particle is removed from the original particle swarm, we get a new swarm composed by several local groups. It can be found that in the fission process each particle will be only selected once. It means that the time complexity of the fission method is $o(n)$. For a better explanation, an example with 10 particles is shown in Fig. 9.1. Particles are marked from the best to the worst using 1, 2, until 10. It can be seen that one selection order will create one

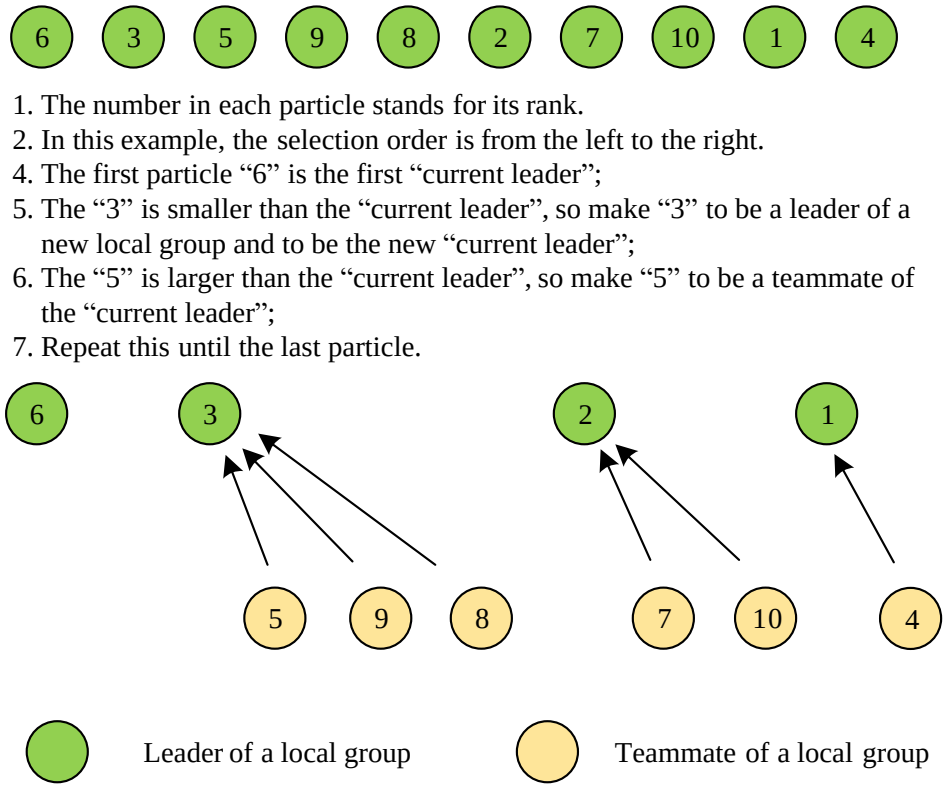

Figure 9.1: An example for the fission process. A teammate particle uses an arrow to point to its leader.

structure of the local groups. For more rigor, the pseudo code of the fission process is shown in Algorithm 7.

The intention of the fission part is dividing the particle swarm into several local groups. Each local group will only focus on its corresponding region to find the local best point. In normal algorithms, the number of local groups should be a very important parameter. But in the FBBPSO, the number of local groups is affected by the selection order. In theory, there are $n!$ different orders by the random selection, where n is the number of particles. In some extreme case, for instance, if the global best particle was selected as the first “current” leader, all particles will be placed in one local group. This situation is the same as the standard BBPSO. Hence, we can claim that the results of the fission process contain multiple possibilities including the

Algorithm 7 Fission: dynamic division

Require: Fitness function, F
Require: Search Space, R
Require: Dimension of the function, D
Require: Particle swarm, $X = (x_1, x_2, ...x_n)$
Require: Personal best position, $Pbest = (p_1, p_2, ..., p_n)$
Require: Global best position, $Gbest$
Require: Number of local groups, TN
Require: Matrix of local groups, $Localgroup$
Require: List of the leader of each local group, $LeaderList$

```
Randomly take out a particle x_i from the X
TN=1
Make x_i be a leader of the Localgroup(TN)
Select a new position for x_i according Equation 9.1
LeaderList(TN) = x_i
while X ≠ ∅ do
    Randomly take out a particle x_j from the X
    if Pbest(j) <= Pbest(LeaderList(TN)) then
        TN=TN+1
        Make x_j be a leader of Localgroup(TN)
        LeaderList(TN) = x_j
        Select a new position for x_j according Equation 9.1
    else
        Make x_j be a teammate of Localgroup(TN)
        Select a new position for x_j according Equation 9.2
    end if
end while
Update Pbest
Update Gbest
```

BBPSO. Also, the selection order could be from the worst particle to the best particle. In this situation, the number of local groups will be n. So the number of local groups will be in the range from 1 to n. These rich and varied forms of the local groups greatly increase the diversity of the particle swarm. More importantly, no parameters and human intervention are needed in the grouping process, which means the DDS can be easily applied to different problems.

On the mathematical side, the fission method contains every possibility of the partition of local groups. The partition of local groups means the number of local groups and the number of particles in each local group. To prove this, we need to use a string to represent the structure of the local groups. Then, we will prove that each order of the string can convert to a kind of local structure and each local structure can convert to a string.

For easy description, we need to rank all the particle and mark them from the best to the worst using 1, 2, until n, n is the number of the particles. Then we need to sort all of the local groups from the worst to the best. After that, we will record the leader of the worst group, the teammates of the worst group, the leader of the second worst local group, the teammates of the second worst local group, until the best local group. By doing this we can convert any combination of local groups to an only array. One of a possible example is shown in Fig. 9.2.

