Research on Homogeneous and Heterogeneous Particle Swarm Optimization for Global Optimization Problems

YANG, Shiqin / 楊, 詩琴

(開始ページ / Start Page)
1
(終了ページ / End Page)
180
(発行年 / Year)
2017-03-24
(学位授与番号 / Degree Number)
32675甲第399号
(学位授与年月日 / Date of Granted)
2017-03-24
(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/10.15002/00013955

Doctoral Dissertation Reviewed by
Hosei University

大域的最適化問題のための同種および異種粒子群最適化法の研究

Research on Homogeneous and Heterogeneous Particle Swarm Optimization for Global Optimization Problems

楊 詩琴
Shiqin Yang

Supervisor: Professor **Yuji Sato**, Hosei University

Abstract

The premature convergence problem and the exploration-exploitation trade-off problem are the two major problems encountered by many swarm intelligence algorithms in both global optimization and large scale global optimization. This thesis proposes that the two main problems could be handled by several variants of Particle Swarm Optimization (PSO) developed below. Five variants of homogeneous PSO have been developed for multimodal and large scale global optimization problems, and two variants of dynamic heterogeneous PSO for complex real-world problems.

First of all, an individual competition strategy is proposed for the new variant of PSO, namely Fitness Predator Optimization (FPO), for multimodal problems. The development of individual competition plays an important role for the diversity conservation in the population, which is crucial for preventing premature convergence in multimodal optimization.

To enhance the global exploration capability of the FPO algorithm for high multimodality problems, a modified paralleled virtual team approach is developed for FPO, namely DFPO. The main function of this dynamic virtual team is to build a paralleled information-exchange system, strengthening the swarm's global searching effectiveness. Furthermore, the strategy of team size selection is defined in DFPO named as DFPO-r, which based on the fact that a dynamic virtual team with a higher degree of population diversity is able to help DFPO-r alleviate the premature convergence and strengthen the global exploration simultaneously. Experimental results demonstrate that both DFPO-r and DFPO have desirable performances

for multimodal functions. In addition, DFPO-r has a more robust performance in most cases compared with DFPO.

Using hybrid algorithms to deal with specific real-world problems is one of the most interesting trends in the last years. In this thesis, we extend the FPO algorithm for fuzzy clustering optimization problem. Thus, a combination of FPO with FCM (FPO-FCM) algorithm is proposed to avoid the premature convergence and improve the performance of FCM.

To handle the large scale global optimization problem, a variant of modified BBPSO algorithm incorporation of Differential Evolution (DE) approach, namely BBPSO-DE, is developed to improve the swarm's global search capability as the dimensionality of the search space increases.

To the best of our knowledge, the Static Heterogeneous PSO (SHPSO) has been studied by some researchers, while the Dynamic Heterogeneous PSO (DHPSO) is seldom systematically investigated based on real problems. In this thesis, two variants of dynamic Heterogeneous PSO, namely DHPSO-d and DHPSO-p are proposed for complex real-world problems. In DHPSO-d, several differential update rules are proposed for different particles by the trigger event. When the global best position p_g is considered stagnant and the event is confirmed, then p_g is reset and all particles update their positions only by their personal experience. In DHPSO-p, two proposed types of topology models provide the particles different mechanism choosing their informers when the swarm being trapped in the local optimal solution. The empirical study of both variants shows that the dynamic self-adaptive heterogeneous structure is able to effectively address the exploration-exploitation trade-off problem and provide excellent optimal solutions for the complex real-world problem.

To conclude,the proposed biological metaphor approaches provide each of the PSO algorithms variants with different search characteristics, which makes them more suitable for different types of real-world problems.

Contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AFSA	Artificial Fish-Swarm Algorithm
AIS	Artificial Immune System
BA	Bat-inspired Algorithm
BBPSO-DE	Bare Bones Particle Swarms Optimization with Differential Evolution
BBPSO-MC	Bare Bones Particle Swarms Optimization with Mutation and Crossover
BBPSO	Bare Bones Particle Swarms Optimization
BPSO	Binary Particle Swarm Optimization
CC	Cooperative Co-evolution
CI	Computational Intelligence
CLPSO	Comprehensive Learning Particle Swarm Optimization
CSO	Competitive Swarm Optimization

DE	Differential Evolution
DFPO-r	Dynamic Fitness Predator Optimization with random team size
DFPO	Dynamic Fitness Predator Optimization
DHPSO-d	Dynamic Heterogeneous Particle Swarm Optimization with differential rules
DHPSO-p	Dynamic Heterogeneous Particle Swarm Optimization with polymorphic models
DHPSO	Dynamic Heterogeneous Particle Swarm Optimization
DPO	Dolphin Partner Optimization
EA	Evolutionary Algorithms
EC	Evolutionary Computation
EDA	Estimation of Distribution Algorithms
EPUS-PSO	Efficient Population Utilization Strategy for Particle Swarm Optimization
FA	Firefly Algorithm
FCM	Fuzzy c-means algorithm
FIPS	Fully Informed Particle Swarm
FPO-FCM	Fuzzy c-means Clustering with Fitness Predator Optimization
FPO	Fitness Predator Optimization
FS	Fuzzy System
GBPSO	Global Best Particle Swarm Optimization

GO	Global Optimization
GWO	Grey Wolf Optimization
HGLPSO	static Heterogeneous Particle Swarm Optimization combined with GBPSO and LBPSO
HPSO	Heterogeneous Particle Swarms Optimization
KHM	K-Harmonic Means algorithm
LBPSO	Local Best Particle Swarm Optimization
LSGO	Large Scale Global Optimization
MCPSO	Modified Competitive Particle Swarm Optimization
NN	Neural Networks
PC	Premature Convergence
PPO	Predator-Prey Optimization
PPSO-FCM	Fuzzy C-Mean based on Picard iteration and PSO
PSO	Particle Swarm Optimization
QE	Quantum Error
QPSO	Quantum-behaved Particle Swarm Optimization
SGA	Simple Genetic Algorithm
SHPSO	Static Heterogeneous Particle Swarm Optimization
SI	Swarm Intelligence

SPSO	Standard Particle Swarms Optimization
WDBC	Wisconsin Diagnostic Breast Cancer

Chapter 1

Introduction

1.1 Research Background

Computational intelligence (CI) is a set of nature-inspired computational methodologies and approaches to provide solutions for complicated problems and inverse problems [82]. It primarily includes artificial neural networks (NN), evolutionary computation (EC), swarm intelligence(SI), artificial immune system(AIS), and fuzzy systems (FS). Each of the CI parts has its origins in biological systems. NNs model biological neural systems, EC models natural evolution (including genetic and behavioral evolution), SI models the social behavior of organisms living in swarms or colonies, AIS models the human immune system and FS originated from studies of how organisms interact with their environment. The techniques from these five components can be combined to form hybrid systems.

Swarm intelligence (SI) is the collective behavior of decentralized, self-organized systems, natural or artificial. The SI systems typically consist of a population of simple agents interacting locally with one another and with their environment. Examples in natural systems of SI include ant colonies, bird flocking, animal herding, bacterial growth and so on. Nowadays, swarm intelligence is becoming a new research highlight. Most of research theo-

ries and applications related to swarm intelligence proves that it is a kind of effective method to solve variety of global optimal problems.

Particle Swarm Optimization (PSO) belongs to the field of Swarm Intelligence and is a sub-field of Computational Intelligence. It is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995 [42], inspired by social behavior of bird flocking or fish schooling. Based on the research of PSO, in this thesis, we propose several improved homogeneous and heterogeneous PSO variants for global optimization problems.

1.2 Research Motivation

1.2.1 Global Optimization Problem

Global Optimization (GO) aims at characterizing and computing global optimal solutions to problems with nonconvex, multimodal, or badly scaled objective functions [37]. Many real-world problems, such as engineering, biotechnology, data analysis, environmental management, financial planning and other related areas, can be formulated as global optimization problems.

This thesis considers the following global optimization problem:

$$\begin{cases} f : S \to \mathfrak{R} \\ f(x^*) \leqslant f(x), \exists x^* \in S; \forall x \in S \end{cases} \tag{1.1}$$

where $S \subset \mathfrak{R}^D$ is a nonempty compact set with D dimensions [74]. $f : S \to \mathfrak{R}$ stands for a real-valued nonlinear objective function for mapping from D dimensional space to one dimensional fitness value. In general, due to the absence of structural information and the presence of many local extrema, global optimization problems are extremely difficult to solve exactly. To solve such a problem, there are many different types of methods in

the literature on global optimization, which can be categorized based on different criteria. For instance, they can be classified by the properties of algorithms that search for new candidate solutions, such as deterministic and stochastic algorithms [74]. Most deterministic algorithms involve the application of heuristics, such as modifying the trajectory (trajectory methods) or adding penalties (penalty-based methods), to escape from local minima. On the other hand, stochastic algorithms do not require any properties of the objective function. Therefore, more attention has been paid to stochastic algorithms. Perhaps the most common problem encountered by many GO methods, either deterministic or stochastic is the problem of local minima. Especially in multimodal functions, the existence of many local minima makes it quite difficult for most techniques to detect the global minimum.

1.2.2 Large Scale Global Optimization Problem

A Large Scale Global Optimization (LSGO) problem can be mathematically stated in the equation 1.1. Specifically, there are D number of variables in large scale setting ($D > 100$). Most real-world optimization problems involve a large number of decision variables may be formulated as large scale global optimization problems. For example, inverse problems in the biological systems are a large-scale and highly time-consuming optimization problems [67], [60]. With the arrival of big data, there is an unprecedented demand to solve optimization problems with a number of feature variables, training instances, and classes [10]. The recent advance in the area of machine learning has witnessed very large scale optimization problems encountered in training deep neural network architectures (so-called deep learning), some of which are involved in optimizing over a billion of connection weights in a very large neural network [33]. These models are faced with some challenging characteristics such as strong interaction among parameters and high multimodality.

Many optimization algorithms attempt to solve LSGO problems efficiently in a given number of fitness evaluation budget. Unfortunately, most optimization methods suffer from

the "curse of dimensionality" [4], which implies that their performance deteriorates quickly as the dimensionality of the search space increases. There are two major reasons for the performance deterioration of these algorithms. Firstly, the solution space of a problem often increases exponentially with the problem dimension and more efficient search strategies are required to explore all promising regions within a given time budget. Secondly, the search space exponentially increases with the problem size; so an optimization algorithm must be able to explore the entire search space efficiently, which is not a trivial task. In addition, the characteristics of a problem may change with the scale. Because of such a worsening of the features of an optimization problem resulting from an increase in scale, a previously successful search strategy may no longer be capable of finding the optimal solution.

1.2.3 Motivations of Our Research

Individual Competition Strategy for Multimodal Optimizations

A major issue in global optimization, especially in multimodal optimization is premature convergence problem. The term of premature convergence means that a population for an optimization problem converged too early, resulting in being suboptimal. An accepted hypothesis is that maintenance of high diversity is crucial for preventing premature convergence in multimodal optimization [27], [73]. Many kinds of optimization algorithms are proposed to improve the diversity of the population. Some of them are inspired by the social behavior of swarms, herds in nature. In addition, the hunting and search behaviors of predator are implemented by more and more researchers and proved to be an effective method. However, few of SI techniques focus on individual competition and independent self awareness. We note that the individual competition is more likely to reduce the rapid social collaboration process and increase the ability of being out of the local optimum. This motivated our attempt to propose a new population-based algorithm with an individual competition strategy to avoid premature convergence for multimodal optimization problems.

Hybrid Algorithms for Specific Problem

The use of hybrid algorithms to deal with specific real-world problems is a fact that proves that hybridization is a powerful tool far beyond the discrete algorithm. For instance, in the field of clustering, Fuzzy c-means (FCM) is one of the most popular algorithms. The objective function of the FCM is the multimodal function which means that it may contain many local minima. Consequently, while minimizing the objective function, there is possibility of getting stuck at local minima or saddle points. To increase the probability of locating the global optimum, some researchers adopt the stochastic methods such as evolutionary or swarm-based methods to increase the global convergence ability of fuzzy clustering. In this thesis, we propose a hybrid fuzzy clustering algorithm combination of our new population-based algorithm to provide the excellent optimal solution and avoid to be trapped in the local minima simultaneously.

Differential Evolution Strategy for LSGO Problems

Recently, LSGO has become a well-recognized field of research and various meta-heuristic algorithms such as Simulated Annealing (SA) [8] and variant population-based algorithms, such as, Evolutionary Algorithms (EAs) [3], Estimation of Distribution Algorithms (EDAs) [52], [65], Particle Swarm Optimization (PSO) [42] have been applied to solve them. However, generally speaking, the performance of these algorithms deteriorates when tackling the high dimensional problems [89]. Several particular mechanisms have been proposed to handle LSGO problems. Basically, two main categories of approaches can be found, namely, Cooperative Coevolution (CC) algorithms with problem decomposition strategy [76], and non-decomposition based methods [62]. The results of these methods show an efficient performance on scalable LSGO benchmark functions (up to 1000D), however, its performance influenced by their control parameters. The best values for the control parameters remain problem dependent, and need to be tuned for each problem. It is motivating to consider these

reasons and difficulties to propose new approaches for tackling LSGO problems. In this thesis, the mutation and crossover operators of Differential Evolution (DE) are employed for the modified Bare Bones Particle Swarm Optimization (BBPSO) algorithm, which increases the swarm diversity and enhances the search performance as the dimensionality of problem increases. In addition, one key advantage of our proposed algorithm is that there is no need to specify any control parameter for each problem that is often required in traditional stochastic algorithms.

Dynamic Heterogeneity to Complex Real-World Problems

In population-based algorithms, finding the global optimal solution of a problem is based on two cornerstones, namely exploration: global search, exploring all over the search space to find promising regions and exploitation: local search, exploiting the identified promising regions to tune the search for the global optimum. It is worth noting that, emphasizing on exploration will lead to waste of time searching over inferior regions of the search space and slow down the convergence rate. On the other hand, emphasizing on exploitation will cause loss of diversity early in the search process, thereby possibly getting stuck into a local optimum. Therefore, in the population-based evolutionary algorithms, it is important to obtain the balance between exploration and exploitation of the search space [23], [17].

Most population-based algorithms and their modifications make use of homogeneous swarms where all of the individuals follow exactly the same behavior. However, modelled with populations of heterogeneous individuals, the optimization algorithm has the ability to maintain an appropriate balance between exploration and exploitation throughout the search process. Recently, heterogeneous systems have drawn the attention of researchers working in different areas of swarm intelligence because designing heterogeneous models more accurately and approximately resembles real circumstances. To the best of our knowledge, the Static Heterogeneous Particle Swarm Optimization (SHPSO) has been studied by some

researchers, while the Dynamic Heterogeneous Particle Swarm Optimization (DHPSO) is seldom systematically investigated based on real problems. In this thesis, we propose two different DHPSO models and extend the analysis of DHPSO models to study their scalability to different dimensional optimization problems.

1.3 Overview of the Thesis

1.3.1 Evaluation Method

All experiments are implemented on a PC with an Intel Core i7 860, 2.8 GHz CPU and Microsoft Windows 7 Professional SP1 64-bit operating system, and all algorithms are written in the Matlab language. Each test algorithm was run on an array of generic benchmark problems. All algorithms were randomly initialized in an area equal to one quarter of the feasible search space in every dimension that was guaranteed not to contain the optimal solution.

Benchmark Problems

For any new optimization, it is essential to validate its performance and compare with other existing algorithms over a good set of test functions. Benchmark problems is a set of test functions which can be used to test the effectiveness of global optimization algorithms. There have been many tests or benchmark functions reported in the literature [38]. In this thesis, we carefully chose some benchmark problems for their variety, for instance, some functions are simple unimodal problems, some are highly complex multimodal problems with many local minima, and the others are multimodal problems with a few local minima. All these benchmark problems are collected from the website of the Virtual Library of Simulation Experiments [Surjanovic and Bingham].

Statistical Analysis

For each algorithm, the statistical results including the best global optimum, the worst optimum solution, the average of best global value of each benchmark function and running time with the same fixed times iteration after a number of independent trials. Having gathered some empirical data, differences in performance between several versions of algorithms can become notable. In this thesis, we partly adopt the practice of performing t-tests on pairs of groups of data; these tests give a $p-$value which is compared to a constant called α to determine whether a difference is significant or not.

1.3.2 Organization of the Thesis

The main structure of the thesis is thus twofold. On the first hand, five variants of homogeneous PSO have been developed, each of them implementing the biological metaphor in their own particular way. This provides each of these approaches with different search characteristics, which make them more suitable for different types of problems. On the second hand, two dynamic Heterogeneous PSO variants were proposed for complex real-world problems. In **Chapter 3**, a variant of PSO algorithm employed an individual competition strategy, namely FPO, is proposed for multimodal problems. Then in **Chapter 4**, this variant combination of Fuzzy c-means algorithm, named as FPO-FCM, is proposed to provide the excellent data clustering solution. To enhance the global exploration of the FPO algorithm for high multimodality problem, a paralleled virtual team approach is introduced in FPO. Thus, an improved Dynamic FPO algorithm (DFPO) is presented in **Chapter 5**. Meanwhile, the DFPO with dynamic team size selection strategy, named as DFPO-r, is also provided in **Chapter 5**. Finally, to handle the large scale global optimization problem, the fifth variant of BBPSO algorithm incorporation of Differential Evolution (DE) approach, namely BBPSO-DE, is developed in **Chapter 6**. Finally, a comparative analysis has been performed to some homogeneous PSO algorithms and static heterogeneous PSO algorithms,

we prefer to propose two dynamic Heterogeneous PSOS variants (DHPSO-d and DHPSO-p) in **Chapter 7** for complex real-world problems.

In summary, the thesis is organized as follows.

Chpater 1 presents the background and the motivation of our research, summarizes the solutions of our research and provides an overview of the thesis.

Chpater 2 introduces the standard particle swarm optimization algorithm and several variants of PSO.

Chpater 3 proposes a variant of PSO algorithm employed an individual competition strategy, namely FPO, for multimodal problems.

Chpater 4 proposes a hybrid fuzzy clustering algorithm (FPO-FCM) which combination of FPO and FCM for data clustering.

Chpater 5 presents DFPO and DFPO-r algorithms to enhance the global exploration of FPO for multimodal problem.

Chpater 6 develops a variant of BBPSO algorithm with Differential Evolution (DE) approach, namely BBPSO-DE, for large scale global optimization problem.

Chpater 7 proposes two dynamic Heterogeneous PSO variants, DHPSO-d and DHPSO-p for complex real-world problems.

Chpater 8 gives a conclusion on the current research work presented in this thesis and summarizes the future research trends we are focusing on.

Chapter 2

Standard Particle Swarm Optimization (SPSO)

Recently, many nature-inspired optimization algorithms have been developed successfully for solving a wide range of optimization problems. For instance, Evolutionary Algorithms (EAs), Simulated Annealing, Differential Evolution (DE), Ant Colony Optimization (ACO), Estimation of Distribution Algorithms (EDA) and Particle Swarm Optimization (PSO) are just some representative examples among many others. These meta-heuristic algorithms do not rely on gradient information, and are less likely to be stuck on local optima because of their use of population-based candidate solutions, thereby offering significant advantages over traditional single-point and derivative-dependent methods. Among these population-based algorithms, Particle Swarm Optimization (PSO) is a member of the wide category of Swarm Intelligence methods for solving optimization problems.

2.1 Basic Concept of SPSO

Particle Swarm Optimization (PSO) was introduced by Russell C. Eberhart and James Kennedy in 1995. The original PSO algorithm [42] was inspired by the social behaviour of biological

organisms, birds flocking while searching for a food source in a given area. There is a general belief, and numerous instances coming from nature enforce the view, that social sharing of information among the individuals of a population, may provide an evolutionary advantage. This was the core idea behind the development of PSO [20].

In SPSO, each individual is named as a particle which, in fact, represents a potential solution to a problem. Particles move through the search space being combined of an attraction to the best solution that they individually have found, and an attraction to the best solution that any particle in their neighborhood has found. Generally, a neighborhood is defined for each individual as the subset of particles which it is able to communicate with. These neighborhoods can involve two or more particles which are predetermined to act together, or subsets of the search space that particles happen into during testing. The use of neighborhoods often helps the algorithm to avoid getting stuck in local minima. When the neighborhood is expanded into include all members of the population, so all particles are influenced by the best solution found by any member of the swarm, which has been known as a global neighborhood, or *gbest* model. The *lbest* model, often referred to as a local topology, constitutes perhaps the most significant variation to the original PSO algorithm, and was proposed in one of the very first PSO publications [21]. Much of the PSO literature uses the local topology to describe not just a single swarm model, but applies it to any swarm model without global communication. A number of different limited neighborhood communication topologies have been listed in this chapter.

2.1.1 Neighborhood Communication Topology

There are four main communication topologies shown in Fig.2.1 with populations of 8 particles: star, circle, wheel and isolated.

* **Star** (*gbest*): Each particle is connected to every other member of the swarm.

* **Circle** (*lbest*): Each individual is connected to its K immediate neighbors only. In a regular ring topology, $K = 2$, the individual is affected by only its immediately adjacent neighbors.

* **Wheel**: One particle is connected to all others, and they are connected to only that one.

* **Isolated**: Where individuals only compare to those within specified groups.

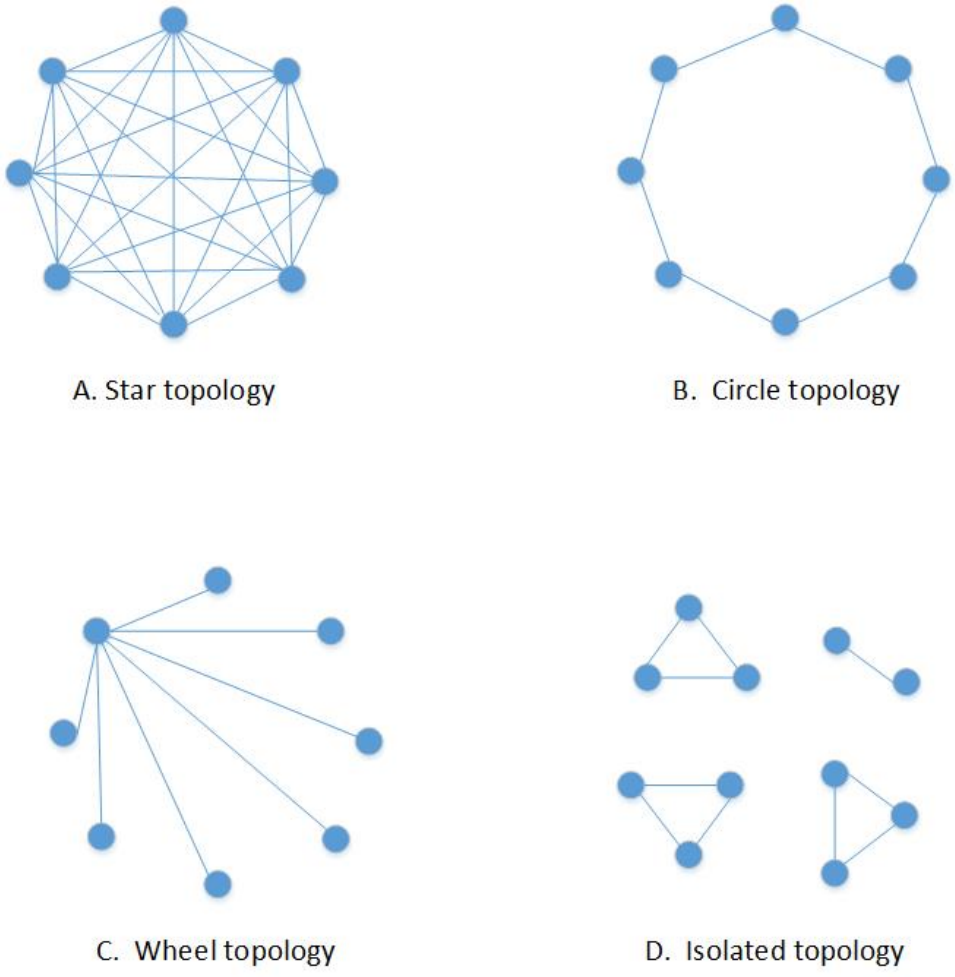


Fig. 2.1 A few of swarm communication topology types

The star or *gbest* topology shown in Fig.2.1(A) links every particle with every other, so that the social source of influence is in fact the best-performing member of the swarm.