Conversely, if we have an array that contains the combination of the numbers from 1 to n, we can convert it to a combination of local groups. To do this, we need to set the first number as the leader of the first local group. Then, we will use the second number to compare with the previous leader. If it is larger than the previous leader, it will be a teammate of the previous leader. Otherwise, it will be a leader of a new local group. After that, we will select the coming numbers in the array to do the same thing until the last one. One of a possible example is shown in Fig. 9.1. In that situation, the string "6, 3, 5, 9, 8, 2, 7, 10, 1, 4" is converted to an only structure of local groups. By using this method, we can convert any arrays to an only combination of local groups. So every combination of local groups can be converted to arrays

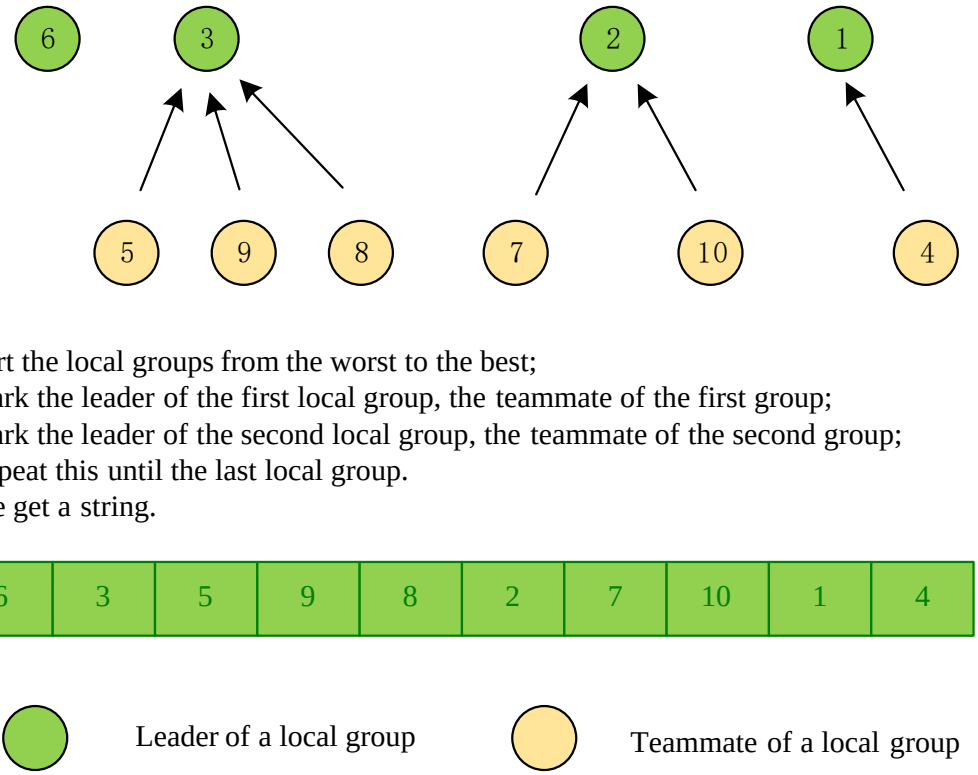

Figure 9.2: Convert a local structure to a string. A teammate particle uses an arrow to point to its leader.

and every arrays can convert to the combination of local groups. Moreover, from the previous introduction, it can be found that the selection in the fission process is all random. So any selection order can occur, which means that fission method contains every possibility of the partition of local groups.

9.2.4 The fusion

The fusion strategy is inspired by the competition and cooperation of chimpanzees in the forest. The resource in the forest is limited and uneven. Different groups of chimpanzees have to compete with each other. Strong groups will occupy the resource-rich areas and getting stronger while the weak groups gradually become marginalized. From the perspective of biological evolution, it is reasonable that stronger individuals get more resource. Applying this theory to the fusion part, the best local group increase its search ability by devouring the worst local group in one single iteration. The middle-rank local groups will keep their structure in the iteration and keep searching their corresponding areas. They still have the chance to be the best if they can find a new global best position. Using this method, the best group will keep engulfing the worst group until there is only one local group left. Every particle in the worst group will be moved to the best group while other groups will keep their structure. The pseudo code of the fusion process is shown in Algorithm 8.

The propose of the fusion part is withdrawing the particle in the worst local group and putting them into the best local group. It is obvious that in a population-based algorithm, the population and search capabilities are equivalent. Hence in the fusion process, the best local group increase its search ability by merging the worst local group. At the same time, other local groups still have a chance to sample their corresponding areas. If one of the local groups find a better point than the current global best position, it can be the best local group in the next iteration. Then it gains the right to engulf the worst local group.

Algorithm 8 Fusion: local team elimination

Require: Fitness function, F
Require: Search Space, R
Require: Dimension of the function, D
Require: Particle swarm $X = x_1, x_2, ...x_n$
Require: Personal best position $Pbest = p_1, p_2, ...p_n$
Require: Global best position $Gbest$
Require: The number of local teams TN
if $TN == 1$ **then**
  do the Fission
end if
if $TN > 1$ **then**
  Find the *bestgroup* and the *worstgroup* by ranking
  Merge the *bestgroup* and the *worstgroup*
  for Each particle in each local group **do**
    if the particle x_i is a leader **then**
      Select a new position for x_i according Equation 9.1
    end if
    if the particle x_i is a teammate **then**
      Select a new position for x_i according Equation 9.2
    end if
    TN=TN-1
  end for
end if
Update *Pbest*
Update *Gbest*

On the mathematical side, the merging process will not increase the calculation time. The next position of a particle is still selected by the Gaussian distribution. So the time complexity of the fusion process is $o(n)$.

9.2.5 Hybrid of the fission and the fusion

In this section, the complete process of the FHBBPSO is presented. After the initialization, the number of the local group is one. Then the fission part is on working. One point has to be mentioned here, it is possible that the number of local groups is still one after the fission if the global best particle was selected as the leader of the first local group. So a judgment is set, if the number of local groups equals one, the algorithm will do the fission. If the number of local groups is more than one, the algorithm will do the fusion. The pseudo code of the FHBBPSO is shown in Algorithm 9.

Algorithm 9 The Fission-Fusion Hybrid Bare Bones Particle Swarm Optimization Algorithm

Require: Max iteration time, T
Require: Fitness function, F
Require: Search Space, R
Require: Dimension of the function, D
Require: Particle swarm $X = x_1, x_2, ...x_n$
Require: Personal best position $Pbest = p_1, p_2, ...p_n$
Require: Global best position $Gbest$
Require: The number of local teams $Teamnumber$

```
1: t = 1
2: TN = 1
3: while t <= T do
4:    if TN > 1 then
5:       Do the Fusion
6:    end if
7:    if TN == 1 then
8:       Do the Fission
9:    end if
10:   t = t + 1
11: end while
```

Here, we discuss the complexity of the FHBBPSO. In the fission part, each particle will calculate one time, so the time complexity is $o(n)$. In the fusion part, the main action is merging

the groups. No calculation is added while moving a particle from one local group to another. So the time complexity of the fusion process is still $o(n)$. In the hybrid part, only one of the fusion process or the fission process will be held in one single iteration. So the time complexity of the FHBBPSO is $o(n)$. Also, it can be seen that during the iteration process, no parameter is used. The construct of the local groups is depending on the test functions. This means the FHBBPSO can be easily applied to different problems without previous knowledge. So it is reasonable to believe that the FHBBPSO is a fast calculating and fast applying algorithm.

9.3 Experiments

9.3.1 Experimental methods

To verify the optimization ability of the FHBBPSO, The CEC 2014 benchmark functions [45] are used in the experiments. The functions are divided into four groups:

1) unimodal functions($f_{01} - f_{03}$);

2) simple multimodal functions ($f_4 - f_{16}$);

3) hybrid functions ($f_{17} - f_{22}$);

4) composition functions ($f_{23} - f_{30}$).

More details of the test functions can be found in Table 9.1 and 9.2.

9.3.2 Experimental results and discussion

The experimental results and discussion are displayed in this subsection. The ETI-DE [28], DE [27] [28], DLS-BBPSO [36] are selected to compare with the FHBBPSO. The population of the ETI-DE, the DE and the DLS-BBPSO is set to 100, 100, 30. To make a fair competition, the population size of the FHBBPSO is set to 100. More details can be found in the original references. In this part, the maximum number of function evaluations (MAXFES) is

Table 9.1: Experimental functions, the CEC 2014 benchmark functions, the search range for each function is (-100,100).

Types	Function	Minimum
Unimodal Functions	f_1 = Rotaten High Conditioned Elliptic Function	100
	f_2 = Rotated Bent Cigar Function	200
	f_3 = Rotated Discus Function	300
Simple Multimodal Functions	f_4 = Shifted and Rotated Rosenbrock's Function	400
	f_5 = Shifted and Rotated ACKLEY's Function	500
	f_6 = Shifted and Rotated Weierstrass's Function	600
	f_7 = Shifted and Rotated Griewank's Function	700
	f_8 = Shifted Rastrigin's Function	800
	f_9 = Shifted and Rotated Rastrigin's Function	900
	f_{10} = Shifted Schwefel's Function	1000
	f_{11} = Shifted and Rotated Schwefel's Function	1100
	f_{12} = Shifted and Rotated Katsure Function	1200
	f_{13} = Shifted and Rotated HappyCat Function	1300
	f_{14} = Shifted and Rotated HGBat Function	1400
	f_{15} = Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	1500
	f_{16} = Shifted and Rotated Expanded Scaffer's F6 Function	1600

Table 9.2: Experimental functions, the CEC 2014 benchmark functions, the search range for each function is (-100,100).

Types	Function	Minimum
	f_{17} = Hybrid Function 1 (N=3)	1700
	f_{18} = Hybrid Function 2 (N=3)	1800
Hybrid	f_{19} = Hybrid Function 3 (N=4)	1900
Functions	f_{20} = Hybrid Function 4 (N=4)	2000
	f_{21} = Hybrid Function 5 (N=5)	2100
	f_{22} = Hybrid Function 6 (N=5)	2200
	f_{23} = Composition Function 1 (N=5)	2300
	f_{24} = Composition Function 2 (N=3)	2400
	f_{25} = Composition Function 3 (N=3)	2500
Composition	f_{26} = Composition Function 4 (N=5)	2600
Functions	f_{27} = Composition Function 5 (N=5)	2700
	f_{28} = Composition Function 6 (N=5)	2800
	f_{29} = Composition Function 7 (N=3)	2900
	f_{30} = Composition Function 8 (N=3)	3000

Table 9.3: Experimental algorithms, including parameters and references, the dimension is 50, the MAXFES equals *dimension**1.00E+5.

Algorithm	Population	Parameters	Citation
ETI-DE	100	ETI-DE/rand/1/bin	[28]
DE/rand/1/bin	100	DE/rand/1/bin	[27][28]
DLS-BBPSO	30	None	[36]
FHBBPSO	100	None	

set to *dimension**10000. To reduce accidental errors, the final results are calculated from 51 independent runs. The mean value of the 51 results is recorded as $f(final)$. The *empirical error* is defined as $|f(x*) - f(final)|$, where $x*$ is the theoretical optimal position of each function. The details of the test algorithm are shown in Table 9.3. The experimental results are shown in Table 9.4, 9.5 and 9.6. To make a clear comparison, a rank competition also proposed in the experiments. In each test function, the best result will get 1 point and the worst one will get 4 points. The rank results of each function are also shown in Table 9.4, 9.5 and 9.6. The statistics results of each group are shown in Table 9.7. Experimental results keep four significant digits.

In the Group 1, the FHBBPSO gets 1 point at f_1 and f_3, 3 point at f_2. In the f_1, the result of the proposed method is 45.21% larger than the result of the ETI-DE. In the f_2, the result of the FHBBPSO is more than ten times larger than the result of the DE. In the f_3 the results of the two BBPSO-based algorithm is about 10E$-$14 level while the results of the two DE-based algorithms are around 10E$-$01.