In the circle topology or or *lbest* topology shown in Fig.2.1(B), which is a regular ring lattice as studied by Watts and Strogetz [93], parts of the swarm that are distant from one another are also independent of one another. Thus one segment of the population might converge on a local optimum, while another segment converges on a different optimum or keeps searching. More recent research has revealed that *lbest* swarm return improved results across various standard problem sets when used in conjunction with other improvements to the algorithm [49].

The wheel topology (See figure 2.1(C)) effectively isolates particles from one another, as all information has to be communicated through the focal particle. This focal particle compares solutions of all particles in the neighborhood, and adjusts result in improvement in the focal individual's performance. If adjustments result in improvement in the focal particle's performance, then that improvement is communicated out to the rest of the swarm.

The isolated topology (shown in Fig.2.1(D)) creates islands, or disconnected groups of individuals, which may collaborate among themselves to optimize the function. This would introduce a diminishing of communication, as the isolated particles would not get the information about the best solution found by the population; nor would the rest of the population benefit from their improved performances.

The choice of communication topologies used has been a matter of individual artistry, with some lore and little data to help the researcher choose a strategy. Some references show that highly connected particle swarm might not be as effective at finding optima in a problem space, compared to moderately connected communication topologies.

2.1.2 Definitions and Variables

Each of the particles is depicted by its position vector $\boldsymbol{x}$ and "flying" velocity $\boldsymbol{v}$. The particle adjusts its position according to its own flying experience and its neigborhood' flying experience. The equations for updating the position and velocity of each particle i are the

following[21]:

$$\begin{cases} \boldsymbol{v}_i(t) = \varpi * \boldsymbol{v}_i(t-1) + \varphi_1 * (\boldsymbol{p}_i(t-1) - \boldsymbol{x}_i(t-1)) + \varphi_2 * (\boldsymbol{p}_g(t-1) - \boldsymbol{x}_i(t-1)) \\ \boldsymbol{x}_i(t) = \boldsymbol{x}_i(t-1) + \boldsymbol{v}_i(t) \end{cases} \tag{2.1}$$

where t means the current iteration, $t-1$ means the previous iteration. The parameter ϖ is the linear decreasing inertial weight, which decreases from ϖ_{max} to ϖ_{min} during the iterations. The parameters φ_1 and φ_2 are acceleration coefficients. $\boldsymbol{p}_i$ is the best historical position particle i has found in the search space and $\boldsymbol{p}_g$ is the best position found by any neighbor of the particle. The particle's new velocity is calculated according to its previous velocity and the distances of its current position from its own best experience (position) and the group's best experience. Maintaining its own experience is defined as the "cognition" of a particle, which represents the private thinking of the particle itself. Taking advantage of the group's best experience represents the social collaboration among the particles. The fast social collaboration between particles seems to be the reason for clustering of particles because all the particles will tend to move toward the same position, that is, the search area is contracting through the generations. If the global optimum is out of the current search space, then there is little chance for SPSO to find the global optimum.

Another method of balancing global and local searches known as constriction was being explored simultaneously with the inertia weight method [16]. This method introduced a new parameter χ, known as the constriction factor, which is derived from the existing constants in the velocity update equation:

$$\chi = \frac{2}{\mid 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \mid}, \quad \varphi = c_1 + c_2 \tag{2.2}$$

In order to ensure convergence, the values $\chi \approx 0.72984$ and $c_1 = c_2 = 2.05$ are obtained. This constriction factor is applied to the entire velocity update equation:

$$\boldsymbol{v}_i(t) = \chi * (\boldsymbol{v}_i(t-1) + c_1 * \epsilon_1 * (\boldsymbol{p}_i(t-1) - \boldsymbol{x}_i(t-1)) + c_2 * \epsilon_2 * (\boldsymbol{p}_g(t-1) - \boldsymbol{x}_i(t-1))) \quad (2.3)$$

where, ϵ_1 and ϵ_2 are independent random numbers uniquely generated at every update for each individual dimension in the range [0,1] according to [16]. The parameter values noted above are preferred in most cases when using constriction for SPSO due to the proof of stability.

Similar, the position of each particle in every iteration by the following equation:

$$\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t-1) + \boldsymbol{v}_i(t) \quad (2.4)$$

2.2 Pseudocode of SPSO

Having presented the basic concept and definitions and variables used in SPSO, the main function of SPSO is described as follows.

Algorithm 1 Main function of SPSO

Initialization: $c_1 = c_2 = 2.05$, $\chi = 0.7298$, population n
Initialize each particle's position $\boldsymbol{x}_i$: $\boldsymbol{x}_i = rand() \in (\boldsymbol{x}_{min}, \boldsymbol{x}_{max})$, $i \in [1..n]$
Initialize personal best position $\boldsymbol{p}_i$: $\boldsymbol{p}_i = \boldsymbol{x}_i$, $Fitness(\boldsymbol{p}_i) = Fitness(\boldsymbol{x}_i)$
Find the best position $\boldsymbol{p}_g$ from all of neighbors: $\boldsymbol{p}_g = argmin(Fitness(\boldsymbol{p}_i))$
Initialization: $\boldsymbol{v}_i = 0.5 * rand()$; $rand() \in (\boldsymbol{x}_{min}, \boldsymbol{x}_{max})$
Repeat
for all particles $i \in [1..n]$ **do**
  Generate: $\psi_1 = rand() \in [0, 1]$; $\psi_2 = rand() \in [0, 1]$
  Update particle $\boldsymbol{x}_i$ with equation (2.3) and (2.4).
  Calculate fitness value of each updated particle $Fitness(\boldsymbol{x}_i^*)$
  if $Fitness(\boldsymbol{x}_i^*) < Fitness(\boldsymbol{p}_i)$ **then**
    $\boldsymbol{p}_i = \boldsymbol{x}_i^*$, $Fitness(\boldsymbol{p}_i) = Fitness(\boldsymbol{x}_i^*)$
  end if
end for
Find the new $\boldsymbol{p}_g^*$: $\boldsymbol{p}_g^* = argmin(Fitness(\boldsymbol{p}_i))$
Until maximum iterations are attained

In Algorithm 1, for each iteration, the number of Fitness Evaluations (FEs) is $n * D$, where n is the population size and D is the dimension of the problem. Apart from the FEs, which is problem dependent [39], [57], the main computational cost in SPSO is to find the global best optimum $\boldsymbol{p}_g$ from all of the $\boldsymbol{p}_i$, which is an inevitable operation in most swarm or population based evolutionary algorithms [40], [85]. Consequently, the computational complexity of SPSO is $O(nDT)$ (T is the maximum iteration number of SPSO).

2.3 Variants of Particle Swarm Optimization

The original PSO has undergone a number of changes since it was first proposed. In the SPSO, the behavior of particle swarm depends on at least three factors:

* *population topology:* Which defines the neighborhood relations among particles.
* *model of influence:* Which defines the mechanism to select, from each particle's neighbors, the set of individuals that act as informers.
* *update rule:* That is used to compute the next position of particle using information from its informers.

Different settings for the population topology, the model of influence or the update rule give rise to different PSO algorithms. In the following subsections, we briefly describe some of the most important developments. For a more detailed description of many of the existing particle swarm optimization variants, see ([47], [25], [15] and [75]).

2.3.1 Fully Informed Particle Swarm (FIPS)

The most salient example of *model of influence* is Mendes' *fully-informed* model, in which a particle uses information from all its neighbors, rather than just the best one. In this new

version, the performance of the Fully Informed Particle Swarm algorithm (FIPS) is conceptually more concise and promises to perform more effectively than the traditional particle swarm algorithm.

When constriction coefficient χ is implemented as in SPSO, a condensed form of the constricted SPSO can be implemented as follows:

$$\begin{cases} \boldsymbol{v}_{t+1} = \chi * (\boldsymbol{v}_t + \varphi * (\boldsymbol{p}_m - \boldsymbol{x}_t)) \\ \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{v}_{t+1} \end{cases} \tag{2.5}$$

which was then expanded to partition the acceleration weight φ between the particle's own previous success $\boldsymbol{p}_i$ and the neighborhood's $\boldsymbol{p}_g$, such that $\varphi = \varphi_1 + \varphi_2$. Note that $\boldsymbol{p}_m$ in this deterministic model is calculated as $\boldsymbol{p}_m = (\varphi_1 * \boldsymbol{p}_i + \varphi_2 * \boldsymbol{p}_g)/(\varphi_1 + \varphi_2)$. The search of particle converges on a point $\boldsymbol{p}_m$ in the search space. An alternate form of calculating $\boldsymbol{p}_m$ proposed in FIPS is:

$$\varphi_k = U[0, \frac{\varphi_{max}}{\mid N \mid}], \forall k \in N \tag{2.6}$$

$$p_m = \frac{\Sigma_{k \in N} W(k) \varphi_k \cdot p_k}{\Sigma_{k \in N} W(k) \varphi_k} \tag{2.7}$$

where $U[min, max]$ is a function that returns a vector whose positions are randomly generated following the uniform distribution between min and max, N is the set of neighbors of the particle and φ_k are parameters called *acceleration coefficients*.

In equation (2.7), $\boldsymbol{p}_k$ is the best position found by individual k. The function W may describe any aspect of the particle that is hypothesized to be relevant [63]. For instance, we use the fitness of the best position found by the particle, and the distance from that particle to the current individual, or have return a constant value. We note that the individual does not influence itself in this version. Other models of influence, such as choosing informer from

the neighbors at random [45] or with a a probability proportional to their "attractiveness" have also been proposed in [63].

2.3.2 Bare Bones Particle Swarms Optimization (BBPSO)

The Bare Bones Particle Swarms Optimization (BBPSO) [45] is a version of the particle swarm optimization algorithm in which the velocity and position update rules are substituted by a procedure that samples a parametric probability density function. References [43], [47] and [16] suggest that a particle's trajectory can be described as a cyclic path centred around a randomly weighted mean of the individual's $\boldsymbol{p}_i$ and the best neighbor's previous best points $\boldsymbol{p}_g$ on each dimension. In the bare-bones particle swarm optimization algorithm, a particle's position update rule as follows:

$$\boldsymbol{x}_i = N(\frac{\boldsymbol{p}_i + \boldsymbol{p}_g}{2}, \mid \boldsymbol{p}_i - \boldsymbol{p}_g \mid) \tag{2.8}$$

where $N(\mu, \sigma)$ represents a number drawn from a normal distribution with mean μ $((\boldsymbol{p}_i + \boldsymbol{p}_g)/2)$ and standard deviation σ $(\mid \boldsymbol{p}_i - \boldsymbol{p}_g \mid)$. Note that the barebones PSO facilitates initial exploration, due to large deviations (initially, personal best positions will be far from the informer's best position). As the number of iterations increases, the deviation approaches zero, focussing on exploitation of the average of the personal best and the informer's best positions. Thus, Kennedy modified the barebones equation to improve its exploration abilities. The new update rule is:

$$\boldsymbol{x}_i = \begin{cases} N(\mu, \sigma) & \text{if } U(0,1) < 0.5 \\ p_i & \text{otherwise} \end{cases} \tag{2.9}$$

where $U(0, 1)$ is a function that generates uniformly distributed random numbers between zero and one. Based on the equation (2.9), there is a 50% chance that the particle $\boldsymbol{x}_i$

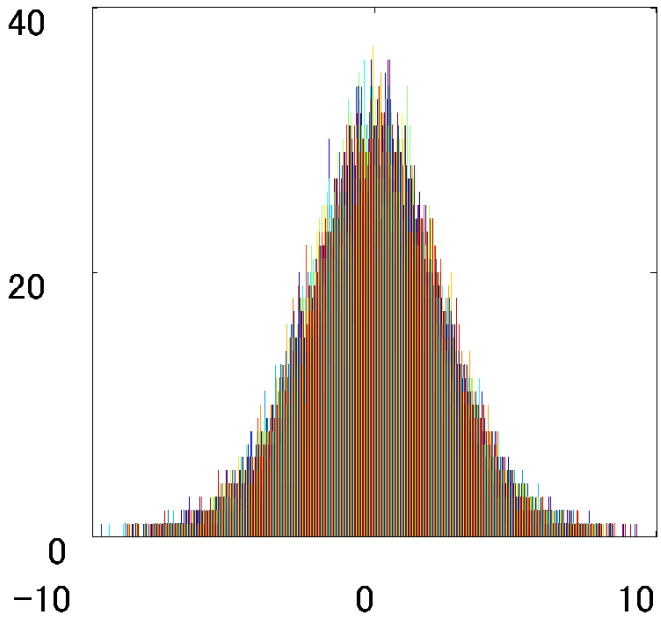


Fig. 2.2 Histogram of points sampled by BBPSO

changes to the corresponding personal best position. This version of BBPSO biases towards exploiting personal best positions, referred to as BBExp [45].

Figure 2.2 shows a histogram of points that were randomly sampled from a normal distribution of BBPSO. Where p_i and p_g were held constant at 1.0 and -1.0 respectively. The Fig.2.2 is a rough bell curve centred midway between the two previous best points and extending symmetrically beyond them. Figure 2.3 shows the positions of 200 particles on one iteration that were randomly sampled by BBPSO in the solution space of a two-Dimensional Ackley function. The versions of BBPSO differed in how g was defined: $gbest$, neighborhood best, or random neighbor. The performance of the random-neighbor version is interesting. Based on the experimental results in [45], the best-performing algorithm was the modified barebones version where g was a randomly selected neighbor.

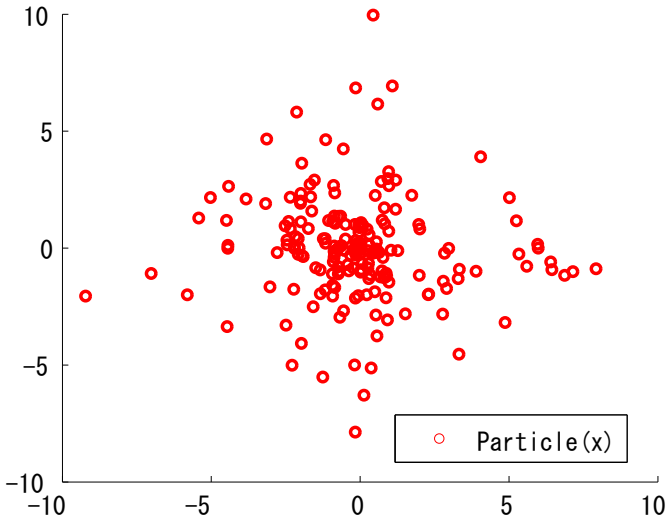


Fig. 2.3 Positions of particles sampled by BBPSO on Ackley function

2.3.3 Binary Particle Swarm Optimization (BBPSO)

Most particle swarm optimization algorithms are designed to search in continuous domains. However, there are a number of variants that operate in discrete spaces. The first variant proposed for discrete domains was the binary particle swarm optimization algorithm [46]. In the binary version, trajectories are changes in the probability that a coordinate will take on a zero or one value. It means that a particle moves in a state space restricted to 0 and 1 on each dimension d, where each v_{id} represents the probability of bit x_{id} taking the value 1. In other words, if $v_{id} = 0.20$, then there is a 20% chance that x_{id} will be the value 1, and an 80% chance it will be the value 0. In sum, the particle swarm formula remains unchanged, except that now p_i and x_i are integers in 0, 1 and v_i, since it is a probability, must be constrained to the interval [0.0, 1.0]. A logistic transformation $S(v_{id})$ can be used to accomplish this last modification. In [51], the sigmoid function used in the binary version is:

$$S(v_{id}) = \frac{1}{1 + e^{-v_{id}}} \tag{2.10}$$

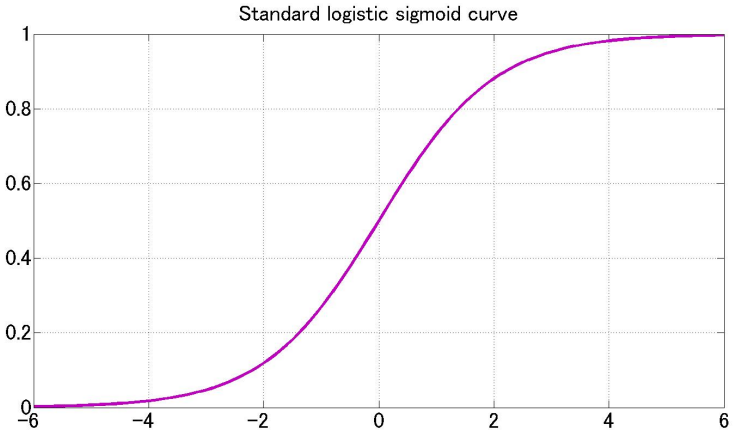


Fig. 2.4 The standard logistic sigmoid curve

Also the equation (2.10) is used to update the velocity vector of the particle. And the new position of the particle is obtained using the equation below [46]:

$$x_{id} = \begin{cases} 1 & \text{if } rand() < S(v_{id}) \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

where $rand()$ is a quasirandom number selected from a uniform distribution in [0.0, 1.0].

In the present discrete version, it appears that with v_{id} functioning as a probability threshold, changes in v_{id} might represent a change in first-order position itself. As $S(v_{id})$ approaches zero, for instance, the position of the particle fixes more probably on the value 0, with less chance of change. Trajectory in the current model is probabilistic, and velocity on a single dimension is the probability that a bit will change. Thus even if v_{id} should remain fixed, the position of the particle on that dimension remains dynamic as it flips polarity with probability following from the value of v_{id}. According to [46] and [51], the binary PSO performs especially well on number of test problems. It also appears that the binary particle swarm is extremely flexible and robust. The binary PSO can be used in variety of applications, especially when the values of the search space are discrete like decision mak-

ing, solving lot sizing problem [88], the travelling salesman problem [107], scheduling and routing [71], [72].

2.4 Strength and Weakness

2.4.1 Strength

PSO is a population based and intelligent method, which is inspired by the emergent motion of a flock of birds searching for food. In PSO, a population of potential solutions is evolved through successive iterations. Compared to other optimization strategies, it can be easily implemented and it is computationally inexpensive, since its memory and CPU speed requirements are low [20]. Moreover, it does not require gradient information of the objective function under consideration, but only its values, and it uses only primitive mathematical operators. PSO has been proved to be an efficient method for many global optimization problems and in some cases it does not suffer the difficulties encountered by other EC techniques [21]. Since PSO algorithm has a number of desirable properties, including simplicity of implementation, scalability in dimension, and good empirical performance, it has been applied to solve many real-world problems, such as capacitor placement problem [69], short term load forecasting [90], soft sensor [91], the voltage stability of the electric power distribution systems [66], [24], the orbits of discrete chaotic dynamical systems towards desired target region [58], and the permutation flow-shop sequencing problem [88].

2.4.2 Weakness

Perhaps the most common problem encountered by many Global Optimization (GO) methods, either deterministic or stochastic, when coping with the GO problem is the problem of local minima. Especially in multimodal functions, the existence of many local minima makes it quite difficult for most techniques to detect the global minimum. PSO, despite

being an efficient method, also suffers from this problem. In order to avoid being stuck in local optima in the convergence process, some improved PSO have been proposed, such as crossover [11], orthogonal learning strategy [104], chaos [1], and elitist learning strategy [103].

Among population-based algorithms, PSO also has difficulties in keeping balance between exploration and exploitation when solving complex multimodal problems. For instance, all particles share its swarm's best experience (the global best) that can lead the particles to cluster around the global best. In case, if the global best is located near a local minimum, escaping from local optimum becomes difficult and PSO suffers diversity loss near the local minimum [79]. In order to address the exploration and exploitation trade-off problem, some other improved PSO have been proposed. Liang proposed comprehensive learning particle swarm Optimization (CLPSO) in which each particle learns from other particles' best experiences for different dimensions via a comprehensive learning strategy [56]. Efficient population utilization strategy for particle swarm optimization (EPUS-PSO) was presented in [36]. In which population size is varied by a population manager according to the status of the solution search. Meanwhile, heterogeneous particle swarm optimization was proposed in [18], [26]. In heterogeneous PSO (HPSO) [26], the particles in heterogeneous swarms were allowed to follow different velocity and position updating rules from a behavior pool, thereby having the ability to explore and exploit throughout the problem search space.

Chapter 3

Fitness Predator Optimization for Multimodal Problems

A major problem with most of swarm intelligent algorithms in multimodal optimization is Premature Convergence (PC), which results in significant performance loss and sub-optimal solutions. To avoid premature convergence by maintaining diversity in the population, many kinds of optimization algorithms are proposed. However, to the best of our knowledge, few of the swarm intelligent techniques focus on the individual competition. The development of individual competition plays an important role of the diversity conservation in the population because it could increase individual independent consciousness and reduce the rapid social collaboration process. In this chapter, a new algorithm, Fitness Predator Optimization (FPO), is proposed based on the conceptions of predators to avoid premature convergence for multimodal optimization problems.

3.1 Overview and Preliminaries

In contrast to the unimodal functions, multimodal functions have many local optima with the number increasing exponentially with dimension. This makes them fairly difficult to

convergence to the global minimum. It is suitable for benchmarking the global search ability and the local optima avoidance of an algorithm. In order to solve the multimodal, nonlinear, discontinuous and non-differentiable optimization problems, researchers have developed population-based algorithms such as Particle Swarm Optimization (PSO), [21], Ant Colony Optimization (ACO) [19], Artificial Bee Colony (ABC) [41] and so on. To the Particle Swarm Optimization (PSO), all particles share its global best experience that can lead the particles to cluster around the global best. In case, if the global best is located near a local minimum, escaping from local optimum becomes difficult. Diversity declines rapidly in the later iteration period, leaving the PSO algorithm with great difficulties of escaping local optima. Consequently, the clustering particles with fitness stagnation further exacerbates the premature convergence situation. An accepted hypothesis is that maintenance of high diversity is crucial for preventing premature convergence in multimodal optimization.

3.1.1 Predator-Prey Optimization (PPO)

Many kinds of optimization algorithms are proposed to improve the diversity of the population. Some of them are inspired by the social behavior of swarms, herds in nature. In addition, the hunting and search behaviors of predator are implemented by more and more researchers and proved to be an effective method. For example, the basic idea of Artificial Fish-Swarm Algorithm (AFSA) [53] is to imitate fish behavior such as preying, swarming, following with local search of individual fish for reaching the global optimum. The Grey Wolf Optimization (GWO) [64] algorithm mimics the leadership hierarchy and hunting mechanism of grey wolves in nature. Other related swarm intelligent algorithms, such as Predator-Prey Optimization (PPO) [83] in 2006, Dolphin Partner Optimization (DPO) [80] in 2009, Bat-inspired Algorithm (BA) [102] in 2010, Krill Herd (KH) [28] in 2012 are also proposed to simulate group hunting behaviors.

The most closely associated algorithm with our proposed method is the Predator-Prey Optimization (PPO) [83]. It is a form of particle swarm optimization where new particles called predators are introduced. The predator's objective is to pursue the best individual in the swarm. The influence of the predator on any individual in the swarm is controlled by a "fear" probability P_f, which is the probability of a particle changing its velocity due to the presence of the predator. That is, if all of the particles tend to move toward the best global particle, a predator particle nearby will disturb these particles' flying velocities to expend the search space. In PPO, the predator update equations are [83]:

$$\begin{cases} \boldsymbol{v}_p(t) = \varphi_4 * (\boldsymbol{x}_g(t-1) - \boldsymbol{x}_p(t-1)) \\ \boldsymbol{x}_p(t) = \boldsymbol{x}_p(t-1) + \boldsymbol{v}_p(t) \end{cases} \tag{3.1}$$

The φ_4 is a positive constant and $\boldsymbol{x}_g$ is the present position of the best particle in the swarm. The rule of predator influence on the velocity of prey is:

$$\begin{cases} v_{ij}(t) = \varpi * v_{ij}(t-1) + \varphi_{1ij} * (p_{ij} - x_{ij}(t-1)) \\ \qquad\qquad +\varphi_{2ij} * (p_{gj} - x_{ij}(t-1)) + \varphi_{3ij} * ae^{-bd} \\ x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t) \end{cases} \tag{3.2}$$

The parameter ϖ is the linear decreasing weight, which decreases from ϖ_{max} to ϖ_{min} during the iterations. v_{ij} represents a particle i's velocity in the dimension j. φ_{1ij}, φ_{2ij} and φ_{3ij} are positive constants. $Z(x)$ is an exponential decreasing distance function defined as:

$$Z(x) = ae^{-bx} \tag{3.3}$$

a represents the maximum amplitude of the predator effect over a prey and b permits to control the distance at which the effect is still significant. The experimental results show

that the PPO has a better performance than the PSO in the multimodal functions, but only by a very small margin. In addition, the parameters of the PPO were empirically defined, which greatly influence the robust performance of the algorithm.

3.1.2 Solutions

Few of optimization algorithms focus on individual competition and independent self awareness. The individual competition is more likely to reduce the rapid social collaboration process and increase the ability of being out of the local optimum. This motivated our attempt to present a new swarm intelligence algorithm, called Fitness Predator Optimization (FPO). First of all, the individual competition is introduced in the FPO to maintain the diversity of the population. Secondly, according to the survival skills of predators, four kinds of searching rules are defined in the FPO. The most important principle is that only the competitive, powerful positions selected as elites could achieve the limited opportunity to update. The characteristic of the elite induces the communication among of the population, making clustering of all the positions difficult. Meanwhile, the defined individual competition strategy forces each elite try every means to aggressively occupy another elite's position, which greatly stimulates the competition among the elite team, and strengthening the ability of the elite to find the best global optimal solution. Eight well-known benchmark functions are used to test the performance of FPO. These experimental results demonstrate the effectiveness of the proposed FPO algorithm. The rest of this chapter is organized as follows. In Section 3.2, basic concepts and the pseudocode of FPO is given, and a method for the parameters selection is presented to reinforce the convergent performance. Experiment results are given in Section 3.3 to verify the efficiency of the FPO. Finally, some concluding remarks are provided in Section 3.4.

3.2 Proposal of Fitness Predator Optimization (FPO)

3.2.1 Basic Concept of FPO

The survival of the fittest is what Charles Darwin has called "natural selection", or the preservation of favored races in the struggle for life. Pressure from Natural selection forces the animals to develop highly optimized organs and skills to take advantage in the fight for food, territory and mates. Some of the organs and skills can be further refined as optimization algorithms, which are effective methods for inspiration to develop intelligent systems and provide solutions to complicated problems. From the ecological viewpoint, the survival abilities of predators can be summarized as four aspects [84]:

- Its power of locomotion
- Its power of perception
- Its power of survival
- Its aggressiveness and persistence

According to these survival skills of predators, four kinds of searching rules are defined in the FPO.

- The better locomotion it has, the more possibility to move on next favorable locomotion.
- The power of perception to find the best way to improve its position.
- The limited opportunities for positions, who access the most opportunities will be the ultimate survivor.
- The territory intrusion is to take advantage of a companion's position information.

All of the individuals in the FPO are defined as predators, the predator's purpose is to find the global optimum (seen as prey) in the search space. Each individual is depicted only by its position vector x, which determines the trajectory of the particle. Then an individual is named as a "position" which comprises the population in FPO. If the position i has a higher value of fitness function, then i has more power of locomotion. If the position i does not know what is the next best place, a sensible way is to dynamically adjust it according to its own experience and its companions' experience. The definition of updated position is:

$$newx_{ij} = x_{ij} + (rand() - 0.5) * c * w * (x_{kj} - x_{ij}) \tag{3.4}$$

To the position i, $\boldsymbol{x}_i = (x_{i1}, x_{i2}, ..., x_{id})$ is a vector with d dimensions. k ($k \neq i$) is a companion of position i randomly chosen from the population. $newx_{ij}$ is the updated position on j^{th} dimension. c is a positive constant, w is defined as inertia weight, first proposed by Y.Shi and Eberhart [78] for PSO. A position with a better value of fitness function having a prior possibility to update is called elite. It's worth noting that the elite is not the group's best one, many positions could be the elites when they receive an update in the previous iteration. It drastically reduced the possibility of all the particles moving toward the same position. For all positions, there are only limited opportunities to update their place. For example, ρ number of chances are released in each iteration, only the competitive position could get the chance to update by (3.4). The remaining of positions keep the previous positions until they get a chance to update. In the later stage of optimization, according to decreasing search space, elites will try every means to aggressively occupy companion's territory until final survival determines the best global optimal solution.

To summarize, in order to avoid premature convergence some new advantages are presented in FPO. First of all, all of the particles in the FPO are predators, the characteristic of the predator is more inclined to competition instead of social collaboration. This competition personality induces the communication among of the population, making clustering of all

the positions difficult. Secondly, there are a number of positions composed of an elite team. It also reduces the possibility of all the particles moving toward the same position. Finally, only a limited number of chances are released in each iteration, which greatly stimulates the competition among the elite team, strengthening the ability of the elite to find the best global optimal solution.

3.2.2 Pseudocode of FPO

Having presented four basic principles, the main function of FPO is described as follows.

Algorithm 2 Main function of FPO

Initialize population $popsize$: $popsize = n$
Initialize chances ρ: $\rho = 0.5 * n$
Initialize $\boldsymbol{x}_i$: $\boldsymbol{x}_i = rand() \in (\boldsymbol{x}_{min}, \boldsymbol{x}_{max})$
Repeat
$s = 0$
while $s < \rho$ **do**
  for $i \in [1..n]$ **do**
    Calculate P_i by equations (3.5), (3.6)
    if $rand() < P_i$ **then**
      Put $\boldsymbol{x}_i$ into an elite team
      Update $\boldsymbol{x}_i$ with territory intrusion
      For each position use the elitism strategy
      $s = s + 1$
      if $s == \rho$ **then** break
      end if
    end if
  end for
end while
For the population use the elitism strategy
Until maximum iterations are attained

In algorithm 2, $popsize$ denotes the population of particles, the amount of chances ρ equals to half of the $popsize$ on each iteration. The fitness function is defined as:

$$fitness(\boldsymbol{x}_i) = exp(-\beta f(\boldsymbol{x}_i)) \tag{3.5}$$

The possibility of position i, P_i is defined as:

$$P_i = rand() * \frac{fitness(\boldsymbol{x}_i)}{\sum_{i=1}^{n} fitness(\boldsymbol{x}_i)} \tag{3.6}$$

Where, $\boldsymbol{x}_i$ denotes its current position, $f(\boldsymbol{x}_i)$ denotes the value of function in the current position. β is the adjustment factor. If the value of β decreases, the selection possibility of the higher value of fitness function will be increased. In our experiments, β is set to 2 for all of the test benchmark functions. In (3.6), $rand()$ is a quasirandom number selected from a uniform distribution in [0.0, 1.0]. It is important to add a random factor in this equation. By this way, position i with a less than ideal place still has an opportunity to move to a better one in the preliminary search stage.

The territory intrusion function is shown in Algorithm 3.

Algorithm 3 Territory intrusion function

1: $k = ceil(rand() * popsize)$
2: $j = ceil(rand() * dimension)$
3: **while** $k == i$ **do**
4: $\quad k = ceil(rand() * popsize)$
5: **end while**
6: $\theta = rand() - 0.5$
7: $newx_{ij} = x_{ij} + \theta * c * w * (x_{kj} - x_{ij})$

Generally, $c \in [0, 2]$ and w is linear regulated in the process of convergence. $k, (k \neq i)$ is a companion of position i.

The elitism strategy of FPO is to reserve the best optimal position as shown in (3.7).

$$f_{l+1}^{*} = \begin{cases} f(X) & \text{if } f(X) < f_l^{*} \\ f_l^{*} & \text{otherwise} \end{cases} \tag{3.7}$$

The f_l^* is the best optimal position fitness value after l times comparison with other positions, X is the new position which will be compared with f_l^* in the $(l+1)th$ time.

Algorithm 4 The elitism strategy function

$GlobalMin = min(f(\boldsymbol{x}_i))$
$newx_{ij} = x_{ij} + \theta * c * w * (x_{kj} - x_{ij})$
$tmp = min(f(newx_{ij}))$
if $tmp < GlobalMin$ **then**
  $GlobalMin = tmp$
end if

3.2.3 A Method for Parameter Control

Yuhui Shi and Russell Eberhart [78] point out that it is a good idea for an optimization search algorithm to possess more exploitation ability at the beginning in order to find a good seed then have more exploration ability to narrow the search of the local area around the seed later. In order to increase the opportunity of exploitation of good candidate regions for positions, an inertia weight ϖ is proposed in FPO. Accordingly, we defined the inertia weight ϖ as a decreasing function of time instead of a fixed constant [78].

$$\varpi(t) = m - (m - n) * \frac{t}{t_{max}} \tag{3.8}$$

where, t is the current iteration, t_{max} is the maximum number of iterations, and ϖ within the range $[n, m]$. In order to find a proper parameters setting, three widely known benchmark functions (Rosenbrock, Rastrigin and Griewank) are used to the parameter setting experiment. The FPO algorithm was run 50 times independent trials with a population size of 20 on each benchmark function. The collecting experimental results shown in the Table 3.1 are the averaged optimum solutions on 30 dimensions of each benchmark function with the FPO algorithm in 1000 times generation (iteration). In this experiment, the parameter c is first fixed to 2 and the value of ϖ decreasing linearly within different ranges. Then ϖ is set

Table 3.1 Experiment results for parameters setting in FPO

$\varpi(c = 2)$	$1.2 \rightarrow 0.6$	$0.9 \rightarrow 0.4$	$0.6 \rightarrow 0.2$
Rosenbrock	**8.50e+01**	1.19e+02	6.45e+01
Rastrigin	**1.99e-01**	9.95e-01	1.99e+00
Griewank	**7.42e-04**	6.92e-06	5.08e-05
$c(\varpi = 1)$	2.5	2	1
Rosenbrock	1.58e+02	**1.13e+02**	1.05e+02
Rastrigin	7.59e-08	**2.71e-07**	1.83e+00
Griewank	4.48e-09	**1.80e-09**	7.60e-04

to 1 and c are fixed at constants which are shown in Table 3.1. From Table 3.1, it can be observed that the FPO generated the better performance on three functions when the value of c was fixed at 2 and ϖ was varying from 1.2 to 0.6. In the next experiment section, ϖ is fixed from 1.2 to 0.4 and $c = 2$ for all of test benchmark problems. However, it must be said that the given method of parameter selection is a trial.

3.3 Experiments

In this section, we will perform a set of experiments conducted on the eight well-known benchmark functions used as performance test problems for all of experimental optimization algorithms. In order to verify the performance of FPO, it has been compared with a number of the state-of-the-art algorithms. The compared algorithms include Simple Genetic Algorithm(SGA) [29], Standard Particle Swarm Optimization (SPSO), Quantum-behaved PSO (QPSO) and Predator-Prey Optimization (PPO).

3.3.1 Evaluation Method

The test environment and experimental execution parameters are shown in Table 3.2 and Table 3.3 respectively. In order to ensure convergence, the inertia weight ϖ in the range [0.4, 0.9] are preferred in most cases of the SPSO algorithm [78]. The bounds and global minimums for all benchmark functions can be found in Table 3.4. All swarms were randomly

Table 3.2 Evaluation test environment

OS	Windows 7
Processor	Intel(R) $Core^{TM}$ i7 CPU 2.80GHz
Memory (RAM)	8.00GB
System type	64-bit operation system
Tool	MATLAB 7.10.0

Table 3.3 Experimental execution parameters

Experiment	Algorithm	Parameters
Experiment A	SPSO	$c1 = c2 = 2, \varpi \in [0.4, 0.9]$
Population=20	PPO	$\varphi_1 = \varphi_2 = 2, \varphi_3 = 1, \varphi_4 = 0.1$
Run Nummber=200		$a \in [0.1X_{max}, 2X_{max}]$
		$b = 10.0/X_{max}, P_f \in [0.001, 0.06]$
	FPO	$C = \beta = 2, \varpi \in [0.4, 0.9]$
Experiment B	SPSO	$c1 = c2 = 2, \varpi \in [0.4, 1.2]$
Population=100	SGA	$cross = 0.8, mutation = 0.1$
Run Nummber=50	QPSO	$\beta \in [0.5, 1]$
	FPO	$c = \beta = 2, \varpi \in [0.4, 1.2]$

initialized in an area equal to the feasible search space in every dimension. In experiment A, four multimodal benchmark functions are used to test the performance of FPO compared with PSO and PPO. In experiment B, four fixed-dimension multimodal optimization problems are selected to make a comparison of the convergence rate between FPO, SGA, SPSO and QPSO algorithms.

3.3.2 Experiment on Multimodal Problems

In experiment A, four typical multimodal benchmark functions are used to test the performance of FPO comparing with SPSO and PPO. Each algorithm was run on an array of common benchmarks, shown in Table 3.4, for 1000, 1500 and 2000 evaluations per function. Performance was measured as the minimum error $\mid f(x) - f^*(x) \mid$ found over the trial, where $f^*(x)$ is the optimum fitness for the problem. Results were averaged over 200 independent trials, and are displayed in Table 3.5 and Table 3.6. The data of SPSO column and the PPO column are adopted from [83]. The FPO algorithm was run 200 times with a

Table 3.4 Bounds and global optimums of benchmark functions

Function	Bound	Optimum
Rosenbrock: $f_1(x) = \sum_{j=1}^{D}(100 * (x_{j+1} - x_j^2)^2 + (x_j - 1)^2)$	$[-100, 100]^D$	0
Rastrigin: $f_2(x) = \sum_{j=1}^{D}(x_j^2 - 10 * \cos(2\pi x_j)) + 10 * D$	$[-5.12, 5.12]^D$	0
Griewank: $f_3(x) = \frac{1}{4000}\sum_{j=1}^{D} x_j^2 - \prod_{j=1}^{D}\cos(\frac{x_j}{\sqrt{j}}) + 1$	$[-5, 10]^D$	0
Ackley: $f_4(x) = -a * \exp(-0.02 * (D^{-1}\sum_{i=1}^{D-1} x_i^2)\frac{1}{2})$	$[-15, 30]^D$	0
$-\exp(D^{-1}\sum_{j=1}^{D}\cos(2\pi x_j)) + a + \exp, a = 20$		
Michalewicz: $f_5(x) = -\sum_{j=1}^{D} sin(x_j)[sin(\frac{j \cdot x_j^2}{\pi})]^{20}$	$[0, \pi]^2$	-1.8013
Levy: $f_6(x) = \sum_{j=1}^{D-1}(\varpi_j - 1)^2[1 + 10sin^2(\pi\varpi_j + 1)]$	$[-10, 10]^{30}$	
$+sin^2(\pi\varpi_1) + (\varpi_D - 1)^2[1 + sin^2(2\pi\varpi_D)]$		0
$\varpi_j = 1 + \frac{x_j - 1}{4}$		
Branin: $f_7(x) = a(x_2 - bx_1^2 + cx_1 - 6)^2$		
$+g(1 - h)\cos(x_1) + 10, a = 1$	$x_1 \in [-5, 10]$	0.397887
$b = 1.25\pi^{-2}, c = 5\pi^{-1}, g = 10, h = 0.125\pi^{-1}$	$x_2 \in [0, 15]$	
Shekel: $f_8(x) = -\sum_{i=1}^{m}(\sum_{k=1}^{4}(x_k - C_{ki})^2 + \beta_i)^{-1}$	$[0, 10]^{10}$	-10.5364
$m = 10, \beta = \frac{1}{10}(1, 2, 2, 4, 4, 6, 3, 7, 5, 5)^T$		
$C = \begin{pmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3 \\ 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3 \end{pmatrix}$		

Note: D represents the dimension of benchmark function.

Table 3.5 Experiment A: Statistical results on various dimensions of functions: $f_1 - f_2$

Fun.	Dim.	Iter.	Algorithm	Mean	Std.
f_1	10	1000	SPSO	1.19e+02	3.40e+01
			PPO	3.97e+01	1.36e+01
			FPO	**1.66e+01**↓	3.18e+01
f_1	20	1500	SPSO	1.85e+02	4.35e+01
			PPO	6.75e+01	1.50e+01
			FPO	**2.64e+01**↓	4.57e+01
f_1	30	2000	SPSO	2.41e+02	5.27e+01
			PPO	1.64e+02	3.77e+01
			FPO	**7.15e+01**↓	7.72e+01
f_2	10	1000	SPSO	5.27e+00	3.71e-01
			PPO	2.39e-01	7.61e-02
			FPO	**0.00e+00**↓	0.00e+00
f_2	20	1500	SPSO	2.32e+01	1.01e+00
			PPO	3.09e+00	3.90e-01
			FPO	**1.03e-13**↓	7.27e-13
f_2	30	2000	SPSO	4.86e+01	1.73e+00
			PPO	1.07e+01	9.31e-01
			FPO	**3.27e-13**↓	1.31e-12

Note: The result marked with down arrow denotes that it is the best average minimum error compared with others.

population size of 20 on each benchmark function. The statistical results including average minimum error and standard deviation are reported in Table 3.5 and Table 3.6.

It is worth noting that function f_2 (Rastrigin function) is a typical example of non-linear multimodal function used as a performance test problem for optimization algorithms. Finding the minimum of this function is a fairly difficult problem due to its large search space and its large number of local minima. From Table 3.5, it can be seen that FPO demonstrates the superior performance compared with SPSO and PPO. According to Table 3.5, the statistical result of the FPO on f_2 with 10 dimensions and 1000 iterations shows that FPO is able to escape the trap of the suboptimal values and to find the global minimum at $f_2(x^*) = 0$ where $x^* = 0$.

The left part of Fig.3.1 illustrates the 2-dimensional image of Rastrigin function. The right part shows the convergence curve of FPO on Rastrigin function with 10 dimensions

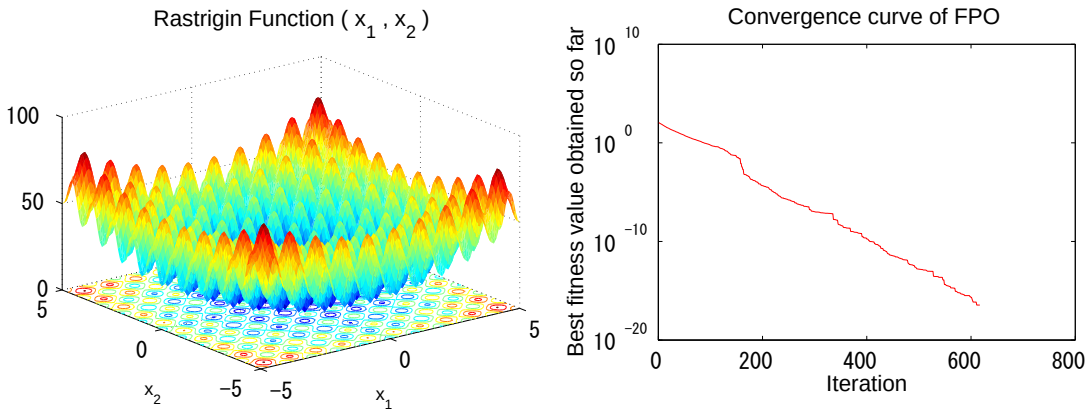


Fig. 3.1 Convergence curve of FPO on 10 dimensions of f_2 with 1000 iterations

and 1000 iterations. When the iterations over to 600, the convergence curve is suspended because the FPO has already found the global minimum $x^* = 0$.

Figure 3.2 shows the trajectory of the individual in FPO on 2-dimensional Rastrigin function. In the first iteration, all individuals are scattered in the search space. When the iterations equals to 50, most of particles gradually gathered around the global minimum $x^* = 0$. In the 120th iteration, FPO converge to the global optimum $f_2(x^*) = 0$ where $x^* = 0$. Figure 3.2 not only shows that FPO has superior performance on f_2 but also reveals that FPO might has a good speed of convergence rate.

The experiment results from f_1 to f_4 proved that FPO could find better global values than SPSO and PPO. The excellent global optimizing ability of FPO also reveals that it might have a good convergence rate compared with other population-based algorithms.

3.3.3 Experiment on Fixed-dimension Problems

In experiment B, four multimodal problems are selected to make a comparison of the convergence rate between FPO and GA, SPSO and QPSO. Table 3.7 and Table 3.8 shows the statistical results including the minimum error and the average of minimum error for each

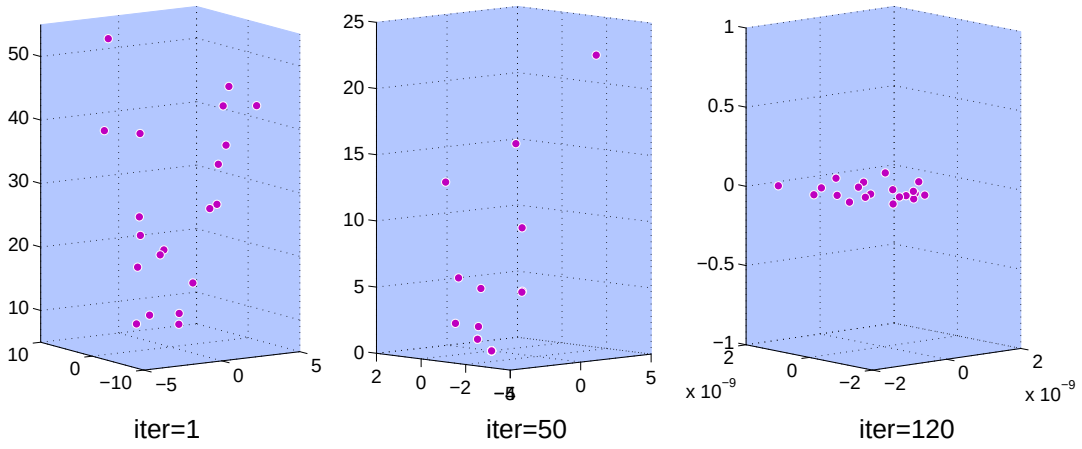


Fig. 3.2 The trajectory of the position of FPO on 2-dimensional Rastrigin function

Table 3.6 Experiment A: Statistical results on various dimensions of functions: $f_3 - f_4$

Fun.	Dim.	Iter.	Algorithm	Mean	Std.
f_3	10	1000	SPSO	9.82e-02	6.70e-03
			PPO	6.43e-02	4.30e-03
			FPO	**1.20e-02**↓	1.01e-02
f_3	20	1500	SPSO	2.84e-02	3.90e-03
			PPO	2.19e-02	3.00e-03
			FPO	**3.25e-15**↓	1.00e-14
f_3	30	2000	SPSO	1.59e-02	2.50e-03
			PPO	1.33e-02	3.00e-03
			FPO	**6.44e-16**↓	1.80e-15
f_4	10	1000	SPSO	2.56e-11	6.03e-12
			PPO	7.83e-08	1.24e-08
			FPO	**8.35e-15**↓	2.18e-15
f_4	20	1500	SPSO	8.20e-03	1.61e-02
			PPO	1.84e-06	2.69e-07
			FPO	**8.99e-14**↓	7.63e-14
f_4	30	2000	SPSO	2.11e-01	7.03e-02
			PPO	1.25e-05	1.71e-06
			FPO	**1.12e-12**↓	8.25e-13

Note: The result marked with down arrow denotes that it is the best average minimum error compared with others.

Table 3.7 Experiment B: Statistical results of fixed dimensions of functions: $f_5 - f_6$

Fun.	Dim.	Iter.	Algorithm	Best	Mean
f_5	2	20	SGA	0.00e-00	3.40e-03
			SPSO	3.00e-04	2.90e-03
			QPSO	0.00e-00	3.00e-04
			FPO	0.00e-00	**1.00e-04**↓
f_5	2	50	SGA	0.00e-00	2.60e-03
			SPSO	0.00e-00	0.00e-00
			QPSO	0.00e-00	0.00e-00
			FPO	0.00e-00	0.00e-00
f_5	2	100	SGA	0.00e-00	2.60e-03
			SPSO	0.00e-00	0.00e-00
			QPSO	0.00e-00	0.00e-00
			FPO	0.00e-00	0.00e-00
f_6	30	20	SGA	5.72e+01	8.62e+01
			SPSO	4.20e+01	6.49e+01
			QPSO	3.72e+00	5.77e+00
			FPO	**2.62e-02**↓	**8.30e-02**↓
f_6	30	50	SGA	1.44e+01	3.04e+01
			SPSO	1.13e+01	2.16e+01
			QPSO	6.52e-01	1.28e+00
			FPO	**2.50e-03**↓	**6.70e-03**↓
f_6	30	100	SGA	6.08e+00	1.63e+01
			SPSO	1.35e+00	3.76e+00
			QPSO	2.20e-03	1.43e-01
			FPO	**3.00e-04**↓	**9.77e-04**↓

Note: The result marked with down arrow denotes that it is the best minimum error or the best of average minimum error compared with others.

benchmark function with 20, 50 and 100 times iteration. All algorithms were run 50 times with a population size of 100 on each benchmark function.