In the Group 2, the FHBBPSO gets 1 point in f_5, f_{10}, f_{14}, f_{16},2 points in $f_{4,8,9}$, f_{11-13}, f_{15}, 3 points in $f_{6,7}$. In the f_4, the two BBPSO-based algorithms occupy the first and the second rank. The result of the FHBBPSO is 173.70% larger than the result of the DLS-BBPSO. In the f_5 the FHBBPSO has a 8.81% lead of the ETI-DE. In the f_6 and f_7, the two DE-based algorithms occupied the first and the second rank. In the f_8, the ETI-DE gets one point while

the FHBBPSO gets two. The result of the FHBBPSO is 59.64% larger than the result of the ETI-DE. In the f_9, the same situation of the f_8 happens. The result of the FHBBPSO is 276.47% times larger than the result of the ETI-DE. In the f_{10}, the FHBBPSO gets the first rank and the result is 41.63% smaller than the one of the ETI-DE. In the f_{11}, f_{12}, f_{13}, f_{15}, the ETI-DE gets the first rank and the FHBBPSO gets the second. The lead of the ETI-DE is 2.73%, 31.85%, 7.60% and 54.89%. In the f_{14} and f_{16}, the FHBBPSO gets the first rank and the ETI-DE gets the second. The lead of the FHBBPSO is 40.87% and 3.17%.

In the Group 3, the FHBBPSO gets 1 point in f_{22}, 2 points in f_{20}, 3 points in $f_{18,19,21}$, 4 points in f_{17}. In the f_{17}, the two DE-based algorithms leads, The result of the FHBBPSO is 2.26% larger than the result of the DLS-BBPSO. From the f_{18} to the f_{21}, the ETI-DE gets the first rank in this four functions. The results of the FHBBPSO is 869.45%, 149.21%, 141.76%, 1012.12% larger than the results of the ETI-DE. In the f_{22}, the FHBBPSO gets the first rank and the DE gets the second. The lead of the FHBBPSO is 64.33%.

In the Group 4, the FHBBPSO gets 1 point in f_{23-26}, f_{29-30}, 2 points in f_{28}, 3 points in f_{27}. In the f_{23} and the f_{24}, the FHBBPSO gets the first rank while the DLS-BBPSO gets the second. The advantage of the FHBBPSO is 1.58% and 0.23%. In the f_{25}, the FHBBPSO and the DLS-BBPSO give the same results and occupy the first rank. In the f_{26}, the FHBBPSO and the ETI-DE get the first rank with a same result. In the f_{27}, the FHBBPSO gets the third rank. The result is 189.46% larger than the result of the DE. In the f_{28}, the FHBBPSO gets the second rank. The result is 2.58% larger than the result of the DLS-BBPSO. In the f_{29} and the f_{30}, the FHBBPSO gets the first rank while the DLS-BBPSO gets the second. The results of the DLS-BBPSO is 3.35%, 57.53% larger than the results of the FHBBPSO.

In the Table 9.7, the rank results are displayed in groups. It can be seen that in the unimodal functions group, the FHBBPSO gets 1.667 points while other 3 algorithms all get 2.667 points. It is reasonable to consider that the FHBBPSO can reach the more accurate point in unimodal

functions. In group 2, the FHBBPSO gets 1.846 and the ETI-DE gets 1.615. Both the DE and the DLS-BBPSO get an average point above 3. We can reasonably believe that the searchability of the FHBBPSO on multimodal functions is near but not as good as the ETI-DE. We believe that the fusion process gives the swarm more chance to escape from a local minimum. In group 3, the two DE-based algorithms lead but the FHBBPSO still perform better than the DLS-BBPSO. In Group 4, the FHBBPSO performances best in the fours test algorithms. It is equitable to claim that the cooperation of the fission and the fusion process works well on the composition functions. From the global average point, there is no doubt that FHBBPSO gives the best results in these experiments.

On the side of the FHBBPSO, the fission process gives a particle the chance to communicate with both adjacent and distant particles. In the normal unimodal functions, the swarm may get a fast convergence and the increase in precision if particles exchange information with adjacent particles. The long-distance communication is a supplementary option. For instance, in the 3-dimension map for the 2-dimension f_3 [45], it can be seen that the function has a smooth but narrow ridge. A particle may cross the ridge if it communicates with a better and distant particle. Also, the fusion process ensures that the swarm has an impartial convergence speed. Particles in different local groups will keep searching around their leaders. The best local group will enhance its search ability by merging the worst local group. This pattern gives the swarm an ability to search in different environments. In the Group 4, composition functions have different properties around different local optima. The fusion process ensures that different local groups have the autonomy in their corresponding area. Hence the results of the FHBBPSO has a great lead in the Group 4. Moreover, the proposed method is exhaustive parameter-free. No human intervention is needed before and during the iteration. The number and the size of the local groups will be only decided by the test functions. All of these features make the FHBBPSO a widely used and powerful method for single-objective problems.

Table 9.4: Comparisons of the empirical errors between the ETI-DE, the DE, the DLS-BBPSO and the FHBBPSO. f_1-f_{10}.