3.3.4 Discussion

According to the results of experiment A, it is easy to see that FPO shows its particular global search ability on multimodal benchmark problems compared with SPSO and PPO. The results of experimentation on f_2 (Rastrigin function) are worth noticing. This function is a fairly difficult problem due to its large search space and its large number of local min-

Table 3.8 Experiment B: Statistical results of fixed dimensions of functions: $f_7 - f_8$

Fun.	Dim.	Iter.	Algorithm	Best	Mean
f_7	2	20	SGA	0.00e+00	4.19e-02
			SPSO	1.00e-04	1.77e-02
			QPSO	0.00e+00	**0.00e+00**↓
			FPO	0.00e+00	5.00e-04
f_7	2	50	SGA	0.00e+00	3.07e-02
			SPSO	0.00e+00	0.00e+00
			QPSO	0.00e+00	0.00e+00
			FPO	0.00e+00	0.00e+00
f_7	2	100	SGA	0.00e+00	2.96e-02
			SPSO	0.00e+00	0.00e+00
			QPSO	0.00e+00	0.00e+00
			FPO	0.00e+00	0.00e+00
f_8	10	20	SGA	7.13e+00	8.36e+00
			SPSO	1.09e+00	6.11e+00
			QPSO	1.521e-01	**4.51e+00**↓
			FPO	**2.09e-02**↓	6.67e+00
f_8	10	50	SGA	7.01e+00	8.08e+00
			SPSO	5.00e-04	3.04e+00
			QPSO	**0.00e+00**↓	3.55e+00
			FPO	7.50e-03	**2.94e+00**↓
f_8	10	100	SGA	6.98e+00	8.07e+00
			SPSO	**0.00e+00**↓	2.65e+00
			QPSO	**0.00e+00**↓	3.36e+00
			FPO	8.36e-04	**1.83e+00**↓

Note: The result marked with down arrow denotes that it is the best minimum error or the best of average minimum error compared with others.

ima. The results of the FPO in Rastrigin function with 10 dimensions and 1000 iterations shows that the FPO is able to escape the trap of the suboptimal values and to find the global minimum. The results of the FPO from f_5 to f_8 proved that FPO could find the best global optimum within 20 iterations. According to the results of Table 3.5, Table 3.6, Table 3.7 and 3.8, the FPO has superior performance in terms of exploiting the global optimum. This is due to the proposed four principles previously discussed. The higher convergence rate of FPO demonstrates that it has a good balance between exploitation and exploration that results in high local optima in avoidance. This superior capability is due to the adaptive inertia weight w in the position updated formula of FPO. The adaptive method of parameter control may lead to a more efficient algorithm and our future work is focusing on this problem.

3.4 Summary

In this chapter, a new algorithm, Fitness Predator Optimization (FPO), is proposed for multimodal problems based on the concept of predator. Pressure from natural selection forces the predators to develop highly optimized organs and skills to take advantage in the fight for food, territory and mates. Some of the organs and skills can be further refined as optimization algorithms, which are effective methods for inspiration to develop intelligent systems and provide solutions for complicated problems. In FPO, the survival abilities of predators are refined as four kinds of researching rules, which are ideal methods for inspiration to provide solutions to multimodal problems. In an FPO system, all of the individuals are seen as predators. Each of the individuals is depicted only by its position. Then the individual is named as a "position" in FPO. Only the competitive, powerful positions selected as elites could achieve the limited opportunities to update. It is worth noting that the elite is not the swarm's best one, many positions could be the elites when they receive an update in the previous iteration. It drastically reduced the possibility of all the particles moving toward the same position. This provides the FPO with individual competition characteristic,

which is more likely to reduce the rapid social collaboration process and increase the ability of being out of the local optimum. Furthermore, only a limited number of chances are released in each iteration, which greatly stimulates the competition among the elite team, strengthening the ability of the elite to find the better place. Finally, in the later stage of optimization, according to decreasing search space, elites will try every means to aggressively occupy the companion's territory until final survival determines the best global optimal solution. Eight well-known benchmark functions are used to test the performance of FPO. Four typical multimodal benchmark functions are used to test the global search ability of FPO and four fixed-dimension multimodal optimization problems are selected to make a comparison of the convergence rate between four well-known algorithms. The experimental results show that the FPO algorithm is able to provide appropriate exploitation, utilizing local minima avoidance and exploration simultaneously.

Chapter 4

A New Hybrid Fuzzy Clustering Algorithm for Multivariate Data

Fuzzy c-means (FCM) is the most common fuzzy clustering model that uses an objective function to measure the desirability of partitions. The objective function of the FCM is the multimodal function which means that it may contain many local minima. Consequently, while minimizing the objective function, there is the possibility of getting stuck in local minima or saddle points. To avoid this problem, a new hybrid fuzzy clustering algorithm incorporated the Fitness Predator Optimization (FPO) is proposed in this chapter. In addition, the performance of FCM depends on the initialization of the cluster centroids. The number of clusters needs to be specified in advance. When the number of clusters is fixed to k, Fuzzy c-means clustering gives a formal definition as an optimization problem: find the k number of cluster centroids and assign the objects to the most probability of cluster centroid. However, the k is hard to be clearly and easily confirmed in a practical application. To determine the best number of classes, the *mixed pseudo F* statistic method is introduced in this chapter according to the theory of difference analysis. Consequently, the FPO-FCM algorithm with *mixed pseudo F* index is proposed in this chapter which automatically deter-

mines the number of classes, provides the promising optimal solution and the local minima avoidance for the clustering problem.

4.1 Overview and Preliminaries

The most common popular data mining techniques discussed are clustering and classification. The clustering aims at identifying and extracting significant groups in the underlying data, which is an unsupervised learning method. In the field of clustering, Fuzzy c-means (FCM) is one of the most popular algorithms.

4.1.1 Fuzzy c-means Clustering Algorithm

There are three main types of fuzzy clustering − fuzzy clustering based on fuzzy relation, fuzzy clustering based on objective function and fuzzy generalized k-nearest neighbour rule. The fuzzy clustering based on objective function is the most popular one, because it is quite facile, and allows the most precise formulation of the clustering criteria. The most popular version is the Bezdek's FCM model [5],[6] with the generalized objective function.

$$J_m(U,V) = \sum_{i=1}^{c}\sum_{k=1}^{n}(u_{ik})^m \mid x_k - v_i \mid^2 \tag{4.1}$$

Where m ($m > 1$) is a scalar for the weighting exponent and controls the fuzziness of the resulting clusters. The FCM model partitions a data set $X = \{x_1, ..., x_n\}$ into c ($1 < c < n$) number of fuzzy clusters with $V = \{v_1, v_2, ..., v_c\}$ cluster centroids by a partition matrix U. The matrix U shows the fuzzy relation from set of data objects, which is expressed as follows:

$$U = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \cdots & u_{ij} & \cdots \\ u_{c1} & \cdots & u_{cn} \end{bmatrix} \tag{4.2}$$

In which u_{ij} is the membership function of the j^{th} data object with the i^{th} cluster within the constraints of $u_{ij} \in [0,1]$ and $\sum_{i=1}^{c} u_{ij} = 1$. Clustering partitions a data set into subsets by finding the maximum membership grade u_{ij} of data object x_i belonging to the cluster j. This model aims to minimize the following objective function with respect to each fuzzy membership grade u_{ij} and each cluster centroid v_i. In most of cases, the distance between x_k and v_i is assigned with the Euclidean norm and the fuzzifier $m = 2$. A popular method to optimize the FCM model is Alternating Optimization (AO) through the necessary conditions extrema of $J_m(U, V)$:

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} (\frac{\mid v_i - x_k \mid}{\mid v_j - x_k \mid})^{2/(m-1)}} \tag{4.3}$$

$$v_i = \frac{\sum_{k=1}^{n} u_{ik}^m x_k}{\sum_{k=1}^{n} u_{ik}^m} \tag{4.4}$$

The reformulated version of $J_m(U, V)$ [32] is obtained by inserting (4.3) into (4.1).

$$J_m(V, X) = \sum_{i=1}^{c} \sum_{k=1}^{n} \frac{\mid v_i - x_k \mid^2}{\sum_{j=1}^{c} (\frac{\mid v_i - x_k \mid}{\mid v_j - x_k \mid})^{2m/(m-1)}} \tag{4.5}$$

In this chapter we consider a widely used FCM model with cluster center prototype. Then FCM-AO-V is described in Algorithm 5.

Figure 4.1 shows $J_m(V, X)$ in 3-dimensional graph with two clustering centroids $v_1 \in [-1, 1]$ and $v_2 \in [-1, 1]$. The reformulated function can be visualized for the trivial data set $X = \{x_1, ..., x_{100}\}$, $x_i \in [-5, 5]$ with the parameters $m = 2, c = 2$. It also shows that the objective function $J_m(V, X)$ is a non-linear multimodal function with a number of local minima. Obviously, the alternating optimization or gradient based methods might get stuck in these local extrema.

Algorithm 5 FCM-AO-V

1: Initialize data: $X = \{x_1, x_2, ..., x_n\}$
2: Initialize the clustering centroids $V = \{v_1, v_2, ..., v_c\}$
3: Initialize the maximum iterations t_{max}
4: **while** $t \leq t_{max}$ **do**
5: Generate the partition matrix u_{ij} by (4.3)
6: Generate the new clustering center v_i by (4.4)
7: **end while**
8: Output U, V

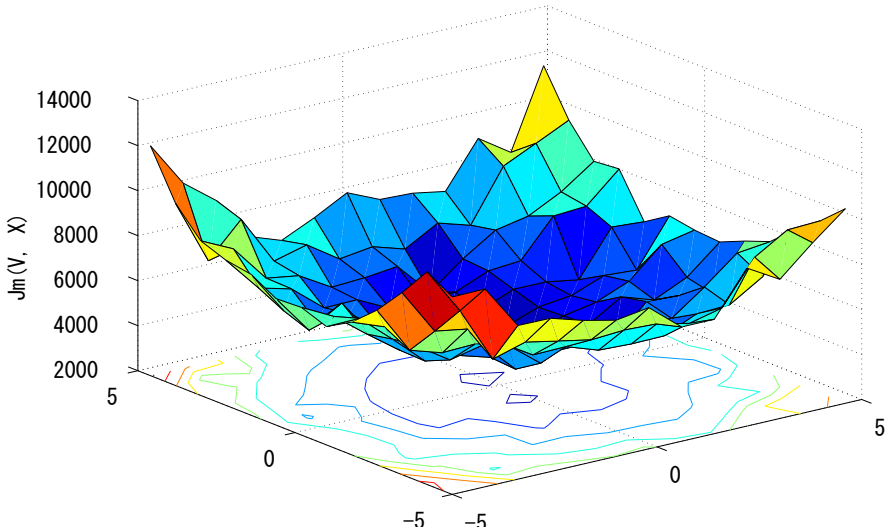


Fig. 4.1 Reformulated objective function $J_m(V, X)$

In order to avoid local minima of the FCM algorithm, various computing techniques such as artificial neural networks [70] [50], hybrid fuzzy time series approach [22], genetic algorithms and PSO-based fuzzy clustering algorithms [30] have been used in FCM recently. Some researchers adopt the stochastic methods such as evolutionary or swarm-based methods to increase the global convergence ability of fuzzy clustering. In [59], authors used a Fuzzy c-means algorithm based on Picard iteration and PSO (PPSO-FCM) to improve the performance of FCM. In [95], a hybrid data clustering algorithm based on PSO and KHM is proposed, which makes full use of the merits of PSO and KHM. The QPSO algorithm proposed by [86] outperforms traditional PSO in search ability as well as having less parameter to control. Then, [92] proposed a new hybrid fuzzy clustering algorithm that incorporates the Quantum-behaved PSO into the FCM model.

4.1.2 Fuzzy Clustering with Quantum-based PSO

In the hybrid fuzzy clustering algorithm QPSO-FCM, a particle represents a set of clustering center. The objective function $J_m(V, X)$ is defined as the particle's fitness function. The pseudocode of QPSO-FCM can be summarized in Algorithm 6.

According to the experimental results compared with FCM, GA-FCM and PSO-FCM [92], this hybrid fuzzy clustering (QPSO-FCM) can not guarantee to converge the global best solution each time, but could find the nearly better solution. Similarly, a new optimization technique named as IQPSO-FCM in [2] is a combination of FCM and improved QPSO to drive the clustering efficiency in standard medical and non-medical data sets. However, the diversity declines rapidly in the later iteration process, leaving the QPSO with great difficulties of escaping local optima.

Furthermore, the number of clusters of QPSO-FCM needs to be specified in advance. However, the number of cluster centroids is hard to be clearly confirmed in a practical application. Some investigations have found that the *Pseudo-F* statistic method is a valid index

Algorithm 6 QPSO-FCM

Initialize the population size M and the parameters of QPSO: α, φ, μ
Normalize the sample data and the maximum iterative count t_{max}
Initialize the p_i for each particle and the p_g for the swarm
while $t \leq t_{max}$ **do**
  Calculate fitness value of each particle using equation (4.5)
  Calculate p_i for each particle and p_g for the swarm
  Calculate the *mbest* by
  $mbest = \frac{1}{M}\sum_{i=1}^{M} p_i = (\frac{1}{M}\sum_{i=1}^{M} p_{i1}, ..., \frac{1}{M}\sum_{i=1}^{M} p_{id})$
  Update the new particle using:
  $p_{id} = \varphi * p_{id} + (1 - \varphi) * p_{gd}$
  $x_{id} = p_{id} \pm \alpha * \mid mbest_d - x_{id} \mid * ln(\frac{1}{\mu})$
end while
Generate the partition matrix u_{ij} by equation (4.3)
Generate the final clusters with the partition matrix
output U, V and $J_m(V, X)$

for cluster number analysis. Thus, the index of *Pseudo-F* can be applied to generate the optimal number of cluster centroids.

4.1.3 Pseudo F Statistic Method

The *Pseudo-F* statistic method describes the ratio of *between-cluster* variance to *within-cluster* variance [9]. *between-cluster* variance measures how separated clusters are from each other. The *between-cluster* is calculated as:

$$\begin{cases} B_c = \sum_{t=1}^{c}\sum_{i=1}^{n_t} n_t(v^t - \overline{v})^2 \\ v^t = \frac{1}{n_t}(x_1^t + x_2^t + \ldots + x_{n_t}^t) \\ \overline{v} = \frac{1}{n}(x_1 + x_2 + \ldots + x_n) \end{cases} \tag{4.6}$$

Where c is the number of clusters, n is the number of observations, and n_t is the number of samples in the cluster t. B_c is the *between-cluster* sum of squares, v^t is the centroid of

the cluster t, and $\overline{v}$ is the centroid of the whole samples. The *within-cluster* variance just measures how tight the clusters fit together. The *within-cluster* is shown as:

$$\begin{cases} P_c = \sum_{t=1}^{c} \sum_{i=1}^{n_t} n_t (x_i^t - v^t)^2 \\ v^t = \frac{1}{n_t}(x_1^t + x_2^t + \ldots + x_{n_t}^t) \end{cases} \tag{4.7}$$

Where P_c is the *within-cluster* sum of squares. Then *Pseudo-F* statistic index is calculated as:

$$Pseudo\text{-}F = \frac{B_c}{P_c} * \frac{n-c}{c-1} \tag{4.8}$$

If *Pseudo-F* is decreasing, it means either the *within-cluster* variance is increasing or the *between-cluster* variance is decreasing (numerator). Larger number of the *Pseudo-F* usually indicates a better clustering solution. For example, when *Pseudo-F* is used to evaluate the number of clusters, the maximum value of *Pseudo-F* indicates that the current used number of clusters may be the optimal class number. Generally, the *Pseudo-F* statistic method is fit for the single dimensional data. However, to the multidimensional data, the evaluation result is not as well as the single dimensional data. Then *Mixed-F* statistic method is adopted in this chapter.

4.1.4 Solutions

In this chapter, an application of the proposed FPO is utilized in the field of fuzzy clustering. In FCM model, the probability of finding the global optimum can be increased by FPO due to its outstanding global searching ability. Thus a new hybrid fuzzy clustering algorithm FPO-FCM [96] is proposed. Furthermore, a *Mixed-F* index is integrated into the FPO-FCM to evaluate the effectiveness of cluster number analysis for the multidimensional data. Consequently, the FPO-FCM algorithm with *Mixed-F* index could automatically determine the optimal number of classes. Experimental results show that the proposed FPO-FCM with a

Mixed-F index could satisfy the pre-selection of the number of the clusters and avoid local minima for the clustering problems. The outline of this chapter is organized as follows. In Section 4.2, a new hybrid fuzzy clustering algorithm based on the FPO (FPO-FCM) is proposed at first. Secondly, the FPO-FCM is verified by five widely used data sets in the pattern recognition literature. To determine the best number of clusters, FPO-FCM algorithm with Mixed-F index is developed in Section 4.3. Finally, some concluding remarks are provided in Section 4.4.

4.2 Fuzzy Clustering with Fitness Predator Optimization

4.2.1 Proposal of FPO-FCM Algorithm

In FPO-FCM, position $\boldsymbol{x}_i$ is a vector with $c * d$ dimensions. It can be expressed as follows:

$$\boldsymbol{x}_i = (v_{11} \cdots v_{1d} \cdots v_{j1} \cdots v_{jd} \cdots v_{c1} \cdots v_{cd}) = (\boldsymbol{v}_1 \cdots \boldsymbol{v}_j \cdots \boldsymbol{v}_c) \tag{4.9}$$

Where c is the number of cluster centroids. The cluster centroid $\boldsymbol{v}_j$ is a vector with d dimensions. There are $\boldsymbol{P}$ number of particles $\boldsymbol{x}_i$ ($i \in [1..\boldsymbol{P}]$) that composed a swarm. In FPO-FCM, we need a function to evaluate the generalized solutions called fitness function. In this chapter, equation (4.5) is used for the fitness function. The FPO-FCM algorithm can be stated as Algorithm 7.

4.2.2 Experiments and Discussion

In this section, the performance of FPO-FCM is verified by five widely used data sets in the pattern recognition literature.

Algorithm 7 FPO-FCM

Initialize the swarm population P, the number of clusters c
Normalize the data set within a range of $[0.1, 0.9]$ on d dimensions
Initialize each particle's position $\boldsymbol{x}_i$: $\boldsymbol{x}_i = rand() \in (0.1, 0.9)$
Get the clustering centroids V: V=reshape($\boldsymbol{x}_i$, c, d)
Initialize the maximum iterative count t_{max}
Initialize the p_i for each particle and the p_g for the swarm
while $t \leq t_{max}$ **do**
  if $rand() < \frac{fitness(x_i)}{\sum_{i=1}^{n} fitness(x_i)}$ **then**
    particle $\boldsymbol{x}_i$ get a chance to update its position
    for each particle **do**
      Use the elitism strategy function
    end for
  end if
  For all the population use the elitism strategy function
  Update the p_i and the p_g
end while
Get the final cluster centroids V^*: V^* =reshape(p_g, c, d)
Partition the data set with the final cluster centroids V^*
Output V^* and the classified data set by V^*

Experimental Method

The FPO-FCM algorithm is compared with K-means, FCM and QPSO-FCM by five benchmark data sets that obtained from the UCI Machine Learning Repository. In Table 4.1, all the data sets are multivariate data type. In particular, the Lung Cancer dataset (LC) described 3 types of pathological lung cancers. In the original data, five instances were missing some feature values, as such only 27 vectors are collected as our experimental samples. These three clusters are more likely to highly overlap [77], so finding the right class distribution is very difficult. The class distribution reflects the number of cluster centroids and the number of instances on each class. Generally, the intra-cluster distance (*Intra-D*) and the inter-cluster distance (*Inter-D*) are used for clustering evaluation index.

$$Intra\text{-}D = [\sum_{i=1}^{c} \sum_{k \in c_i} \| x_k - v_i \|^2]/c \tag{4.10}$$

Table 4.1 Description of benchmark data sets

Data Set	Data Type	Instances	Dimension	Class Distribution
Iris	Multivariate	150	4	(50,50,50)
Wine	Multivariate	178	13	(59,71,48)
Breast Cancer Wisconsin (BCW)	Multivariate	699	10	(458,241)
Wisconsin Diagnostic Breast Cancer (WDBC)	Multivariate	569	32	(357,212)
Lung Cancer (LC)	Multivariate	27	56	(8,10,9)

$$Inter\text{-}D = \sum_{i,j \in c_i} \| v_i - v_j \| \tag{4.11}$$

$$QE = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m d_{ik}^2 (x_k, v_i) \tag{4.12}$$

When the value of *Intra-D* decreasing, it means that the data partition is more accurate. When the value of *Inter-D* increasing, the data partition is more accurate as well. In this chapter, Quantum Error (*QE*) which reflected the tightness of clustering is set as the objective function $J_m(V, X)$ of FCM.

In experiment A, four data sets (iris, wine, BCW and WDBC) are selected to evaluate the performance of FPO-FCM compared with K-means, FCM and QPSO-FCM. Each hybrid fuzzy swarm algorithm was run with 100 iterations and a population size of 30 on each data set. The test environment and experimental execution parameters are shown in Table 4.2 and Table 4.3 respectively.

In experiment B, a relatively high dimensionality data set, lung Cancer data, is selected to evaluate the performance of FPO-FCM comparing with K-Means, FCM and QPSO-FCM. The FPO-FCM and QPSO-FCM terminating condition are limited in 500 consecutive itera-

Table 4.2 Evaluation test environment

OS	Windows 7
Processor	Intel(R) $Core^{TM}$ i7 CPU 2.80GHz
Memory (RAM)	8.00GB
System type	64-bit operation system
Tool	MATLAB 7.10.0

Table 4.3 Parameters Setting of experiment A

Experiment A	Algorithm	Parameters
Population=30	FCM	m=2
Max iteration=100	QPSO-FCM	m=2, $\alpha \in [0.5, 1.0]$
Run Number=50	FPO-FCM	m=2, c=2, $\omega \in [0.2, 1.0]$

tions and a population size of 30 for both of them. All the execution parameters in experiment B are shown in Table 4.4.

Table 4.4 Parameters setting of experiment B

Experiment B	Algorithm	Parameters
Population=30	FCM	m=2
Max iteration=500	QPSO-FCM	m=2, $\alpha \in [0.5, 1.0]$
Run Number=30	FPO-FCM	m=2, c=2, $\omega \in [0.2, 1.0]$

Experimental Results

Table 4.5 Experimental results of A

Data$^{\diamond}$	Alg.	ED	*Intra-D*	*Inter-D*	QE	Acc.(%)
Iris	K-means	(0,3,13)	0.4483	1.0942	1.2393*	89.33
	FCM	(0,7,6)	0.3742	1.0409	1.2701	91.33
	QPSO-FCM	(0,8,4)	0.0065	0.0112	1.1957	92.00
	FPO-FCM	(0,2,9)	0.0062	0.0122	1.1953	92.67
Wine	K-means	(13,21,19)	0.0476	0.2383	0.1429*	70.22
	FCM	(13,20,23)	0.0570	0.1804	6.2100	68.54
	QPSO-FCM	(2,18,29)	0.0033	0.0117	5.6024	72.47
	FPO-FCM	(10,21,17)	0.0025	0.0150	5.5827	73.03
BCW	K-means	(11,18)	275.67	0.9629	551.35*	95.85
	FCM	(14,4)	271.76	0.9212	258.05	97.42
	QPSO-FCM	(61,17)	335.87	0.2877	296.62	88.84
	FPO-FCM	(5,7)	266.29	1.1054	256.62	98.28
WDBC	K-means	(83,0)	1.4610	0.1447	2.9219*	85.41
	FCM	(0,86)	1.7658	0.0897	1.9173	84.89
	QPSO-FCM	(0,77)	1.4462	0.1352	0.0018	86.47
	FPO-FCM	(87,0)	1.7720	0.1317	2.0719	84.71

$^{\diamond}$ Each data set is normalized within a range of $[0.1, 0.9]$

* Quantum Error of K-means algorithm is defined as: $QE = \sum_{i=1}^{c} \sum_{k=1}^{N_i} d_{ik}^2(x_k, v_i)$

* Where c is the total number of clusters and N_i is the count of data in each cluster

Table 4.5 resumes the clustering results of FPO-FCM, K-means, FCM and part of results of QPSO-FCM obtained from [92]. The statistical results include four kinds of clustering evaluation indexes and the partition accuracy rate (Acc.). Error Distribution (ED) reflects

the number of instances that were wrongly assigned to each class. *Intra-D*, *Inter-D* and *QE* are used as clustering evaluation indexes for all the algorithms.