F		ETI-DE[28]	DE[27][28]	DLS-BBPSO [36]	FHBBPSO
	Mean	8.417E+05	1.399E+06	1.021E+06	**4.612E+05**
f_1	SD	(2.873E+05)	(5.347E+05)	(8.8979E+05)	(2.698E+05)
	Point	2	4	3	1
	Mean	3.045E+03	**1.913E+02**	3.310E+04	9.163E+03
f_2	SD	(4.719E+03)	(7.531E+02)	(3.671E+04)	(9.917E+03)
	Point	2	1	4	3
	Mean	3.588E−01	2.863E−01	**5.684e-14**	**5.684E−14**
f_3	SD	(7.479E−01)	(1.144E+00)	(8.159e-14)	(4.177E−14)
	Point	4	3	1	1
	Mean	8.793E+01	7.729E+01	**4.030E+00**	1.103E+01
f_4	SD	(8.976E+00)	(2.730E+01)	(1.026E+01)	(1.873E+01)
	Point	4	3	1	2
	Mean	2.042E+01	2.113E+01	2.104E+01	**2.024E+01**
f_5	SD	(1.022E−01)	(3.349E−02)	(3.860E−02)	(7.150E−02)
	Point	2	4	3	1
	Mean	1.786E+00	**1.215E+00**	4.089E+01	2.268E+01
f_6	SD	(1.671E+00)	(1.384E+00)	(9.534E+00)	(5.701E+00)
	Point	2	1	4	3
	Mean	**1.450E−04**	**1.450E−04**	7.300E−03	6.700E−03
f_7	SD	(1.036E−03)	(1.036E−03)	(9.00E−03)	(8.900E−03)
	Point	1	1	4	3
	Mean	**4.596E+01**	1.942E+02	1.555E+02	7.337E+01
f_8	SD	(1.224E+01)	(4.580E+01)	(3.141E+01)	(1.470E+01)
	Point	1	4	3	2
	Mean	**4.258E+01**	3.517E+02	2.676E+02	1.603E+02
f_9	SD	(1.185E+01)	(1.609E+01)	(7.002E+01)	(4.063E+01)
	Point	1	4	3	2
	Mean	1.007E+03	9.467E+03	3.298E+03	**5.878E+02**
f_{10}	SD	(4.188E+02)	(1.349E+03)	(7.193E+02)	(2.891E+02)
	Point	2	4	3	1

Table 9.5: Comparisons of the empirical error between the ETI-DE, the DE, the DLS-BBPSO and the FHBBPSO. f_{11}-f_{20}

F	Mean	ETI-DE[28]	DE [27][28]	DLS-BBPSO [36]	FHBBPSO
	Mean	**3.912E+03**	1.299E+04	7.063E+03	4.022E+03
f_{11}	SD	(1.003E+03)	(3.968E+02)	(2.360E+03)	(6.077E+0q)
	Point	1	4	3	2
	Mean	**1.181E−01**	3.243E+00	2.701E+00	1.733E−01
f_{12}	SD	(4.673E−02)	(2.661E−01)	(2.085E−01)	(9.240E−02)
	Point	1	4	3	2
	Mean	**2.078E−01**	4.575E−01	3.181E−01	2.249E−01
f_{13}	SD	(4.905E−02)	(4.489E−02)	(5.730E−024)	(4.030E−02)
	Point	1	4	3	2
	Mean	2.968E−01	3.369E−01	3.578E−01	**1.755E−01**
f_{14}	SD	(8.065E−02)	(1.081E−01)	(1.840E−01)	(5.230E−02)
	Point	2	3	4	1
	Mean	**6.090E+00**	3.150E+01	1.407E+01	1.350E+01
f_{15}	SD	(1.801E+00)	(1.147E+00)	(5.670E+00)	(4.578E+00)
	Point	1	4	3	2
	Mean	1.735E+01	2.211E+01	2.028E+01	**1.680E+01**
f_{16}	SD	(1.212E+00)	(3.114E−01)	(6.822E−01)	(8.683E−01)
	Point	2	4	3	1
	Mean	1.726E+04	**1.419E+04**	4.754E+04	4.861E+04
f_{17}	SD	(1.479E+04)	(9.317E+03)	(3.365E+04)	(3.845E+04)
	Point	2	1	3	4
	Mean	**1.342E+02**	**1.342E+02**	9.676E+03	1.301E+03
f_{18}	SD	(2.020E+01)	(1.028E+01)	(1.155E+04)	(1.546E+03)
	Point	1	1	4	3
	Mean	**5.682E+00**	1.191E+01	3.443E+01	1.416E+01
f_{19}	SD	(1.329E+00)	(6.751E−0)	(1.543E+01)	(8.703E+00)
	Point	1	2	4	3
	Mean	**3.173E+01**	9.887E+01	1.458E+02	7.671E+01
f_{20}	SD	(2.098E+01)	(1.069E+01)	(5.199E+01)	(2.681E+01)
	Point	1	3	4	2

Table 9.6: Comparisons of the empirical error between the ETI-DE, the DE, the DLS-BBPSO and the FHBBPSO. f_{21}-f_{30}. The average rank of all the 30 test functions is shown at the bottom of the table.

F	Mean	ETI-DE[28]	DE [27][28]	DLS-BBPSO [36]	FHBBPSO
f_{21}	Mean	**1.980E+03**	2.630E+03	2.606E+04	2.202E+04
	SD	(3.372E+03)	(5.177E+02)	(2.117E+04)	(2.047E+04)
	Point	1	2	4	3
f_{22}	Mean	7.907E+02	7.133E+02	7.808E+02	**5.087E+02**
	SD	(3.243E+02)	(4.126E+02)	(2.613E+02)	(2.016E+02)
	Point	4	2	3	1
f_{23}	Mean	3.440E+02	3.440E+02	3.424E+02	**3.370E+02**
	SD	(2.870E−13)	(4.168E−13)	(1.847E+01)	(1.288E−12)
	Point	3	3	2	1
f_{24}	Mean	2.704E+02	2.704E+02	2.647E+02	**2.641E+02**
	SD	(2.061E+00)	(2.504E+00)	(5.717E−01)	(6.723E+00)
	Point	3	3	2	1
f_{25}	Mean	2.055E+02	2.054E+02	**2.004E+02**	**2.004E+02**
	SD	(4.956E−01)	(4.166E−01)	(1.531E−01)	(7.230E−02)
	Point	4	3	1	1
f_{26}	Mean	**1.002E+02**	1.005E+02	1.003E+02	**1.002E+02**
	SD	(5.262E−02)	(5.418E−02)	(5.080E−02)	(3.4800E−02)
	Point	1	4	3	1
f_{27}	Mean	3.765E+02	**3.662E+02**	1.332E+03	1.060E+03
	SD	(3.819E+01)	(3.538E+01)	(1.090E+02)	(1.358E+02)
	Point	2	1	4	3
f_{28}	Mean	1.086E+03	1.064E+03	**3.910E+02**	4.011E+02
	SD	(3.219E+01)	(4.712E+01)	(1.416E+01)	(5.464E+00)
	Point	4	3	1	2
f_{29}	Mean	1.003E+03	9.907E+02	2.191E+02	**2.120E+02**
	SD	(2.532E+02)	(2.505E+02)	(1.888E+01)	(2.433E+00)
	Point	4	3	2	1
f_{30}	Mean	8.355E+03	8.296E+03	1.083E+03	**6.875E+02**
	SD	(3.688E+02)	(3.421E+02)	(3.453E+02)	(2.090E+02)
	Point	4	3	2	1
Average Point		2.133	2.867	2.900	1.867

Table 9.7: Comparisons of the rank result between the ETI-DE, the PBBPSO, the DLS-BBPSO and the FHBBPSO. The average point is calculate from the 30 results. Best ranks are shown in **bold**.

	ETI-DE[28]	DE [27][28]	DLS-BBPSO [36]	FHBBPSO
Group 1	2.667	2.667	2.667	**1.667**
Group 2	**1.615**	3.385	3.077	1.846
Group 3	**1.667**	1.833	3.667	2.667
Group 4	3.625	2.875	2.125	**1.375**
Average Point	2.133	2.867	2.900	**1.867**

9.4 Summary

A fission-fusion hybrid bare bones particle swarm optimizer (FHBBPSO) is proposed in this chapter. A fission strategy and a fusion strategy are combined in the FHBBPSO for solving the single-objective problems. In the fission process, one complete particle swarm will split to several local groups and occupy different areas of the search region. Each local group will only focus on finding the local minimum of its corresponding region. The fission strategy is a parameter-free method. No previous work or parameter is needed for the iteration. In the fusion process, the best group will keep encroaching the worst group until all of the particles are in one local group. Then the fission process is on working. The fusion method is inspired by the team behavior of the chimpanzees. In the society of chimpanzees, stronger group can hold more resources and will annex weaker groups. By merging the worse group, the best local group increase its search ability by the increasing of its population. To verify the optimization ability of the FHBBPSO, the proposed method runs over the CEC2014 benchmark functions. The test functions are composed of four parts: the unimodal functions, the simple multimodal functions, the hybrid functions, and the composition functions. Also, the DE, the ETI-DE, and the DLS-BBPSO are used in the control group. The FHBBPSO gives best results in the

unimodal group and the composition group. This can prove that the FHBBPSO has excellent search capabilities in complex environments. Consider about the average rank among all 30 test functions, the FHBBPSO still performances best. Hence we are reasonable to believe that the proposed method has the best performance on the CEC2014 functions.

Chapter 10

Conclusions and Future Work

10.1 Contribution of this study

In particle swarm optimization algorithms, parameters are used to control the behavior of particles. It is difficult either to find a set of parameter that able to solve all problems or to adjust parameters for every single problem. In the bare bones particle swarm optimization algorithms, no parameter is need but the accuracy is not low with complex problems. In this study, seven BBPSO-based algorithms are proposed. These algorithms are designed for different purposes. Four of the them are designed with the swarm division which are able to converge to the global optimum fast. The other three algorithms are designed with the swarm reconstruction which are able to slow down the convergence and solve complex problems. Moreover, the convergence speed is not controlled by parameters but by revising the structures of local groups. Compared with existing methods, no parameter is needed to control the behavior of the particles, and higher accuracy can be presented in experiments. These advantages make proposed methods able to apply to life and engineering applications.

10.2 Conclusions

The single-objective optimization problem is the basic component of the optimization problems. It is widely exist in researches and lives. The single-objective optimization problems are composed by four major sub-problems: the control of convergence speed, the precision in unimodal problems, the local minimal escape in multimodal problems, and wide adaptability for multiple problems. In this thesis, seven different evolutionary swarm-based algorithms are proposed for different aspects of the single-objective optimization problems.

In the PBBPSO, every two particles are binding as a computing unit. Different evolutionary rules are applied to the particles in a computing unit to slow down the diversity losing. On the other hand, the DABBPSO divides the swarm to two subgroups. Different evolutionary rules are applied to different groups to get a meticulous search in their corresponding areas.

In the TBBPSO, the three-particle model improves the search ability of the computing units. Two different structures are available for the shifted and rotated problems. In addition, the DLS-BBPSO is a parameter-free algorithm. It can divide the swarm into different model without human intervention. The structure of the particle swarm depends on the test problems which make the swarm able to solve shifted and rotated prblems.

In the BBPSO-C, a shadow particle swarm is used to assist the original swarm. These two swarms will change particles after each iteration. The original swarm will focus on local search while the shadow swarm is aiming at global search. Moreover, to handle the high iteration time problems, the elite selection strategy is used in the DRBBPSO. A dynamic number of particles will be selected from each iteration. The swarm will be reconstructed when the number of elite particles reaches the number of particles in the original swarm. The reconstruction will keep the diversity of the swarm after a huge amount of iteration.

In the FHBBPSO, a fission strategy and a fusion strategy work together for the global opti-

mum. The fission process split the swarm to several subgroups. Each subgroup will only focus on its corresponding area. The fusion will keep the subgroups merging until they become one group again. No parameters are needed during this process. Also, the dynamic structure of the particle swarm makes the FHBBPSO able to solve various optimization problems.

To conclude, seven different evolutionary swarm-based algorithms are developed for single-objective optimization problems. Each of them has shown good performances on one or more aspects of optimization activities. In researches on particle swarm optimization, this thesis has clarified one approach to achieve for realizing high search accuracy without parameter adjustment.