Table 4.6 resumes the lung cancer data's clustering results of K-means, FCM, QPSO-FCM and FPO-FCM. Due to the highly overlapping character of clusters in lung cancer, the clustering accuracy of all of the algorithms are dramatically decreased compared with experiment A. However, the FPO-FCM has a higher clustering accuracy than the others. The experimental results show that the proposed algorithm has a good robustness for the high dimensional and overlapping clusters in data set.

Table 4.6 Experimental results of B

Data◇	Alg.	ED	*Intra-D*	*Inter-D*	*QE*	Acc.(%)
LC	K-means	(2,7,4)	1.98e+01	3.09e+00	5.95e+01*	51.85
	FCM	(4,5,4)	2.36e+01	1.18e-11	1.33e+01	51.85
	QPSO-FCM	(3,4,6)	2.36e+01	1.59e+00	1.32e+01	51.85
	FPO-FCM	(4,3,2)	2.28e+01	9.59e-01	1.27e+01	66.67

◇ The data set is normalized within $[0.1, 0.9]$

* Quantum Error of K-means algorithm is defined as: $QE = \sum_{i=1}^{c} \sum_{k=1}^{N_i} d_{ik}^2(x_k, v_i)$

* Where c is the total number of clusters and N_i is the count of data in each cluster

Figure 4.2 shows the convergence curve of FCM with lung cancer data set. Obviously, the FCM model is trapped into local minima and cannot improve the objective function value after the fourth iteration. Figure 4.3 shows the convergence curves between FPO-FCM and QPSO-FCM in lung cancer data set. Both of the algorithms constantly minimize the objective function within 500 times of the iteration. Compared with QPSO-FCM, the FPO-FCM has the better average global optimal solution during the iteration process.

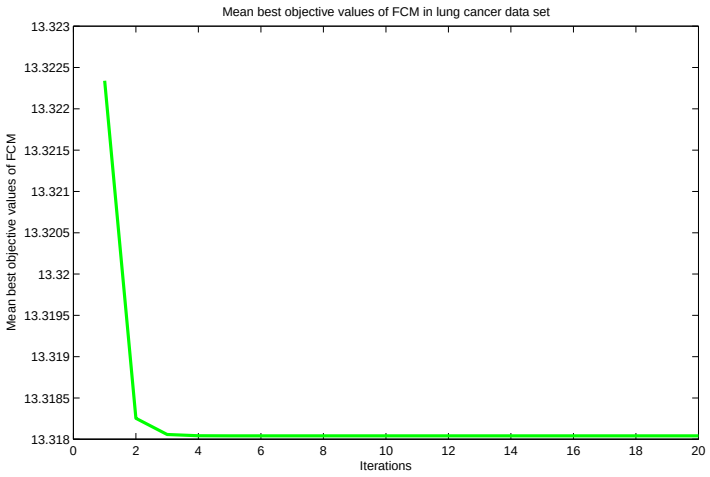


Fig. 4.2 Convergence curve of FCM

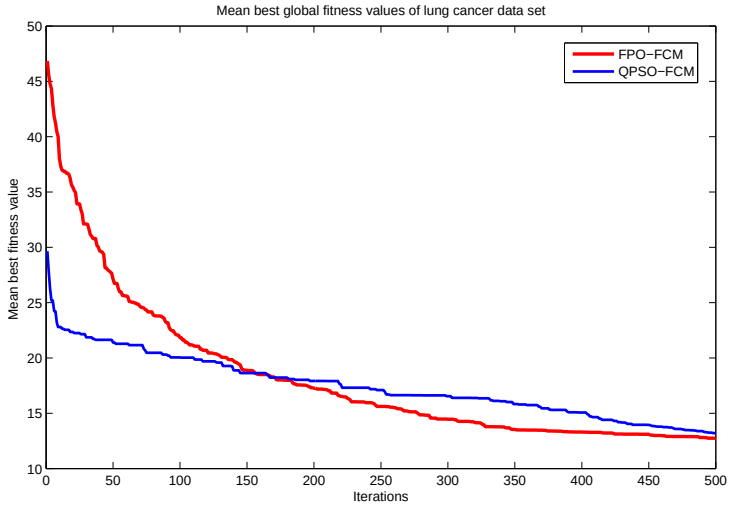


Fig. 4.3 Convergence curve of FPO-FCM and QPSO-FCM

Discussion

According to the results of experiment A, it is easy to see that FPO-FCM shows its competitive global search ability on the three benchmark data sets except WDBC. The best classification results of WDBC in experiment A is QPSO-FCM. We think that QPSO has better global search ability for the clustering of space partition that resembles the shape of a completely round ball. Then QPSO-FCM demonstrates its superior performance to the WDBC that with the similar space distribution. In experiment B, LC is a fairly difficult clustering problem due to its high dimensionality and overlapping clusters. In table 4.6, the *Intra-D* and *Inter-D* of K-means algorithm are better than the others. However, K-means does not work well as we expected. One of reasons is that using euclidean norm as similarity calculation formulation may not fit for the overlapping cluster such as lung cancer data set. Despite the difficulty clustering problem of LC, FPO-FCM is still able to escape the trap of the suboptimal solution and to find the global optimum.

4.3 FPO-FCM Algorithm with Mixed-F Index

4.3.1 The Mixed-F Statistic Method

To the multidimensional data, the reformulated *Pseudo-F* is depicted as following [94]:

$$F(k) = \frac{\sum_{i=1}^{c} n_i (v_{ik} - \overline{v}_k)^2 \cdot (n - c)}{\sum_{i=1}^{c} \sum_{j=1}^{n_j} (x_{ijk} - \overline{v}_{ik})^2 \cdot (c - 1)} \tag{4.13}$$

Where k is the dimensionality of the data, n_i is the number of samples in the cluster i. v_{ik} is the cluster center of k^{th} dimensional data in the cluster i. $\overline{v}_k$ is the k^{th} dimensional average cluster center data, x_{ijk} is the k^{th} dimensional sample data (x_j) in the cluster i. $\overline{v}_{ik}$ is the k^{th} dimensional cluster center v_i. This $F(k)$ statistic method follows the *F-distribution*

with $(c-1, n-c)$ degrees of freedom under the null hypothesis. In order to strengthen on the impact of smaller $F(k)$, a form of multiplicative inverse weighting is used for mixed *Pseudo-F* . Then the mixed *Pseudo-F* is denoted as [94]:

$$Mixed\text{-}F = \sum_{k=1}^{p} \frac{\frac{1}{F(k)}}{\sum_{k=1}^{p} \frac{1}{F(k)}} \cdot F(k) = \frac{p}{\sum_{k=1}^{p} \frac{1}{F(k)}} \tag{4.14}$$

where p denotes the number of dimensions of the data. The $Mixed\text{-}F$ also follows the *F-distribution* with $(c-1, n-c)$ degrees of freedom.

4.3.2 The Flow Chart of FPO-FCM with Mixed-F method

Figure 4.4 illustrates the flow chart of FPO-FCM with validity index *Mixed-F*.

First of all, a range of cluster numbers is initialized in the first stage of FPO-FCM. The position of particle x_i represents a set of clustering centroids. The number of particles composed a swarm of FPO-FCM. Limited number of chances is released in each iteration, only the competitive particle could get the chance to update its position. When the maximum iteration is reached, the final cluster centroids and the value of the fitness function are calculated by the partition matrix U. The value of *Mixed-F* is calculated based on the result of the clustering partition. When the maximum classification number is reached, the maximum value of *Mixed-F* indicates that the current used number of clusters is the optimal class number. In the next section, the performance of FPO-FCM with *Mixed-F* index is verified by two non-medical data sets and three medical data sets from UCI machine learning repository that widely used in the pattern recognition literature.

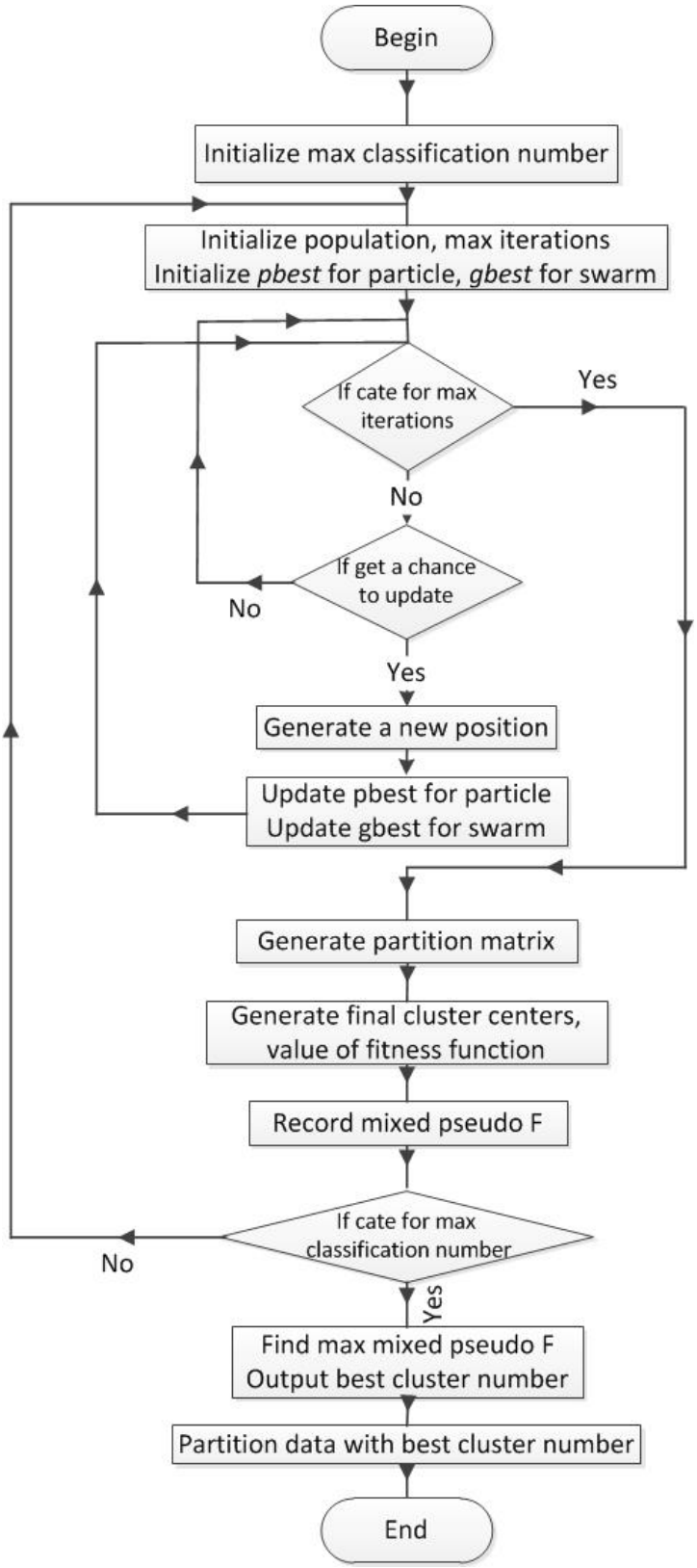


Fig. 4.4 The flow chart of FPO-FCM with *Mixed-F*

Table 4.7 Parameters setting of experiment

Algorithm	Parameters
FCM	$m = 2$
FPO-FCM	$m = 2$, $c \in [2, 5]$ $w \in [0.2, 1.0]$, $P = 30$, $t_{max} = 500$

Table 4.8 Record of *Mixed-F* index of non-medical datasets

No. clusters	*Mixed-F*	
	Iris	Wine
2	44.45	**3.55** ↑
3	**212.13** ↑	3.11
4	143.40	0.52
5	72.26	0.33

4.3.3 Experiments and Discussion

Experimental Method

To test the performance of *Mixed-F* index, in experiment A, Iris flower data set and Wine data set are used to compared with three well-known cluster validity indexes. In experiment B, three medical data sets are used to evaluate the effect of the automated classification of FPO-FCM and global optimization ability of clustering partition. All of the data sets are shown in Table 4.1. In particular, Lung Cancer data set is a relatively high dimensionality data than the others. The test environment and experimental execution parameters are shown in Table 4.2 and Table 4.7 respectively.

Experiment of Non-medical Data Sets

The Iris flower data set and Wine data set are both multivariate data sets that classified 3 clusters. In this experiment, the FPO-FCM algorithm is used to choose an optimal number of clusters for these two non-medical data sets with *Mixed-F* index. The range of cluster numbers is from 2 to 5. The values of *Mixed-F* on different class numbers are presented in Table 4.8 [81].

Table 4.9 Comparison of the optimal number of clusters with four cluster indexes

m	Index	Iris	Wine
2	V_{PE}	2	2
	V_{XB}	2	2
	V_{FS}	5	11
	Mixed-F	3	2

Table 4.10 Record of *Mixed-F* index of medical datasets

$No.$ clusters	*Mixed-F*		
	BCW	WDBC	LC
2	**3.97** ↑	**2.81** ↑	0.01
3	1.39	0.32	**0.02** ↑
4	0.93	0.08	0.01
5	0.07	0.15	0

From Table 4.8, we could find that the maximum *Mixed-F* index is reached when the Iris data set is classified with 3 clusters. Similarly, the maximum *Mixed-F* index is obtained when the Wine data set is divided into 2 clusters. To evaluate the experimental results of *Mixed-F* index, three well known cluster validity indexes are compared: The Partition Entropy (PE) index proposed by Bezdek, the V_{XB} index presented by Xie & Beni and the Fukuyama-Sugeno (FS) index. Comparison results are provided in Table 4.9.

In Table 4.9, it is easy to find that *Mixed-F* has a higher accuracy compared with other three cluster indexes.

Experiment of Medical Data Sets

In this experiment, three kinds of medical data sets are used to verify the automatically optimal number of clusters prediction. The range of cluster numbers is from 2 to 5. The values of *Mixed-F* on different class numbers are shown in Table 4.10.

From Table 4.10, we could find that the maximum *Mixed-F* is 3.97 when the BCW data set is classified with 2 clusters. When the WDBC data set is divided into 2 clusters, *Mixed-F* has the optimal value 2.81. For the Lung Cancer data set, when its number of clusters is 3,

the *Mixed-F* has the maximum value 0.02. The optimal number of clusters for these three medical data sets are completely satisfied with the real situation. Experimental results show that the proposed approach could demonstrate the desirable performance of optimal number of clusters prediction.

4.4 Summary

In this chapter, a cluster optimization methodology is proposed based on the Fitness Predator Optimization (FPO) algorithm. The proposed approach deals with the modified FPO algorithm for fuzzy clustering optimization. In the proposed new hybrid fuzzy clustering algorithm (FPO-FCM), the position of each particle represents a set of clustering centroids, a number of particles composed of a swarm of FPO-FCM. The objective function $J_m(U, V)$ of FCM is used for evaluating the generalized solutions. The experimentation is done with five benchmark data sets covered examples of data from low and high dimensions. Compared with traditional algorithms (K-means and FCM) and hybrid swarm algorithm (QPSO-FCM), FPO-FCM has a higher robustness and better global optimization ability of clustering partition. In addition, a *Mixed-F* index is introduced based on the theory of difference analysis. The proposed approach deals with the modified FPO-FCM algorithm for fuzzy clustering optimization. The same benchmark data sets from the UCI Machine Learning Repository are used to evaluate the introduced *Mixed-F* index. All of the experimental results show that the FPO-FCM algorithm could satisfy the pre-selection of the number of clusters and avoid the local optima for clustering problem simultaneously.

Chapter 5

Dynamic Virtual Teams for Fitness Predator Optimization

In chapter 3, Fitness Predator Optimization (FPO) is proposed to avoid premature convergence for multimodal problems. The experimental results show that the FPO algorithm is able to provide excellent exploitation, utilizing local minima avoidance and exploration simultaneously. However, the slow convergence speed in the early iterations and more elapsed time of the whole optimization process become the bottleneck to restrict the improvement of the FPO. This motivated our attempt to present new approaches to improve the performance of FPO. In this chapter, two variants of FPO, namely, DFPO and DFPO-r are proposed. First of all, a dynamic team model is utilized in FPO named DFPO to accelerate early convergence rate. Secondly, a method of team size selection is proposed for DFPO-r to increase the population diversity. Ten well-known multimodal benchmark functions are used to evaluate the solution capability of DFPO and DFPO-r. Experimental results show that both DFPO and DFPO-r demonstrate the desirable performance. Furthermore, DFPO-r shows a more robust performance compared with DFPO in experimental study.

5.1 Overview and Preliminaries

Dynamic virtual team proposed by the Dolphin Partner Optimization (DPO) [80] mimics the hunting mechanism of dolphins in nature. The performance of DPO with the virtual team model is evaluated on several benchmark functions. Experiments show that it is able to demonstrate the desirable global searching ability on multimodal functions. The basic concepts of dynamic virtual team, the algorithm of a dynamic virtual team model and the mechanism of Dolphin Partner Optimization (DPO) are primarily presented as follows.

5.1.1 Dynamic Virtual Team Model

The dynamic virtual team is first proposed by the Dolphin Partner Optimization (DPO) [80]. It mimics the hunting mechanism of dolphins in nature. During the hunting process, a dolphin will look for his neighbors and select some of them as his partners. All of his partners and himself form a self-organizing team, which is defined as a dynamic virtual team. In a swarm, each dolphin will do the same clustering behavior at the same time. Here are the related definitions of the dynamic team.

Team: According to the nearest neighbor principle, one dolphin and some of his neighbors are composed of a dynamic virtual team. It should be noted that one dolphin could belong to multiple teams when teams are connected.

Role recognition: A dolphin evaluates himself by comparing the best fitness value with other partners in his virtual team. Normally, the one that has the best fitness value will be selected as the team leader. Each member of the team is recognized as either a leader or an ordinary member.

Exchange: A dolphin provides his best experience information to the team. The team's best experience could be concluded by the comparison of individual experience among team members. Information expansion of individuals and teams experience is carried out by the dolphin that belongs to multiple teams.

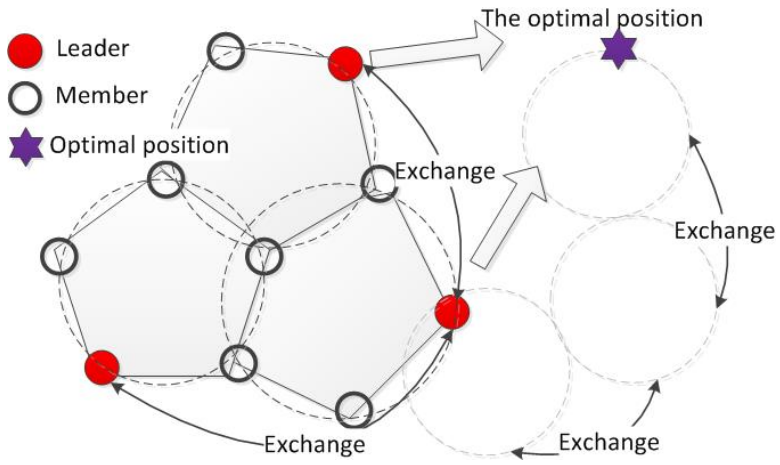


Fig. 5.1 Diagram of dynamic virtual teams

Leader: The function of team leader dolphin is to analyze all communication information and to predict the most available position for the next step.

Member: The function of the ordinary member is to follow up with the team leader.

The simple diagram of a virtual team is shown in Fig.5.1. The solid red circles are team leader and the blank rings present ordinary team members. According to the nearest neighbor rule, each individual dynamically select its 4 immediate neighbors as a team on each iteration. The structure of this team is seemed as an appropriate circle topology. In the circle topology, if two teams have the same members, they are interrelated and affect with each other. Otherwise, they are distant from one another and also independent of one another.

Generally, the leader marked with red color has the best fitness value among the team. The main function of team leader is to analyse all communication information and to predict the most available position for the further search. The star in purple is the future optimal position that is obtained by one of team leaders from the exchanged information with interrelated teams. The expansion of personal experience and team's experience is carried out by some interrelated members between the teams. Another function of the ordinary team member is to follow up with the team leader which belongs to. Having accomplished the

description of original concepts of dynamic virtual teams, the main organization algorithm of a dynamic virtual team model will be demonstrated in the next subsection.

5.1.2 Algorithm of Dynamic Virtual Team Model

The implementation of dynamic virtual team model is shown in Algorithm 8. Two important key points that are marked with underline and bold type in Algorithm 8 are worthy of note. One of them is to predict the next best team position by the team leader, and the other one is to get the next position for the team's ordinary members.

Algorithm 8 Team Organization Function

Initialization: $popsize$, $teamsize$, each dolphin $\boldsymbol{x}_i$
for $i \in [1..popsize]$ **do**
  for $g \in [1..popsize]$ **do**
    Calculate distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_g$
    $disM(i,g) = Edistance(\boldsymbol{x}_i, \boldsymbol{x}_g)$
  end for
  Sort $disM(i,g)$ in ascending
  $Team_i = sort(disM(i,g))$
  Select $\boldsymbol{x}_i$'s dynamic partners from $Team_i$
  for $h \in [2..teamsize]$ **do**
    $partners(i) = h$
  end for
  Organize dynamic virtual team for $\boldsymbol{x}_i$
  $\boldsymbol{x}_{team(i)} = partners(i)$
end for
for all $\boldsymbol{x}_{team(i)}$ **do**
  Specify the best value of team as team leader
  if $\boldsymbol{x}_i$ is a team leader **then**
    Exchange its best position within team
    ▶**Predict the next best position**
  end if
  if $\boldsymbol{x}_i$ is an ordinary member **then**
    ▶ **Update its next position by equation (5.1)**
  end if
end for

It is an open issue as to how to predict and calculate the next team's best position and ordinary member's position. In DPO, definitions of these two points are not fit for all of the optimal problems. The dynamic team organization mimics the dolphin's teamwork during the process of searching for prey, and attacking prey. It should be noted that the structure of teamwork is a really complicated system, the proposed dynamic virtual team model is only a simplified prototype.

In order to come out the next optimal position, a so-called "Nucleus" is presented in DPO to predict the best position according to the information of individual experience and the team's best experience. As an ordinary member, its next position is updated by the equation (5.1).

$$\begin{aligned} \boldsymbol{newx}_i = \boldsymbol{x}_i + c_1 r_1 (T_{best} - \boldsymbol{x}_i) \\ + c_2 r_2 (N_{best} - \boldsymbol{x}_i) \end{aligned} \tag{5.1}$$

Where, T_{best} is the best fitness solution among the team, namely the solution of the team leader. N_{best} is the best fitness solution coming from his neighbor teams, which could be acquired from the neighboring team leader. Under the guidance of the team leader and the influence of neighboring team leaders, members are pulled into the potential global optimum. c_1 and c_2, called cognitive factor and social factor, are positive numbers defined by their upper limit(usually equals to 2.0). Two values r_1 and r_2 are random numbers generated in the interval [0,1]. Based on the dynamic virtual team model, a mechanism of Dolphin Partner Optimization is formulated and will be demonstrated in the next subsection.

5.1.3 Dolphin Partner Optimization

The Dolphin Partner Optimization mimics the dolphin's teamwork during the process of searching for prey, and attacking prey. The main two types of the "Nucleus" are defined as:

- Inner Nucleus: the nucleus is located in the inner space of team members.
- Outer Nucleus: the nucleus is located in the outer space of team members.

The calculation formula of the Inner Nucleus is stated as:

$$InN(k) = \frac{\sum_{i=1}^{n} x_i f_i^{'}(\boldsymbol{x})}{\sum_{i=1}^{n} f_i^{'}(\boldsymbol{x})} \tag{5.2}$$

Where, $f^{'}(\boldsymbol{x})$ is defined as the translation function ($f^{'}(\boldsymbol{x}) > 0$), $\boldsymbol{x}_i$ is the position of the team member. The tentative translation function is described as two forms according with disparate global best solutions: $f_{gbest}^{max}(\boldsymbol{x})$ and $f_{gbest}^{min}(\boldsymbol{x})$.

$$f_H^{'}(\boldsymbol{x}) = f(\boldsymbol{x}) - f_{gbest}^{max}(x) \tag{5.3}$$

$$f_L^{'}(\boldsymbol{x}) = \frac{1}{f(\boldsymbol{x}) - f_{gbest}^{min}(x)} \tag{5.4}$$

$f(\boldsymbol{x})$ denotes the fitness function of the DPO. $f_{gbest}^{max}(\boldsymbol{x})$ is the global maximum solution and $f_{gbest}^{min}(\boldsymbol{x})$ is the global minimum solution of the fitness function. Correspondingly, the Inner Nucleus formula has two particular descriptions: $InN_H(k)$ and $InN_L(k)$.