10.3 Future work

First of all, compared with the best of the methods that require parameter adjustment, proposed algorithms have not catch up with hybrid functions. In future work, I want propose new structures that able to present more precise results on hybrid functions. The proposed algorithms has shown excellent performances on single-objective optimization problems. The convergence control and parameter-free strategies have the potential to be applied to multi-objective problems. Hence, one of the future work is apply the proposed methods to multi-objective problems.

Also, in real-world problems, the dynamic problems grow complex and high dimensional. Applying proposed methods to these problems is an important part of future work.

Bibliography

[1] T. Bäck, U. Hammel, and H. P. Schwefel, "Evolutionary computation: Comments on the history and current state," *Evolutionary Computation: The Fossil Record*, no. April 2014, pp. 15–28, 1998.

[2] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 2. IEEE, 1999, pp. 1470–1477.

[3] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical Computer Science*, 2005.

[4] C. Zhang, D. Ouyang, and J. Ning, "An artificial bee colony approach for clustering," *Expert Systems with Applications*, 2010.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, pp. 1942–1948 vol.4, 1995.

[6] Z.-L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *IEEE Transactions on Power Systems*, vol. 18, no. 3, pp. 1187–1195, aug 2003.

[7] Y. del Valle, G. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. Harley, "Particle

Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171–195, apr 2008.

[8] S. Naka, T. Genji, T. Yura, and Y. Fukuyama, "A hybrid particle swarm optimization for distribution state estimation," *IEEE Transactions on Power Systems*, vol. 18, no. 1, pp. 60–68, feb 2003.

[9] S.-W. Lin, K.-C. Ying, S.-C. Chen, and Z.-J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1817–1824, nov 2008. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957417407003752

[10] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle Swarm Optimization in Wireless-Sensor Networks: A Brief Survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 2, pp. 262–267, mar 2011.

[11] K. Parsopoulos and M. Vrahatis, "Recent approaches to global optimization problems through Particle Swarm Optimization," *Natural Computing*, vol. 1, no. 2/3, pp. 235–306, 2002.

[12] J. Hu, Y. Wang, E. Zhou, M. C. Fu, and S. I. Marcus, "A Survey of Some Model-Based Methods for Global Optimization," in *Optimization, Control, and Applications of Stochastic Systems*. Boston: Birkhäuser Boston, 2012, pp. 157–179.

[13] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," *Proceedings of the 2007 IEEE Swarm Intelligence Symposium, SIS 2007*, no. Sis, pp. 120–127, 2007.

[14] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[15] J. Kennedy and R. Mendes, "Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 36, no. 4, pp. 515–519, 2006.

[16] J. J. Liang, A. K. Qin, S. Member, P. N. Suganthan, S. Member, and S. Baskar, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[17] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03*, 2003, pp. 80–87.

[18] M. Omran and S. Al-Sharhan, "Barebones particle swarm methods for unsupervised image classification," in *2007 IEEE Congress on Evolutionary Computation*, IEEE. IEEE, sep 2007, pp. 3247–3252.

[19] C. H. Chen, "Bare bone particle swarm optimization with integration of global and local learning strategies," *Proceedings - International Conference on Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 692–698, 2011.

[20] T. Blackwell, "A study of collapse in bare bones particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 354–372, 2012.

[21] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 634–647, 2013.

[22] R. A. Krohling and E. Mendel, "Bare bones particle swarm optimization with Gaussian or cauchy jumps," *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, pp. 3285–3291, 2009.

[23] C. H. Chen, "A revised bare bone particle swarm optimizer and its variant," *iFUZZY 2013 - 2013 International Conference on Fuzzy Theory and Its Applications*, pp. 488–493, 2013.

[24] M. Campos, R. A. Krohling, and I. Enriquez, "Bare bones particle swarm optimization with scale matrix adaptation," *IEEE Transactions on Cybernetics*, vol. 44, no. 9, pp. 1567–1578, 2014.

[25] T. J. Richer, "The Lévy Particle Swarm," *IEEE Congress on Evolutionary Computation*, pp. 808–815, 2006.

[26] R. Vafashoar and M. R. Meybodi, "Multi swarm bare bones particle swarm optimization with distribution adaption," *Applied Soft Computing*, vol. 47, pp. 534–552, 2016.

[27] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2202–2215, 2013.

[28] W. Du, S. Y. S. Leung, Y. Tang, and A. V. Vasilakos, "Differential Evolution With Event-Triggered Impulsive Control," *IEEE Transactions on Cybernetics*, vol. 47, no. 1, pp. 244–257, 2016.

[29] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," *Report*, no. May, pp. 1–50, 2005.

[30] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[31] 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 Constrained Real-Parameter Optimization," *KanGAL*, no. May, pp. 251–256, 2005.

[32] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization." *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 39, no. 6, pp. 1362–1381, 2009.

[33] Z. H. Zhan, J. Zhang, Y. Li, and Y. H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, 2011.

[34] Y. F. Li, Z. H. Zhan, Y. Lin, and J. Zhang, "Comparisons study of APSO OLPSO and CLPSO on CEC2005 and CEC2014 test suits," *2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings*, pp. 3179–3185, 2015.

[35] J. Guo and Y. Sato, "A pair-wise bare bones particle swarm optimization algorithm," in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, no. 1. IEEE, may 2017, pp. 353–358.

[36] ——, "A Bare Bones Particle Swarm Optimization Algorithm with Dynamic Local Search," in *Studies in Computational Intelligence*, 2017, vol. 248, pp. 158–165.

[37] P. Ghamisi, M. S. Couceiro, F. M. Martins, and J. A. Benediktsson, "Multilevel image segmentation based on fractional-order darwinian particle swarm optimization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 5, pp. 2382–2394, 2014.

[38] L. D. S. Coelho and B. M. Herrera, "Fuzzy Identification Based on a Chaotic Particle Swarm Optimization Approach Applied to a Nonlinear Yo-yo Motion System," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 6, pp. 3234–3245, 2007.