The outer nucleus is evaluated by:

$$OutN(k) = InN_L(k) \pm log(\frac{1}{\mu}) * (InN_L(k) - InN_H(k)) \tag{5.5}$$

Where μ is a random value within $(0, 1)$. There are many other methods to estimate the positions of inner nucleus and outer nucleus. The definitions mentioned above are provided a tentative approach and verified by some well-known multimodal benchmark problems. The DPO algorithm can be summarized in Algorithm 9.

Algorithm 9 The DPO Algorithm

Initialization: $popsize$, $teamsize$
Initialization: each dolphin $\boldsymbol{x}_i$
Do
for all dolphin $\boldsymbol{x}_i$ **do**
    Find out its partners and form a virtual team
    Find out its role in his team
    Exchange team best position within partners
end for
for Each dolphin $\boldsymbol{x}_i$ **do**
    Calculate the inner nucleus and outer nucleus
    if This dolphin $\boldsymbol{x}_k$ is the leader of his team **then**
        if fitness of Inner Nucleus is better **then**
            $\boldsymbol{x}_k = InN(k)$
        end if
        if fitness of Outer Nucleus is better **then**
            $\boldsymbol{x}_k = OutN(k)$
        end if
        Follow up with the leader
    end if
end for
Until termination criterion is met

Experimental study shows that DPO could demonstrate the desirable exploration, local minima avoidance and reasonable convergence rate simultaneously. However, the definitions of "next best team position" and "ordinary member's position" can not be fit for all kinds of optimization problems. In addition, the multivariate team size selection of DPO is not discussed in the original reference.

5.1.4 Solutions

Experiment analysis of the dynamic virtual team model reveals that the independent individual consciousness is weakened by the team leader. Our solution is to propose an appropriate wheel topology instead of the original topology structure in DPO. In this chapter, the modified structure of the dynamic virtual team as an appropriate wheel topology is applied to FPO named as a Dynamic Fitness Predator Optimization (DFPO) [97]. The modified dynamic virtual team is a self-organization team that according to the nearest neighbor clustering. It is able to build paralleled exchanging information system of DFPO with the aim of accelerating the early convergence rate and improving the global searching capability for multimodal function. In addition, a variant of DFPO, namely DFPO-r is proposed to improve the performance of DFPO [100]. In DFPO-r, a method of team size selection is developed to increase the population diversity. The population diversity is one of the most important factors that determines the performance of the optimization algorithm. A higher degree of swarm diversity is able to help DFPO-r alleviate premature convergence. In DFPO-r, two kinds of team sizes are randomly selected from the size pool, which is designed by the real situation. The strategy of selection is to choose the team size according to a higher degree of population diversity. Ten well-known multimodal benchmark functions are used to evaluate the performance of DFPO and DFPO-r. Experimental results show that both DFPO and DFPO-r could demonstrate the desirable performance.

Table 5.1 Evaluation test environment

OS	Windows 7
Processor	Intel(R) $Core^{TM}$ i7 CPU 2.80GHz
Memory (RAM)	8.00GB
System type	64-bit operation system
Tool	MATLAB 7.10.0

5.2 Proposal of DFPO

5.2.1 Study of Dynamic Virtual Team Model

In this section, the rapid convergence qualification of virtual team model will be discussed by the further study. First of all, the effect of various team sizes on accuracy of optimization and running time is tested with Griewank function and Rosenbrock function. The Griewank function is a function widely used to test the convergence of optimization functions. It has 191 minima with global minimum $f(x^*) = 0$ at $x^* = 0$. The Rosenbrock function is also a non-convex function used as a performance test problem for optimization algorithms. The global minimum is inside a long, narrow, parabolic shaped flat valley. To converge to the global minimum is fairly difficult. The bounds and global minimums of these benchmark functions as shown in Table 5.5. The evaluation test environment parameters is shown in Table 5.1. The DPO algorithm was run 50 times with a population size of 100 on each benchmark function. The statistical results including average optimum and elapsed time on each run are reported in Table 5.2.

It is worth noting that the best mean optimum is indicated with bold type and the down arrow in Table 5.2. Both of best global values for two functions are obtained when the team size equals to 9. Thus, we have reason to hypothesize that highly team size might be as good at finding optima in a problem searching space. Then the greater number of team size is used to test the performance of DPO on Rosenbrock function. Table 5.3 shows the statistical results of DPO with the range of team size $T_s \in [10, 90]$. Similarly, the DPO algorithm was

Table 5.2 The best mean optimum obtained by DPO with various team size on Griewank & Rosenbrock functions

T_s	Dim.	Iter.	Griewank		Rosenbrock	
			Mean	Time(s)	Mean	Time(s)
2	30	500	3.12e-02	1.86e+01	1.92e+02	1.83e+01
3	30	500	8.40e-03	1.76e+01	1.01e+02	2.15e+01
4	30	500	6.50e-03	1.75e+01	2.08e+02	1.72e+01
5	30	500	6.10e-03	1.75e+01	1.83e+02	1.71e+01
6	30	500	1.44e-02	1.75e+01	1.18e+02	1.70e+01
7	30	500	9.79e-02	1.74e+01	7.45e+01	1.72e+01
8	30	500	2.50e-03	1.73e+01	5.16e+01	1.71e+01
9	30	500	**2.00e-03**↓	1.73e+01	**4.80e+01**↓	1.72e+01
10	30	500	3.00e-03	1.75e+01	7.66e+01	1.70e+01

Table 5.3 The mean optimum obtained by DPO with various team size on Rosenbrock function

T_s	Dim.	Iter.	Rosenbrock	Time(s)
10	30	500	4.95e+01	1.72e+01
20	30	500	2.78e+01	1.75e+01
30	30	500	2.78e+01	1.76e+01
40	30	500	2.78e+01	1.78e+01
50	30	500	2.75e+01	1.99e+01
60	30	500	2.88e+01	2.16e+01
70	30	500	2.81e+01	2.03e+01
80	30	500	2.84e+01	2.25e+01
90	30	500	2.85e+01	2.29e+01

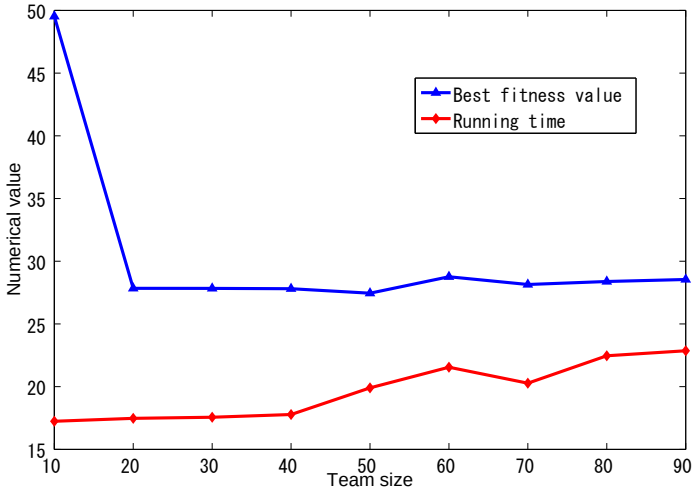


Fig. 5.2 Mean optimum and running time of DPO with different team size on Rosenbrock function

run 50 times with a population size of 100. An interesting finding from Table 5.2 and Table 5.3 is that the more number of the team size indicates the better optimal solution. It is also easily to be confirmed by Fig.5.2. The blue curve of best fitness value shown in Fig.5.2 is obviously decreased when the team size increases to 20. However, the red curve of average running time is not increased accordingly. With the corresponding experimental analysis, we believe that this dynamic virtual team model might represent a good fit for the FPO system, as it combines the potential parallelism with the iterative, synchronous nature of the FPO.

5.2.2 Modified Topology of Dynamic Virtual Team Model

In this part, our main work is to modify the generic definitions of virtual team and propose a new diagram. The original function of team leader is defined to analyze all of the communication information and to predict the most available position as a guidance for the team members. The function of the ordinary members is defined to follow up the team leader. However the individual independent consciousness is weakened by the team leader because if the particle is an ordinary team member, it will follow up with the team leader and

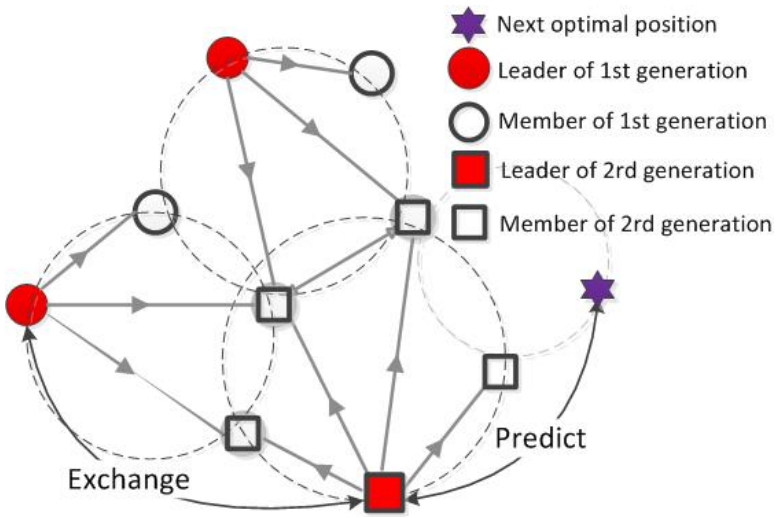


Fig. 5.3 The modified dynamic virtual team topology

lose its independent consciousness. In order to resolve this problem, the role recognition is deleted from the team, which means that each individual is the leader of its own team, the other members of the team only provide their experience as the reference information to the leader. Similarly, the other team members would not follow up with the team leader and keep their independent consciousness all the time.

The modified diagram of virtual team is shown in Fig.5.3. In the first iteration, each individual is connected to its 3 immediate neighbors. The solid red circles shown in Fig 5.3 are the team leaders. The neighbors of this team denoted with black circles are not guaranteed to connect with each other. The structure of this team is an appropriate wheel topology. The wheel topology effectively isolates individuals from one another. In the second iteration, the team size is increased to 5, the solid red box shown in Fig 5.3 is the new team leader. Which is connected to its 4 new immediate neighbours shown with black square. Obviously shown in Fig. 5.3, the team size is dynamic changed during the searching process. In addition, team members would not follow up with the team leader and keep their independent consciousness at all time. The function of the team leader is to provide team's best position for the team members. Information of individual best experience and the team's best experience is spread by weak ties of these black squared members. Each of the individuals independently depicts

its next position only according to its own experience, the random elite's experience and the neighboring teams' experience. The modified definition of updated position is:

$$\begin{aligned} newx_{ij} = x_{ij} + \theta * c * w * (x_{kj} - x_{ij}) \\ +\theta * c * w * (N_{best} - x_{ij}) \end{aligned} \tag{5.6}$$

Where $\theta = (rand - 0.5)$ is a random decimal, c is a positive constant defined by its upper limit (usually equals to 2.0). w is defined as inertia weight. N_{best} is the best fitness solution coming from the neighboring teams, which could be acquired from the neighboring team leader.

5.2.3 Pseudocode of DFPO

In this part, the dynamic virtual team is implemented in FPO and the pseudocode of DFPO is shown as below.

Algorithm 10 Main Function of DFPO

```
Initialize population: popsize, teamsize
Initialize each position: x_i = rand() ∈ (x_min, x_max)
for each position x_i do
    Create its team by Dynamic Team Organization Function
end for
for each chance of updating do
    if rand() < rand() * fitness(x_i) / Σ_{i=1}^{n} fitness(x_i) then
        get a chance to update its position
        Generate a new position x*_ij by equation (5.6)
        For each position use the elitism strategy
    end if
end for
For all population use the elitism strategy
```

Table 5.4 Comparison of mean optimum obtained by DFPO with various team size on Griewank & Rastrigin & Rosenbrock functions

T_s	Dim.	Griewank	Time(s)	Rastrigin	Time(s)	Rosenbrock	Time(s)
10	30	2.21e-04	9.81e+00	5.34e+00	8.64e+00	2.17e+01	8.59e+00
20	30	1.28e-04	9.98e+00	4.32e+00	8.74e+00	6.40e+00	8.68e+00
30	30	**9.98e-05**↓	1.07e+00	3.32e+00	8.92e+00	**1.33e+00**↓	8.73e+00
40	30	2.49e-04	1.03e+01	**2.19e+00**↓	8.96e+00	1.62e+00	8.88e+00
50	30	3.68e-04	9.98e+00	2.94e+00	9.13e+00	2.15e+01	9.10e+00

5.2.4 A Method of Team size Selection in DFPO

Three widely known benchmark functions (Griewank, Rastrigin and Rosebrock) are used to the team size selection for DFPO. The population is set to 100 and the range of team size is from 10 to 50. The maximum generation (iteration) is set as 500, 50 trial runs for each function and the mean best fitness is recorded in Table 5.4.

From Table 5.4, it can be observed that the DFPO generated the best performance on Griewank function when team size equates to 30. Such was the same case on Rosenbrock function. For Rastrigin function, the best optimum is scored when the team size is 40. Since DPO and DFPO have the same experiment environment setting (same swarm size, same team size, same number of iteration and same dimension of Rosenbrock function), the comparison between the mean best optimum and running time is shown in Fig. 5.4 and Fig. 5.5.

Figure 5.4 and Figure 5.5 indicate that DFPO with modified dynamic virtual team is able to provide very competitive results on the average of best global value in each team size selection and has faster computation time compared with DPO.

To summarize, the modified dynamic virtual team model strengthens the global optimal solution and provides the desirable speed of convergence rate for the DFPO algorithm.

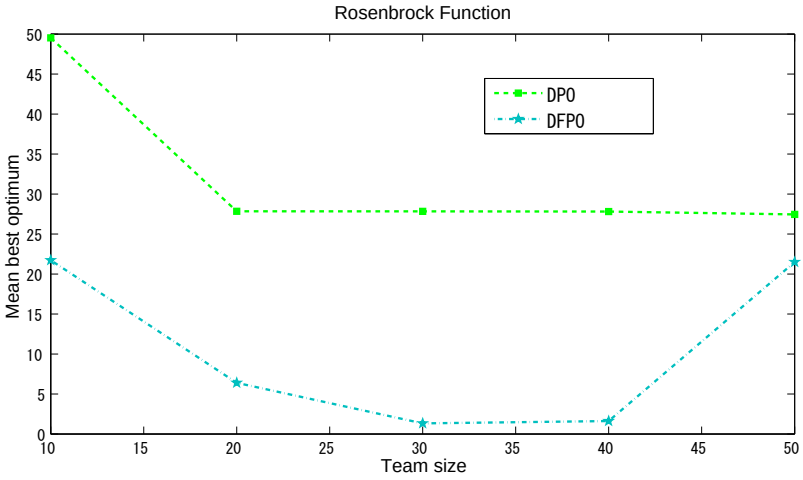


Fig. 5.4 The performance comparison between DPO and DFPO on Rosenbrock function

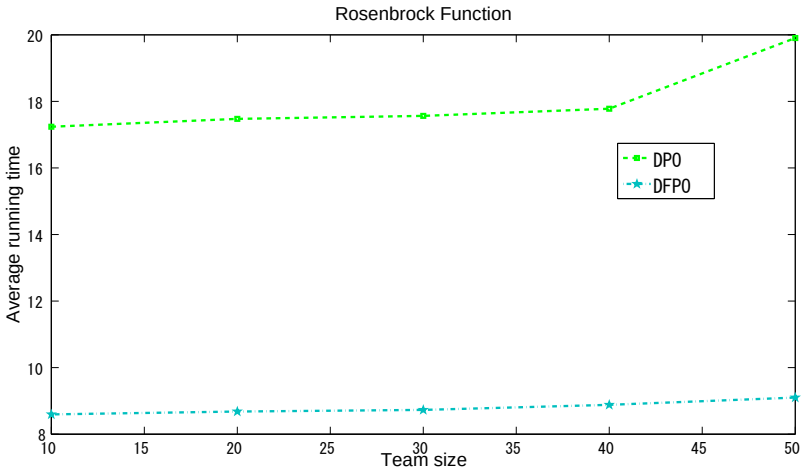


Fig. 5.5 The running time comparison between DPO and DFPO on Rosenbrock function

5.3 Proposal of DFPO-r

5.3.1 A Method for Dynamic Team Size Selection

As shown in Fig.5.3 that the team size could dynamically change during the search process. In addition, the different teams could have a various number of team members in the process of optimization. The method of team size selection is tentatively proposed by experimental study in this chapter.

Generally, we believe that increasing the population size enables the optimization algorithm to search more points and thereby obtain a better result. However, a large swarm population is not guaranteed to get the better optimized performance. A similar discussion could be found in [108] and [12]. Thus, the team size should be diversified for dynamic environment and various optimization problems.

The diversity of the population is one of the most important factors that determines the performance of the optimal algorithm. We consider that the population diversity is useful to team size section. A diversity measure $S_{(p)}$ introduced in [13] and [14] is adopted to indicate the changed swarm diversity with various populations.

$$\begin{cases} \overline{x}_j = \dfrac{1}{n}\sum_{i=1}^{n} x_{ij} \\ S_j = \dfrac{1}{n}\sum_{i=1}^{n} |x_{ij} - \overline{x}_j| \\ S_{(p)} = \dfrac{1}{D}\sum_{j=1}^{D} S_j \end{cases} \tag{5.7}$$

Where $S_{(p)}$ denotes the diversity of the swarm p, n is the swarm size, D is the dimension of the function, $\overline{x}_j$ represents the pivot of solutions in dimension j, and S_j measures solution diversity based on L_1 norm for dimension j.

An experiment is carried out to confirm the proper population size that fits for various dimensional function. It is shown in Fig.5.6 that the swarm population is not always 'the

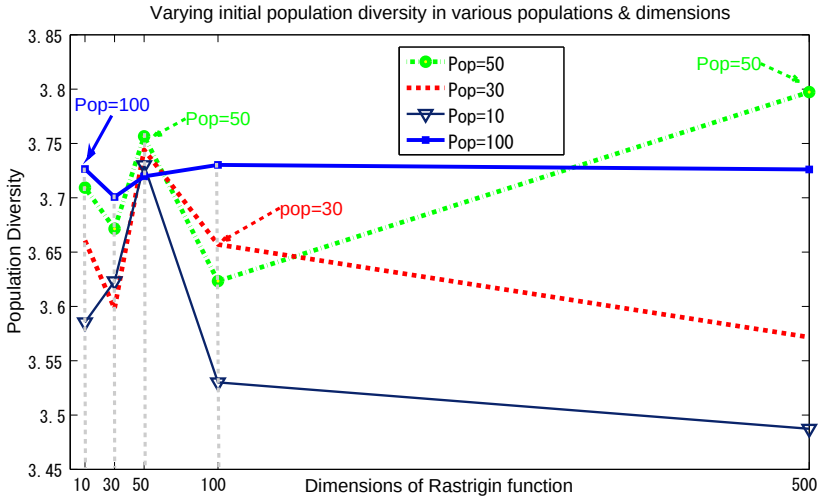


Fig. 5.6 Different population diversity in various population and dimensions on Rastrigin function

more, the better'. When dimension reaches 500, a swarm size of 50 has higher initial population diversity than that of 100. Similarly, the population diversity of the swarm size of 30 is better than that of 50 on 100 dimensions. Inspired by Competitive Swarm Optimization [12] and based on this experiment, the method of team size selection is presented here.

First of all, two kinds of team size are randomly selected from the sizes pool. The sizes pool is a boundary of team size, which could be designed according to the real situation. Secondly, both of these team sizes are compared by their population diversities. Only the better one is kept as the team size. Once the dynamic team size is researched, it will be shared with different teams during the whole process of optimization.

5.3.2 Pseudocode of DFPO-r

Algorithm 11 demonstrates the main function of DFPO-r. It should be noted that DFPO-r is a variant of DFPO. The difference between them is the method of team size selection. If the third sentence in Algorithm 11 is deleted, it will be the main function of DFPO. Thus,

Algorithm 11 Main Function of DFPO-r

```
Initialize population: popsize
Initialize x_i: x_i = rand() ∈ (x_min, x_max)
Confirm team size by population diversity
for Each position x_i do
   Create its team by Dynamic Team Organization Function
end for
for all particles do
   if x_i gets a chance to update its position then
      Generate a new position x_i^* by equation (5.6)
      if x_i^* is better than x_i then
         Record x_i^* as a new elite
      end if
   end if
end for
For all population use the elitism strategy
```

the team size of DFPO-r is dynamic confirmed under the guidance of swarm diversity while the team size of DPFO is set to one third of the population.

5.4 Experiments

In this section, ten well-known benchmark functions are used for comparison with Constricted GBest PSO[7], Constricted LBest PSO[7], DPO, FPO, DFPO and DFPO-r. The aim of the experiments is to compare the global convergence and optimization ability of DFPO and DFPO-r with other four swarm intelligence algorithms on various dimensions of multmodal problems. The analysis and discussion of the experiments are also shown later.

5.4.1 Evaluation Method

The experiments include two parts. In part A, four fixed-dimension multimodal problems are used to make a comparison of the convergence rate on Constricted GBest PSO (GBPSO), DPO, FPO and DFPO. In part B, six flexible dimension functions are used to test the capa-

bility of global optimal solution on Constricted LBest PSO (LBPSO), DPO, FPO, DFPO and DFPO-r.

The experiments are implemented on a PC with an Intel Core i7 860, 2.8 GHz CPU and Microsoft Windows 7 Professional SP1 64-bit operating system, and all algorithms are written in Matlab language. The bounds and global minimums of ten functions are shown in Table 5.5. The parameters setting are illustrated in Table 5.6 and Table 5.7.

5.4.2 Experimental Results

Table 5.8 shows the statistical results of optimization errors on four fixed-dimension multimodal functions among GBPSO, DPO, FPO and DFPO. Table 5.9, Table 5.10, Table 5.11 and Table 5.12 show the statistical results of optimization errors and average computation time on six flexible dimensional functions. To all statistical results, the best global optimization errors and mean of best global optimization errors are marked with bold type and down arrow. From Table 5.8, we found that in the early 20 times of iteration process, DFPO is able to provide very competitive results on the global optimum and the mean best optimum on most of benchmark functions.

5.4.3 Discussion

Figure 5.7, 5.8 and 5.9 are the most important processing curves of mean optimization errors obtained by Table 5.8. From Fig. 5.7, we found that three algorithms are able to accurately find the global optimum after 50 iterations. GBPSO is trapped into the local solution and could not find the accurate global optimum. In Fig.5.8, DFPO is the best ranked and DPO is the second ranked of four algorithms. In Fig.5.9, DFPO still keeps the best global searching capability, but DPO shows the worst performance compared with others. According to the results of Table 5.8 and figures, DFPO could find the satisfied global optimum within 50 iterations. While, GBest PSO and DPO are trapped into local optimal solutions on Branin

Table 5.5 Bounds and global optimums of benchmark functions

Function	Bound	Optimum
Michalewicz: $f_1(x) = -\sum_{j=1}^{D} sin(x_j)[sin(\frac{i.x_i^2}{\pi})]^{20}$	$[0, \pi]^2$	-1.8013
Branin: $f_2(x) = a(x_2 - bx_1^2 + cx_1 - 6)^2$ $+g(1-h)\cos(x_1) + 10$, $a = 1$ $b = 1.25\pi^{-2}, c = 5\pi^{-1}, g = 10, h = 0.125\pi^{-1}$	$x_1 \in [-5, 10]$ $x_2 \in [0, 15]$	0.397887
Shekel: $f_3(x) = -\sum_{k=1}^{m}(\sum_{j=1}^{4}(x_j - C_{jk})^2 + \beta_k)^{-1}$ $m = 10, \beta = \frac{1}{10}(1,2,2,4,4,6,3,7,5,5)^T$ $C = \begin{pmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3 \\ 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3 \end{pmatrix}$	$[0, 10]^4$	-10.5364
Shaffers N.2: $f_4(x, y) = 0.5 + \frac{sin^2(x^2 - y^2) - 0.5}{(1 + 0.001 * (x^2 + y^2))^2}$	$[-100, 100]^2$	0
Rosenbrock: $f_5(x) = \sum_{j=1}^{D}(100 * (x_{j+1} - x_j^2)^2 + (x_j - 1)^2)$	$[-5, 10]^D$	0
Rastrigin: $f_6(x) = \sum_{j=1}^{D}(x_j^2 - 10 * \cos(2\pi x_j)) + 10 * D$	$[-5.12, 5.12]^D$	0
Griewank: $f_7(x) = \frac{1}{4000}\sum_{j=1}^{D} x_j^2 - \prod_{j=1}^{D}\cos(\frac{x_j}{\sqrt{j}}) + 1$	$[-5, 10]^D$	0
Ackley: $f_8(x) = -a * \exp(-0.02 * (D^{-1}\sum_{j=1}^{D-1} x_j^2)\frac{1}{2})$ $-\exp(D^{-1}\sum_{j=1}^{D}\cos(2\pi x_j)) + a + \exp, a = 20$	$[-15, 30]^D$	0
Levy: $f_9(x) = \sum_{j=1}^{D-1}(\varpi_j - 1)^2[1 + 10sin^2(\pi\varpi_j + 1)]$ $+sin^2(\pi\varpi_1) + (\varpi_D - 1)^2[1 + sin^2(2\pi\varpi_D)]$ $\varpi_j = 1 + \frac{x_j - 1}{4}$	$[-10, 10]^D$	0
Schwefel: $f_{10}(x) = 418.9829 * D - \sum_{j=1}^{D} x_j sin(\sqrt{\lvert x_j \rvert})$	$[-500, 500]^D$	0

Note: D represents the dimensions of benchmark function, $j \in [1, D]$.