[39] Y. Gao, G. Zhang, J. Lu, and H. M. Wee, "Particle swarm optimization for bi-level pricing problems in supply chains," *Journal of Global Optimization*, vol. 51, no. 2, pp. 245–254, 2011.

[40] J. H. Zhang, Y. Zhang, and Y. Zhou, "Path planning of mobile robot based on hybrid multi-objective bare bones particle swarm optimization with differential evolution," *IEEE Access*, vol. 6, pp. 44 542–44 555, 2018.

[41] B. Jiang and N. Wang, "Cooperative bare-bone particle swarm optimization for data clustering," *Soft Computing*, vol. 18, no. 6, pp. 1079–1091, 2014.

[42] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[43] S. Sarkar and S. Das, "Multilevel image thresholding based on 2D histogram and maximum tsallis entropy - A differential evolution approach," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4788–4797, 2013.

[44] S. Das, A. Abraham, and A. Konar, "Automatic Clustering Using an Improved Differential Evolution Algorithm," vol. 38, no. 1, pp. 218–237, 2008.

[45] J. J. Liang, B. Y. Qu, P. N. Suganthan, and Q. Chen, *Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization*, 2014, no. November 2014.

Appendices

Appendix A

List of Research Paper

Refereed Journal Papers

First Author

[1] J. Guo and Y. Sato, A Pair-wise Bare Bones Particle Swarm Optimization Algorithm for Nonlinear Functions, International Journal of Networked and Distributed Computing, Volume 5, Issue 3, June 2017, pp. 143- 151.

[2] J. Guo and Y. Sato, A dynamic allocation bare bones particle swarm optimization algorithm and its application, Artif Life Robotics 23, 353–358 (2018).

[3] J. Guo and Y. Sato, A fission-fusion hybrid bare bones particle swarm optimization algorithm for single-objective optimization problems, Appl Intell 49, 3641–3651 (2019)

Refereed Conference Papers

First Author

[4] J. Guo and Y. Sato, Modified bare bones particle swarm optimization with separate exploration, The Twenty-Second International Symposium on Artificial Life and Robotics 2017, pp. 545- 548.

[5] J. Guo and Y. Sato, A pair-wise bare bones particle swarm optimization algorithm, 2017 IEEE/ACIS 16th International Conference on Computer and Information Science. pp. 353- 358.

[6] J. Guo and Y. Sato, A bare bones particle swarm optimization algorithm with dynamic local search, Lecture Notes in Computer Science, vol 10385, Advances in Swarm Intelligence, pp. 158-165.

[7] J. Guo and Y. Sato, A Hierarchical bare bones particle swarm optimization algorithm. 2017 IEEE International Conference on Systems, Man, and Cybernetics, pp. 1936–1941.

[8] J. Guo and Y. Sato, A resources pre-allocated bare bones particle swarm optimization algorithm, The Twenty-Third International Symposium on Artificial Life and Robotics 2018, pp. 470–473.

[9] J. Guo and Y. Sato, A Dynamic reconstruction bare bones particle swarm optimization algorithm, 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1-6.

[10] J. Guo and Y. Sato, A novel bare-bones particle swarm optimization algorithm with co-evaluation, The 22nd Asia Pacific Symposium on Intelligent and Evolutionary Systems 2018, pp 116-119.

Corresponding Chapters with Papers

Chapter 1. Introduction

Chapter 2. Related Work

Chapter 3. The Pair-wise Bare Bones Particle Swarm Optimization Algorithm

[1] J. Guo and Y. Sato, A Pair-wise Bare Bones Particle Swarm Optimization Algorithm for Nonlinear Functions, International Journal of Networked and Distributed Computing, Volume 5, Issue 3, June 2017, pp. 143- 151.

[4] J. Guo and Y. Sato, Modified bare bones particle swarm optimization with separate exploration, The Twenty-Second International Symposium on Artificial Life and Robotics 2017, pp. 545-548.

[5] J. Guo and Y. Sato, A pair-wise bare bones particle swarm optimization algorithm, 2017 IEEE/ACIS 16th International Conference on Computer and Information Science. pp. 353- 358.

Chapter 4. The Bare Bones Particle Swarm Optimization Algorithm with Pre-processing

[2] J. Guo and Y. Sato, A dynamic allocation bare bones particle swarm optimization algorithm and its application, Artif Life Robotics 23, 353–358 (2018).

[8] J. Guo and Y. Sato, A resources pre-allocated bare bones particle swarm optimization algorithm, The Twenty-Third International Symposium on Artificial Life and Robotics 2018, pp. 470–473.

Chapter 5. The Hierarchical Bare Bones Particle Swarm Optimization Algorithm

[7] J. Guo and Y. Sato, A Hierarchical bare bones particle swarm optimization algorithm. 2017 IEEE International Conference on Systems, Man, and Cybernetics, pp. 1936–1941 .

Chapter 6. The Bare Bones Particle Swarm Optimization Algorithm with Dynamic Local Search

[6] J. Guo and Y. Sato, A bare bones particle swarm optimization algorithm with dynamic local search, Lecture Notes in Computer Science, vol 10385, Advances in Swarm Intelligence, pp. 158-165.

Chapter 7. The Bare Bones Particle Swarm Optimization Algorithm with Co-evaluation

[10] J. Guo and Y. Sato, A novel bare-bones particle swarm optimization algorithm with co-evaluation, The 22nd Asia Pacific Symposium on Intelligent and Evolutionary Systems 2018, pp 116-119.

Chapter 8. The Bare Bones Particle Swarm Optimization Algorithm with Reconstruction

[9] J. Guo and Y. Sato, A Dynamic reconstruction bare bones particle swarm optimization algorithm, 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1-6.

Chapter 9. The Fission-Fusion Bare Bones Particle Swarm Optimization Algorithm

[3] J. Guo and Y. Sato, A fission-fusion hybrid bare bones particle swarm optimization algorithm for single-objective optimization problems, Appl Intell 49, 3641–3651 (2019)

Chapter 10. Conclusions and Future Work