Table 5.6 Parameters setting on fixed-dimension multimodal functions

Experiment A	Algorithm	Parameters
Population=100 Runs=50	GBPSO	$c1 = c2 = 2.05$, $\chi = 0.72984$
	DPO	$c = 2$, $w \in [0.4, 0.9]$, $teamsize = 10$
	FPO	$c = 2$, $w \in [0.4, 0.9]$
	DFPO	$c = 2$, $w \in [0.4, 0.9]$, $teamsize = 30$

Table 5.7 Parameters setting on six flexible dimensional functions

Experiment B	Algorithm	Parameters
Population=50 Runs=20	LBPSO	$c1 = c2 = 2.05$, $\chi = 0.72984$
	DPO	$c = 2$, $w \in [0.4, 0.9]$, $teamsize = 5$
	FPO	$c = 2$, $w \in [0.4, 0.9]$
	DFPO	$c = 2$, $w \in [0.4, 0.9]$, $teamsize = 15$
	DFPO-r	$c = 2$, $w \in [0.4, 0.9]$, $teamsize \in [10, 30]$

Table 5.8 Statistical results of optimization errors on fixed-dimension multimodal functions

Alg.	Iter.		f_1	f_2	f_3	f_4
GBPSO	20	Best	3.00e-04	1.30e-05	1.09e+00	6.17e-06
		Mean	2.90e-03	1.77e-02	6.11e+00	3.70e-03
DPO	20	Best	**0.00e+00**	**0.00e+00**↓	5.33e-02	2.31e-06
		Mean	**0.00e+00**	**9.55e-06**↓	4.09e+00	8.00e-03
FPO	20	Best	**0.00e+00**	1.30e-05	2.09e-02	3.59e-06
		Mean	1.00e-04	5.00e-04	6.67e+00	8.30e-03
DFPO	20	Best	**0.00e+00**	1.30e-05	**4.00e-03**↓	**1.28e-06**↓
		Mean	**0.00e+00**	1.13e-04	**1.07e+00**↓	**3.20e-03**↓
GBPSO	50	Best	0.00e+00	1.30e-05	5.00e-04	4.04e-10
		Mean	0.00e+00	1.30e-05	3.04e+00	**1.09e-06**↓
DPO	50	Best	0.00e+00	**0.00e+00**↓	2.00e-04	**7.35e-13**↓
		Mean	0.00e+00	**0.00e+00**↓	1.41e+00	5.30e-03
FPO	50	Best	0.00e+00	1.30e-05	7.50e-03	1.78e-07
		Mean	0.00e+00	1.30e-05	2.94e+00	2.98e-04
DFPO	50	Best	0.00e+00	1.30e-05	**0.00e+00**↓	1.82e-08
		Mean	0.00e+00	1.30e-05	**1.40e-03**↓	1.17e-04
GBPSO	100	Best	0.00e+00	1.30e-05	0.00e+00	1.33e-14
		Mean	0.00e+00	1.30e-05	2.65e+00	**6.42e-12**↓
DPO	100	Best	0.00e+00	**0.00e+00**↓	0.00e+00	**0.00e+00**↓
		Mean	0.00e+00	**0.00e+00**↓	1.26e+00	4.80e-03
FPO	100	Best	0.00e+00	1.30e-05	0.00e+00	5.85e-09
		Mean	0.00e+00	1.30e-05	1.83e+00	5.76e-05
DFPO	100	Best	0.00e+00	0.00e+00	0.00e+00	1.83e-09
		Mean	0.00e+00	0.00e+00	**1.00e-04**↓	1.36e-05

Table 5.9 Statistical results of optimization errors on $f_5 - f_7$ after 20 trials of 10^3, 2×10^3 function evaluations, respectively

Alg.	10-D	f_5	f_6	f_7
LBPSO	Best	3.15e-02	2.00e+00	7.40e-03
	Worst	4.41e+00	9.97e+00	4.43e-02
	Mean	**1.83e+00**↓	6.02e+00	1.64e-02
DPO	Best	2.48e-02	0.00e+00	2.46e-02
	Worst	3.97e+00	2.69e+00	6.60e-01
	Mean	2.97e+00	4.98e-01	2.28e-01
FPO	Best	2.20e-03	0.00e+00	1.11e-16
	Worst	1.84e+01	0.00e+00	2.21e-02
	Mean	2.81e+00	0.00e+00	5.90e-03
	Time	4.28e+00	4.31e+00	4.61e+00
DFPO	Best	**4.91e-04**↓	0.00e+00	**0.00e+00**
	Worst	1.72e+01	0.00e+00	1.23e-02
	Mean	2.14e+00	0.00e+00	1.20e-03
	Time	5.26e+00	6.99e+00	5.61e+00
DFPO-r	Best	8.70e-03	0.00e+00	0.00e+00
	Worst	7.04e+00	0.00e+00	1.23e-02
	Mean	2.49e+00	**0.00e+00**↓	**2.73e-03**↓
Alg.	50-D	f_5	f_6	f_7
LBPSO	Best	3.40e+01	1.50e+02	9.85e-12
	Worst	1.36e+02	2.78e+02	7.40e-03
	Mean	6.74e+01	2.29e+02	4.00e-04
DPO	Best	9.64e+01	5.12e+01	3.66e-05
	Worst	2.69e+03	2.13e+02	5.81e-01
	Mean	3.83e+02	1.10e+02	2.48e-02
FPO	Best	**1.60e-02**↓	4.34e-09	7.57e-11
	Worst	1.43e+02	5.97e+00	2.83e-09
	Mean	1.70e+01	1.95e+00	7.62e-10
	Time	8.67e+00	8.96e+00	9.62e+00
DFPO	Best	4.25e-02	8.64e-11	1.98e-14
	Worst	9.84e+01	3.99e+00	5.18e-12
	Mean	**1.49e+01**↓	1.41e+00	3.25e-13
	Time	1.12e+01	1.15e+01	1.21e+01
DFPO-r	Best	1.15e-01	2.26e-12	**1.11e-16**↓
	Worst	7.97e+01	3.80e-09	2.03e-12
	Mean	2.50e+01	**2.49e-10**↓	**1.13e-13**↓

Table 5.10 Statistical results of optimization errors on $f_8 - f_{10}$ after 20 trials of 10^3, 2×10^3 function evaluations, respectively

Alg.	10-D	f_8	f_9	f_{10}
LBPSO	Best	4.71e-14	**0.00e+00**	2.37e+02
	Worst	1.01e-12	0.00e+00	9.51e+02
	Mean	3.98e-13	**0.00e+00**	5.64e+02
DPO	Best	**4.44e-15**	**0.00e+00**	1.18e+02
	Worst	8.00e-15	0.00e+00	1.19e+03
	Mean	**6.04e-15**↓	**0.00e+00**	6.60e+02
FPO	Best	2.60e-12	6.42e-17	1.27e-04
	Worst	2.55e-10	3.08e-16	3.55e+02
	Mean	4.09e-11	2.33e-16	2.11e+02
	Time	4.64e+00	5.08e+00	4.56e+00
DFPO	Best	8.00e-15	5.16e-17	1.27e-04
	Worst	2.00e-14	2.28e-16	1.27e-04
	Mean	1.41e-14	1.03e-16	1.27e-04
	Time	5.69e+00	5.99e+00	1.06e+01
DFPO-r	Best	**4.44e-15**	6.16e-17	1.27e-04
	Worst	8.00e-15	1.09e-16	1.27e-04
	Mean	7.64e-15	8.41e-17	1.27e-04
Alg.	50-D	f_8	f_9	f_{10}
LBPSO	Best	2.00e-04	2.56e-05	5.68e+03
	Worst	1.95e+00	3.99e+01	8.71e+03
	Mean	3.93e-01	1.05e+01	7.49e+03
DPO	Best	7.70e-03	1.20e-03	4.65e+03
	Worst	1.06e+01	3.45e-01	1.14e+04
	Mean	2.34e+00	7.08e-02	7.20e+03
FPO	Best	1.75e-05	3.21e-12	1.78e+03
	Worst	1.32e-04	1.66e-09	2.49e+03
	Mean	6.27e-05	1.09e-10	2.23e+03
	Time	9.60e+00	1.22e+01	9.44e+00
DFPO	Best	1.74e-06	4.11e-07	1.18e+03
	Worst	8.63e-06	2.27e-06	1.90e+03
	Mean	4.08e-06	9.54e-07	1.61e+03
	Time	1.21e+01	1.21e+01	1.19e+01
DFPO-r	Best	**3.80e-08**↓	**2.55e-15**↓	6.36e-04
	Worst	1.59e-07	1.16e-11	3.55e+02
	Mean	**1.00e-07**↓	**1.63e-12**↓	**9.51e+01**↓

Table 5.11 Statistical results of optimization errors on 100-Dimension $f_5 - f_7$ after 20 trials of 2×10^3 function evaluations

Alg.	100-D	f_5	f_6	f_7
LBPSO	Best	3.81e+02	4.97e+02	1.65e-05
	Worst	9.23e+02	7.28e+02	9.00e-04
	Mean	5.65e+02	6.43e+02	8.77e-05
DPO	Best	9.50e+02	2.09e+02	1.53e-02
	Worst	3.34e+03	3.88e+02	9.99e-01
	Mean	1.86e+03	2.88e+02	2.30e-01
FPO	Best	2.89e-01	6.13e+00	2.79e-05
	Worst	1.40e+02	4.16e+01	1.30e-03
	Mean	**3.13e+01**↓	2.26e+01	5.42e-04
	Time	8.85e+00	9.17e+00	1.02e+01
DFPO	Best	**1.23e-01**↓	1.54e+01	1.60e-06
	Worst	1.39e+02	2.52e+01	3.45e-04
	Mean	3.50e+01	2.05e+01	7.88e-06
	Time	1.22e+01	1.31e+01	1.32e+01
DFPO-r	Best	2.02e+00	**3.07e-08**↓	**1.70e-08**↓
	Worst	2.12e+02	9.04e+00	2.03e-07
	Mean	7.68e+01	**3.42e+00**↓	**1.03e-07**↓

Table 5.12 Statistical results of optimization errors on 100-Dimension $f_8 - f_{10}$ after 20 trials of 2×10^3 function evaluations

Alg.	100-D	f_8	f_9	f_{10}
LBPSO	Best	1.95e+00	6.41e+01	1.48e+04
	Worst	6.97e+00	1.44e+02	2.17e+04
	Mean	2.76e+00	9.11e+01	1.90e+04
DPO	Best	2.16e+00	1.71e+00	1.22e+04
	Worst	1.34e+01	1.07e+01	2.67e+04
	Mean	5.78e+00	4.01e+00	2.04e+04
FPO	Best	3.10e-03	**1.06e-08**↓	4.91e+03
	Worst	2.17e+00	2.29e+00	6.98e+03
	Mean	9.23e-01	2.17e-01	5.87e+03
	Time	1.00e+01	1.47e+01	9.77e+00
DFPO	Best	6.40e-03	1.40e-02	4.51e+03
	Worst	1.25e+00	1.91e-01	6.10e+03
	Mean	5.94e-01	6.15e-02	5.20e+03
	Time	1.31e+01	1.32e+01	1.32e+01
DFPO-r	Best	**1.01e-03**↓	5.70e-08	**1.94e+03**↓
	Worst	4.96e-03	3.95e-06	3.46e+03
	Mean	**2.36e-03**↓	**6.03e-07**↓	**2.82e+03**↓

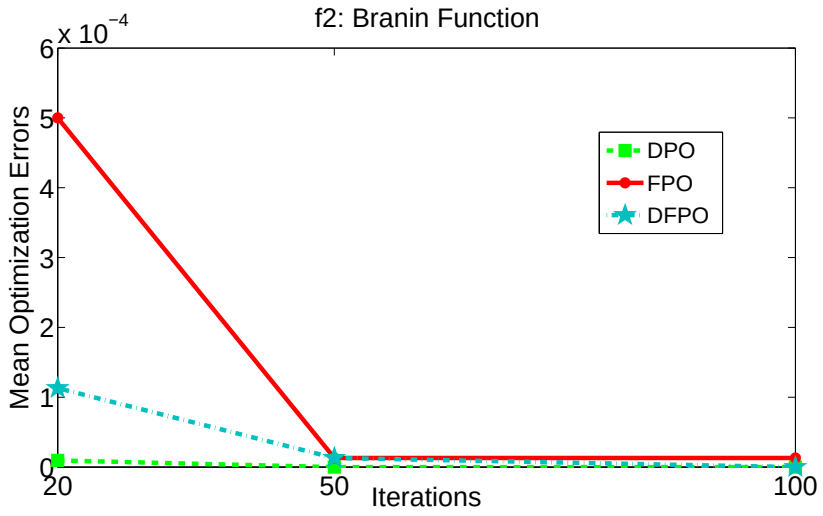


Fig. 5.7 Mean optimization errors of Branin function with three algorithms in 20, 50 and 100 times iteration respectively

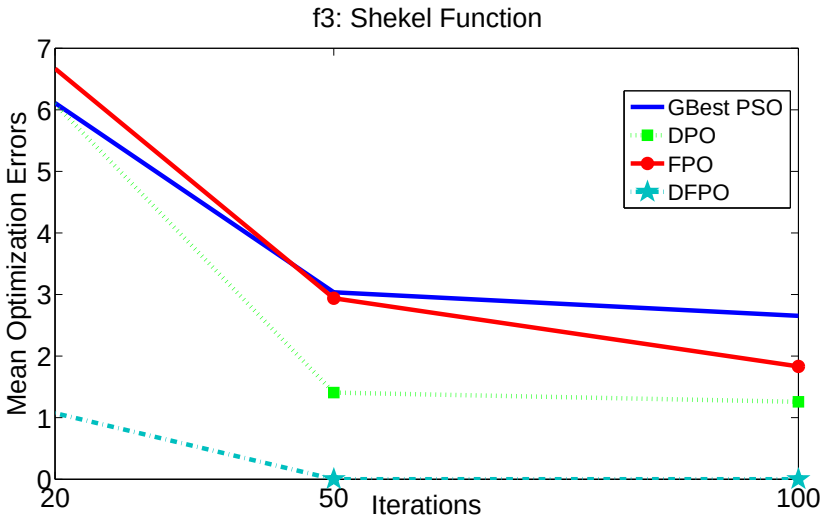


Fig. 5.8 Mean optimization errors of Shekel function with four algorithms in 20, 50 and 100 times iteration respectively

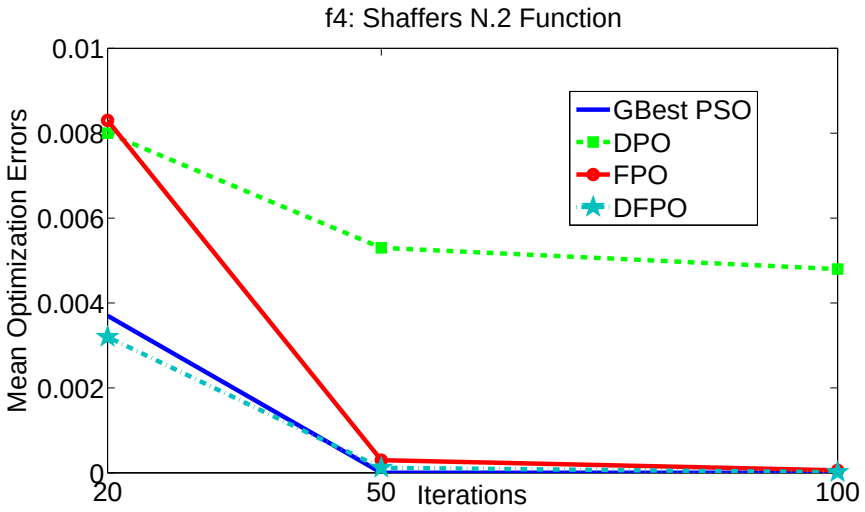


Fig. 5.9 Mean optimization errors of Shaffers N.2 function with four algorithms in 20, 50 and 100 times iteration respectively

function and Shaffers N.2 function respectively. One of the possible reasons is that competitions of predators in DFPO increase individual independence and reduce rapid social collaboration on multimodal functions. These characters of DFPO are able to provide the excellent optimal solution and utilizing local minima avoidance simultaneously.

In Table 5.13 and Table 5.14, the results of ranking are grouped by functions, dimensions, the best global solutions and mean of optimum solutions, respectively. From the total statistic data at the bottom of the table 5.13 and Table 5.14 which can be observed that DFPO-r is able to provide very competitive results both on the best solution quality and on the average of best optimum solution quality even on the flexible dimensional functions. Specifically, as the dimensionality of the multimodal function is increased to 100, DFPO-r successfully demonstrates the superior scalability over all the compared algorithms. It reveals that DFPO-r has significant performance for high dimensional multimodal problem.

Figure 5.10 shows the growth rate of mean global optimum of DFPO to FPO, DFPO-r to FPO and the growth rate of computation time of DFPO to FPO. The computation time is only focused on DFPO and FPO because the computation time of DFPO-r is very close to

Table 5.13 Ranking of the best solution quality obtained by Table 5.9, Table 5.10, Table 5.11 and Table 5.12 with 5 kinds of algorithms on $f_5 - f_{10}$

Fun.	Dim.	Best				
		PSO	DPO	FPO	DFPO	DFPO-r
f_5	10	5	4	2	1	3
	50	4	5	1	2	3
	100	4	5	2	1	3
f_6	10	5	2.5	2.5	2.5	2.5
	50	5	4	3	2	1
	100	5	4	2	3	1
f_7	10	4	5	3	1.5	1.5
	50	3	5	4	2	1
	100	3	5	4	2	1
f_8	10	4	1.5	5	3	1.5
	50	4	5	3	2	1
	100	4	5	2	3	1
f_9	10	1.5	1.5	5	3	4
	50	4	5	2	3	1
	100	5	4	1	3	2
f_{10}	10	5	4	2	2	2
	50	5	4	3	2	1
	100	5	4	3	2	1
Total	Median	4	4	2.75	2	1.25

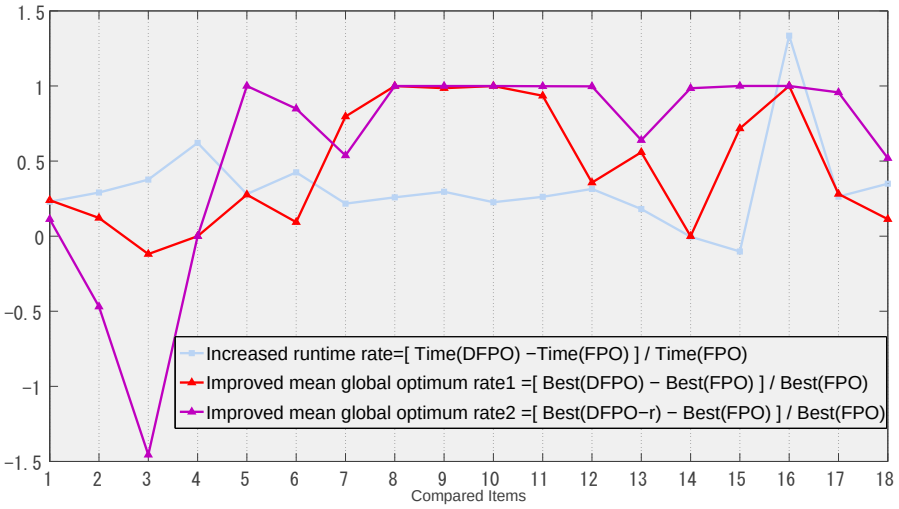


Fig. 5.10 The enhanced rate of mean global optimum and increased time rate

Table 5.14 Ranking of the averaged best solution quality obtained by Table 5.9, Table 5.10, Table 5.11 and Table 5.12 with 5 kinds of algorithms on $f_5 - f_{10}$

Fun.	Dim.	Mean				
		PSO	DPO	FPO	DFPO	DFPO-r
f_5	10	1	5	4	2	3
	50	4	5	2	1	3
	100	4	5	1	2	3
f_6	10	5	4	2	2	2
	50	5	4	3	2	1
	100	5	4	3	2	1
f_7	10	4	5	3	2	1
	50	4	5	3	2	1
	100	3	5	4	2	1
f_8	10	4	1	5	3	2
	50	4	5	3	2	1
	100	4	5	3	2	1
f_9	10	1.5	1.5	5	4	3
	50	5	4	2	3	1
	100	5	4	3	2	1
f_{10}	10	4	5	3	1.5	1.5
	50	5	4	3	2	1
	100	4	5	3	2	1
Total	Median	4	5	3	2	1

DFPO's. In Fig.5.10, 18 kinds of comparable items come from six multimodal functions on 10, 50 and 100 dimensions, respectively. As shown in Fig.5.10, the improved global solution ratio of DFPO-r is better than DFPO in most cases. It reveals that the dynamic team size selection method for DFPO-r is validity for multimodal problems.

To summarize, the excellent global optimizing abilities of DFPO and DFPO-r show that the paralleled information-exchange system constructed by the dynamic virtual teams is able to accelerate the early convergence rate and improve the global searching capability. It should be noted that during the optimization process of of DFPO and DFPO-r, the computation time of both them are slightly increased. However, the increasing space is limited within the reasonable range.

5.5 Summary

To avoid premature problem and enhance the capability of global exploration for multimodal problems, a new algorithm DFPO is proposed in this chapter, to build a paralleled information-exchange system based on dynamic virtual team. The dynamic virtual team is a self-organization team that according to the nearest neighbour clustering. Each individual automatically is denoted as the team leader. The main function of the team leader is to obtain the neighbour team's best experience. It is worth noting that one of team members may belong to another group concurrently. The expansion of individual best experience and the team best experience is carried out by this kind of multiple memberships. Consequently an improved Dynamic Fitness Predator Optimization (DFPO) is proposed to strengthen the global searching effectiveness. We extensively evaluated the new approach and compared it with FPO, Dolphin Partner Optimization (DPO) and Standard Particle Swarm Optimization (SPSO), with respect to both running time and global solution capability. Experimental results show that the proposed DFPO could demonstrate the desirable performance for multimodal problems.

However, fixed team size is not suitable for all of the real situations. The strategy of team size selection is also presented in this chapter based on the idea that a team size with a higher population diversity is able to prevent solutions from clustering too tightly in the local search space. Then DFPO with dynamic team size selection strategy is utilized in DFPO-r. In DFPO-r, two kinds of team sizes are randomly selected from the sizes pool, which is designed by a real situation. Comparing their population diversities, the one with higher population diversity will be kept as the current team size. A paralleled information-exchange system is created by these dynamic virtual teams. Which is also able to enhance the global optimal capability and speed up convergence rate during the process of searching. To evaluate the performance of DFPO-r and DFPO, ten well-known benchmark functions are used to compare with GBPSO, LBPSO, DPO and FPO. Experimental results demonstrate that both DFPO-r and DFPO have desirable performances for multimodal functions. In addition, DFPO-r shows better robust performance in most cases compared with DFPO.

Chapter 6

Modified Bare Bones Particle Swarms for Large Scale Global Optimization

In today's digital world, with ever increasing amounts of readily-available data comes the need to solve optimization problems of unprecedented sizes. Machine learning, compressed sensing, natural language processing, truss topology design and computational genetics are some of many prominent application domains where it is easy to formulate optimization problems with tens of thousands or millions of variables. To the best of the author's knowledge, a number of modified Particle Swarm Optimization algorithms have been suggested to solve this large scale global optimization. For instance, EPUS-PSO [35], a variation of the traditional PSO algorithm, which is proposed to solve high-dimensional problems by adjusting the population size to enhance the searching ability and drive particles more efficiently. DMS-PSO [55], a dynamic multi-swarm PSO algorithm, which is proposed to dynamically change the neighbourhood structure for a higher degree of swarm diversity. However, the performance of these algorithms are influenced by their control parameters. The best values for the control parameters remain problem dependent, and need to be tuned for each problem. The Bare Bones Particle Swarms Optimization (BBPSO) is a simple and parameter-free algorithm where the positions of particles are sampled from Gaussian distribution. Although

BBPSO is extensively used in literature, it also suffers premature convergence problem. In this chapter, a new modified Bare Bones Particle Swarms Optimization employed Differential Evolution strategy (BBPSO-DE) is proposed for large scale optimization.

6.1 Overview and Preliminaries

The bare bones Particle Swarms Optimization (BBPSO) proposed by Kennedy [45] is a simple, control parameter free algorithm. Similarly, BBPSO also faces premature convergence problem, and can be trapped easily at local optimum when the global best position $gbest$ of the whole swarm couldn't be updated any more. BBPSO-MC-lbest is a new improved BBPSO algorithm incorporating mutation and crossover operators of Differential Evolution, which has achieved better performance for some application problems [106], [105].

6.1.1 BBPSO-MC-lbest Algorithm

In this part, we basically introduce the BBPSO integrated Differential Evolution strategy in $lbest$ topology, BBPSO-MC-lbest algorithm, which has achieved better performance for the phase equilibrium, chemical equilibrium and phase stability application problems [105].

It is worth noting that the standard deviation $\sigma(|\boldsymbol{p}_i - \boldsymbol{p}_g|)$ is an important parameter for scaling the amplitude of particles' trajectories. If the standard deviation is zero, that is the current particle's $\boldsymbol{p}_i$ position happens to be the same as that of $\boldsymbol{p}_g$, then the stochastic mean μ is the value of the particle's $pbest$ position itself. In this case, the $gbest$ of the whole swarm will not be updated. Consequently, the algorithm faces premature convergence problem, and can be trapped easily at local optimum. It seems reasonable that the performance of BBPSO could be improved by redefining the standard deviation between the current particle's $pbest$ and $gbest$ points. A simplified approach to get around this is to assign a small constant (such as 0.001) for the standard deviation when it equals to zero. However, this implies that the

extent of exploration and exploitation for *gbest* position will be constant in the remaining iterations. In order to have an adaptive balance between exploration and exploitation search nature, a new BBPSO incorporating mutation and crossover operators of Differential Evolution (DE) [68] is confirmed to be a very competitive Optimization for the standard Gaussian version. In BBPSO-MC-lbest algorithm, mutation and crossover operators of DE are employed for updating $\boldsymbol{p}_l$ when the current particle's $\boldsymbol{p}_i$ objective function value happens to be the same as that of $\boldsymbol{p}_l$:

$$\boldsymbol{p}_l = \begin{cases} \boldsymbol{p}_{i_1} + 0.5 * (\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_3}) & \text{if } U(0,1) < 0.5 \\ \boldsymbol{p}_l & \text{otherwise} \end{cases} \tag{6.1}$$

Where $i \neq i_1 \neq i_2 \neq i_3$, and $\boldsymbol{p}_{i_1}, \boldsymbol{p}_{i_2}, \boldsymbol{p}_{i_3}$ are randomly chosen from the previous best positions. $U(0,1)$ is a function that generates uniformly distributed random numbers between zero and one. There is a 50% chance that the $\boldsymbol{p}_l$ is updated by the mutation operator of DE, whose information comes from the randomly chosen previous best positions other than the current particle's previous *lbest* position. In this case, the mutation and crossover rates of 0.5 each are implemented without any selection operation of DE. Furthermore, based on the equation (6.1), there is also a 50% possibility that the $\boldsymbol{p}_l$ keeps the previous position. If the standard deviation σ approaches zero, that is the current particle's *pbest* equals to the *lbest* position, then the σ is set as a small constant (0.001) according to the publication of Zhang et al. [106].

In short, the performance of BBPSO could be improved by updating the *lbest* position with mutation and crossover operators of DE and setting the standard deviation σ to a small fixed number when the difference of *pbest* and *lbest* reaches zero. The main function of BBPSO-MC-lbest can be summarized as shown in Algorithm 12. Algorithm 12 demonstrates the main function of BBPSO-MC-lbest. The algorithm checks the objective value of $\boldsymbol{p}_i$ if it is the same as that of $\boldsymbol{p}_l$, the *lbest*-index particle is updated using equation (6.1). Other

Algorithm 12 Main Function of BBPSO-MC-lbest

Initialize particles $\boldsymbol{x}_i$: $\boldsymbol{x}_i = rand() \in (\boldsymbol{x}_{min}, \boldsymbol{x}_{max})$, $i \in [1..n]$
Initialize personal best particles $\boldsymbol{p}_i, i \in [1..n]$
Initialize local best particles $\boldsymbol{p}_l, l \in [1..n]$
for each particle $\boldsymbol{x}_i$ **do**
  if $Fitness(\boldsymbol{p}_i) = Fitness(\boldsymbol{p}_l)$ **then**
    Randomly set $i_1, i_2, i_3.(i \neq i_1 \neq i_2 \neq i_3)$
    if $U[0,1] < 0.5$ **then**
      $\boldsymbol{p}_l = \boldsymbol{p}_{i_1} + 0.5 * (\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_3})$
    end if
  end if
  $\mu = \dfrac{\boldsymbol{p}_i + \boldsymbol{p}_l}{2}, \sigma = |\boldsymbol{p}_i - \boldsymbol{p}_l|$
  if $\sigma = 0$ **then**
    $\sigma = 0.001$
  end if
  $\boldsymbol{x}_i = \begin{cases} N(\mu, \sigma) & \text{if } U(0,1) < 0.5 \\ \boldsymbol{p}_i & otherwise \end{cases}$
  if $\boldsymbol{x}_i < x_{min}$ OR $\boldsymbol{x}_i > x_{max}$ **then**
    $\boldsymbol{x}_i = x_{min} + U[0,1] * (x_{max} - x_{min})$
  end if
end for

particles are updated using BBExp algorithm as shown in equation (2.9). If the position of the particle happens to be the same as that of $\boldsymbol{p}_l$, then σ is assigned a value of 0.001. The *lbest* swarm model, often referred to as a local topology is an open issue to BBPSO-MC-lbest algorithm. There are number of different local topologies such as ring, wheel, or won Neumann have been mentioned by some publications [44], [16], [49]. In [106], three kinds of local topology are tested for unimodal and multimodal functions with neighbourhood size of 2, 4 and 6 respectively. Experimental results show that BBPSO-MC-lbest with 2 neighbours perform relatively better for majority of the benchmark functions, especially for the multimodal functions. Furthermore, as the size of neighbourhood increases from 2 to 6 for BBPSO-MC-lbest, the mean value moves closer to that obtained by BBPSO-MC-gbest. This implies that increased neighbourhood size results in higher extent of local exploitation than global exploration. It is important to note that these experimental analyses are based on the 30-dimensional benchmark functions in [106]. In this chapter, six widely used benchmark functions are extensively set to 1000 dimensions to investigate the performance of BBPSO-MC-lbest. However, experimental results show that the performance of BBPSO-MC-lbest with 2 neighbours has no advantages compared with BBPSO-MC-gbest. So in the next section, a new BBPSO-DE algorithm is proposed for large scale optimization.

6.1.2 Solutions

This chapter empirically analysed the BBPSO-MC-lbest and proposed an improved BBPSO-DE algorithm using three strategies for the Large Scale Global Optimization (LSGO) problems. First of all, the ring topologies of *explorer-swarm* and *memory-swarm* are defined for the BBPSO-DE. Specifically, the *memory-swarm* using ring topology is proposed to improve the swarm diversity. Each particle defined in the BBPSO-DE possesses local memory including its personal best memory $\boldsymbol{p}_i$ and its local best memory $lbest_i$. The particles can explore the search space more broadly by using the local *memory-swarm* and form a sta-

ble network retaining the best positions found so far, which is more likely to strengthen the global exploration while properly maintain the local exploitation. Secondly, the mutation and crossover operators of Differential Evolution (DE) are employed for updating the local memory $lbest_i$ when the current particle's $\boldsymbol{p}_i$ objective function value happens to be the same as that of $lbest_i$. The global search performance can be improved by updating the $lbest_i$ with the DE strategy. However, there is also a 50% possibility that the $lbest_i$ keeps the previous position. It seems reasonable that the search performance of BBPSO-DE can be improved by redefining the mean μ and the standard deviation σ. Thus, the center position of all particles $\overline{X}$ is proposed to replace the $\boldsymbol{p}_i$. In this case, the new definitions of the mean μ and the standard deviation σ maintain the swarm diversity by increasing the particles' variance. The empirical study reveals that the proposed strategies of BBPSO-DE are capable of inducing individual independent behaviours which are robust and effective premature in avoidance.

6.2 Proposal of BBPSO-DE for LSGO Problems

In this section, a simple yet effective BBPSO-DE algorithm [98] using the ring neighbourhood topology, which does not require any parameters, is proposed for large scale global optimization.

6.2.1 Ring Topology of BBPSO-DE

Niching is an important technique for large scale multimodal optimization. Reference [54] demonstrates that an *lbest* PSO using a ring topology is able to induce stable niching behaviours. The definition originally introduced by M.Clerc [15] is that a swarm can be viewed as a combination of *explorer-swarm* and *memory-swarm* according to their differences in functionality. The *explorer-swarm* is composed of particles moving around in large step sizes and more frequently, each particle strongly influenced by its velocity and previous po-

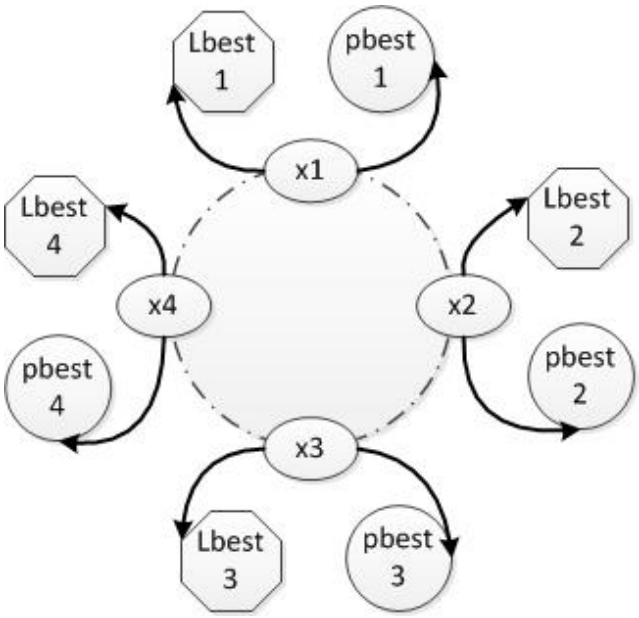


Fig. 6.1 Ring topology of *explorer-swarm*

sition. The *memory-swarm* consists of personal bests of all particles. The *memory-swarm* is more effective in retaining better positions found so far by the swarm as a whole. In this part, the ring topologies of *explorer-swarm* and *memory-swarm* used by BBPSO-DE are described respectively.

Figure 6.1 shows an example of *explorer-swarm* using a ring topology with a population of four particles. Here the *explorer-swarm* consists of particles ($\boldsymbol{x}_i$) as marked from numbers 1 to 4. Each member interacts only with its immediate left and right neighbours. The *explorer-swarm*, sampled from Gaussian distribution, is more effective in exploring the widely available search space. It is important to note that each particle defined in the BBPSO-DE possesses local memory including its personal best memory $\boldsymbol{p}_i$ and its local best memory $lbest_i$.

Figure 6.2 shows the *memory-swarm* using ring topology with a population of four personal best memory ($\boldsymbol{p}_i$) and four local best memory ($lbest_i$). Each $lbest_i$ has three informants, from two immediate neighbouring particles' best memory and its own best memory. It also illustrates the relationship between $\boldsymbol{p}_i$ and $lbest_i$. Figure 6.3 describes the detailed proce-

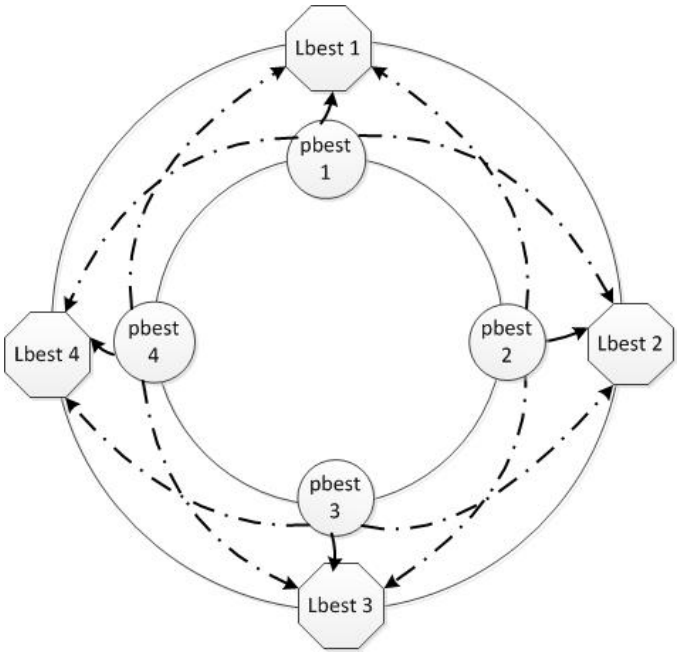


Fig. 6.2 Ring topology of *memory-swarm*

dure of how to achieve the $lbest$ swarm. For instance, we suppose that $pbest_n$ represents the left neighbour of $pbest_1$, and $pbest_2$ shows the right neighbour. The best one from the three informants is chosen as $lbest_1$. Similarly, a swarm of $lbest_i$ is created by chosen the best one from the $Lpbest$ swarm, the $pbest$ swarm and the $Rpbest$ swarm.

6.2.2 Integrated BBPSO-DE with Differential Evolution

The mutation and crossover operators of DE are employed for updating $lbest_i$ when the current particle's $\boldsymbol{p}_i$ objective function value happens to be the same as that of $lbest_i$:

$$lbest_i = \begin{cases} \boldsymbol{p}_{i_1} + 0.5 * (\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_3}) & \text{if } U(0,1) < 0.5 \\ lbest_i & \text{otherwise} \end{cases} \tag{6.2}$$

Where $i \neq i_1 \neq i_2 \neq i_3$, and $\boldsymbol{p}_{i_1}, \boldsymbol{p}_{i_2}, \boldsymbol{p}_{i_3}$ are randomly chosen from the previous personal best positions. $U(0,1)$ is a function that generates uniformly distributed random numbers between zero and one. From equation 6.2, we can see that there is a 50% chance that the $lbest_i$

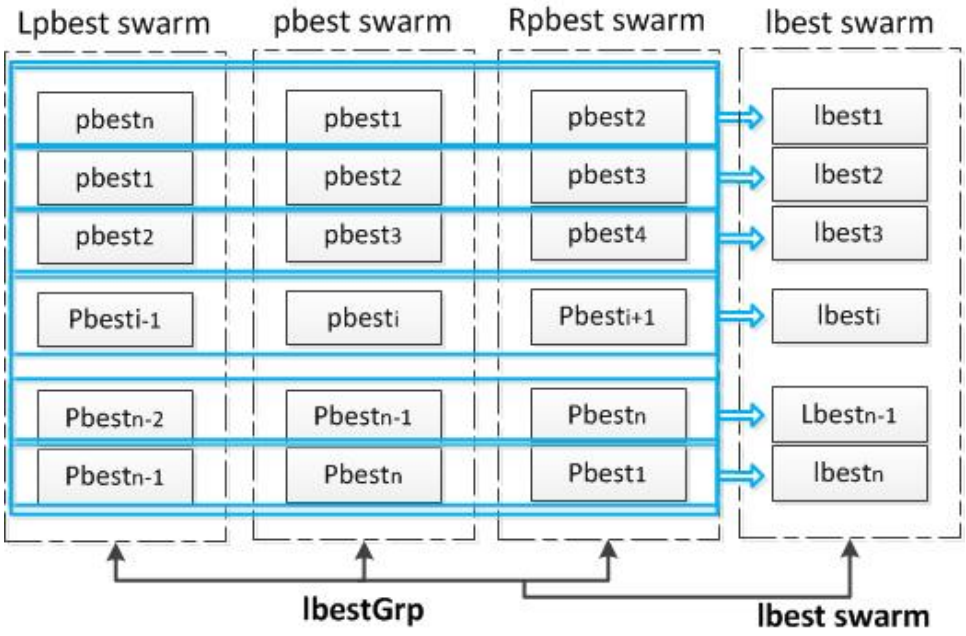


Fig. 6.3 Group three swarms as one team swarm

is updated by the mutation operator of DE, whose information comes from the randomly chosen previous best positions. However, there is a 50% possibility that the $lbest_i$ keeps the previous position. If the standard deviation σ approaches zero, that is the current particle's *pbest* equals to the *lbest* position, then the global best position of the whole swarm will not be updated. Thus, the new definitions of the mean μ and the standard deviation σ is developed to maintain the swarm diversity by increasing the particles' variance.

6.2.3 Redefining the Sample Equations

In general, the particles can explore the search space more broadly by using individual's local *memory-swarm* to form a stable network retaining the best positions found so far. In particularly for the $lbest_i$ shown in Fig.6.2, which is more likely to strengthen the global exploration capability while successfully maintain the local exploitation functionality. In BBPSO-DE, we try to redefine the equations of mean μ and standard deviation σ with $lbest_i$ and $\overline{X}$. Here is the new equations used in BBPSO-DE.

$$\mu = \frac{lbest_i + \overline{X}}{2} \tag{6.3}$$

$$\sigma = |lbest_i - \overline{X}| \tag{6.4}$$

$$x_i = \begin{cases} N(\mu, \sigma) & \text{if } U(0,1) < 0.5 \\ lbest_i & \text{otherwise} \end{cases} \tag{6.5}$$

Where $\overline{X}$ denotes the centroid position of all particles. The swarm's centroid position is first used in Competitive Swarm Optimization (CSO) [12], which is a novel swarm intelligence algorithm for large scale optimization. The motivation for us to introduce centroid position in BBPSO-DE is to increase swarm diversity, which potentially enhances the global search capability of BBPSO-DE.

6.2.4 Pseudocode of BBPSO-DE

BBPSO-DE is a variant of BBPSO-MC-lbest. The differences between them are the new equations for the Gaussian distribution and ring topology of BBPSO-DE. The main function of BBPSO-DE is demonstrated in Algorithm 13. In algorithm 13, all the different lines with BBPSO-MC-lbest are marked with underline. As shown in lines 13, 15 and 20, a new method is proposed to calculate the Gaussian distribution. We consider that this method is useful to improve the search performance of BBPSO-DE.

In order to investigate the performance of BBPSO-DE, we present here a compared algorithm named Modified Competitive Particle Swarm Optimization (MCPSO). Which can be seemed as a variant of PSO used for large scale optimization. In MCPSO, each particle only has one specified neighbourhood. The updated rules are borrowed from CSO [12].

Algorithm 13 Pseudocode of BBPSO-DE

Initialize particles $\boldsymbol{x}_i$: $\boldsymbol{x}_i = rand() \in (\boldsymbol{x}_{min}, \boldsymbol{x}_{max}), i \in [1..n]$
Initialize personal best particles $\boldsymbol{p}_i, i \in [1..n]$
Initialize local best particles $lbest_i, i \in [1..n]$
Initialize iteration counter, $t = 0, t_{max} = Maxiter$
while $t < Maxiter$ **do**
  for each particle $\boldsymbol{x}_i$ **do**
    if $Fitness(\boldsymbol{p}_i) = Fitness(lbest_i)$ **then**
      Randomly set $i_1, i_2, i_3.(i \neq i_1 \neq i_2 \neq i_3)$
      if $U[0,1] < 0.5$ **then**
        $lbest_i = \boldsymbol{p}_{i_1} + 0.5 * (\boldsymbol{p}_{i_2} - \boldsymbol{p}_{i_3})$
      end if
    end if
    Get the mean value of all particles: $\overline{X}$
    $\mu = \frac{lbest_i + \overline{X}}{2}, \sigma = |lbest_i - \overline{X}|$
    if $\sigma = 0$ **then**
      $\sigma = 0.001$
    end if
    $\boldsymbol{x}_i = \begin{cases} N(\mu, \sigma) & \textbf{if } U(0,1) < 0.5 \\ lbest_i & \textbf{otherwise} \end{cases}$
    if $\boldsymbol{x}_i < x_{min}$ OR $x_i > x_{max}$ **then**
      $\boldsymbol{x}_i = x_{min} + U[0,1] * (x_{max} - x_{min})$
    end if
    if $Fitness(\boldsymbol{x}_i) < Fitness(pbest_i)$ **then**
      $pbest_i = \boldsymbol{x}_i$
    end if
    Create *pbest*'s left and right neighbour swarms
    Group three swarms as a team: $lbestGrp$
    $lbest_i = argmin(Fitness(lbestGrp_i))$
  end for
  $t = t + 1$
end while
Output the global optimum of the team $lbestGrp$

$$\boldsymbol{v}_i(t) = r_1 * \boldsymbol{v}_i(t-1) + r_2 * (lbest_i(t-1) - \boldsymbol{x}_i(t)) + r_3 * (\overline{X}(t-1) - \boldsymbol{x}_i(t-1)) \quad (6.6)$$

$$\boldsymbol{x}_i(t) = \boldsymbol{x}_i(t-1) + \boldsymbol{v}_i(t) \quad (6.7)$$

Where t is the current iteration (generation) number, $\boldsymbol{v}_i(t)$ and $\boldsymbol{x}_i(t)$ represent the velocity and position of the particle i, respectively. r_1, r_2, r_3 are three independent random numbers uniquely generated at every update for each individual dimension in the range [0,1]. If each particle $\boldsymbol{x}_i$ only has one neighbourhood, then its $lbest_i$ can be seemed as a result of pairwise competition. This pairwise competition mechanism has been used by some researchers and showed promising performance on large scale problems.

6.3 Experiment

The aim of the experiment is to compare the global search ability and computation time of BBPSO-DE with GBest PSO, LBest PSO, BBPSO-MC-lbest and MCPSO on various dimentional benchmarks including of unimodal and multmodal problems.

6.3.1 Evaluation Method

All the experiments are implemented on a PC with an Intel Core i7 860, 2.8 GHz CPU and Microsoft Windows 7 Professional SP1 64-bit operating system, and all algorithms are written in Matlab language. The bounds and global minimums of six functions are shown in Table 6.1. These benchmarks were chosen for their variety. Functions f_1--f_3 are continuous, convex and unimodal test problems, f_4--f_6 are highly complex multimodal problems with many local minima. The experimental parameters are shown in Table 6.2.

Table 6.1 The bound and global Minimum of Benchmark Function

Function	Bound	Optimum
Rotated Hyper-Ellipsoid $f_1(x) = \sum_{j=1}^{D} \sum_{k=1}^{j} x_k^2$	$[-65.536, 65.536]^D$	0
Sphere $f_2(x) = \sum_{j=1}^{D} x_i^2$	$[-100, 100]^D$	0
Schwefel 1.2 $f_3(x) = \sum_{j=1}^{D} (\sum_{k=1}^{j} x_k)^2$	$[-100, 100]^D$	0
Ackley $f_4(x) = -a * \exp(-0.02 * (D^{-1} \sum_{j=1}^{D-1} x_j^2)\frac{1}{2})$ $- \exp(D^{-1} \sum_{j=1}^{D} \cos(2\pi x_j)) + a + \exp, a = 20$	$[-32, 32]^D$	0
Rastrigin $f_5(x) = \sum_{j=1}^{D} (x_j^2 - 10 * \cos(2\pi x_j)) + 10 * D$	$[-5.12, 5.12]^D$	0
Griewank $f_6(x) = \frac{1}{4000} \sum_{i=1}^{D} x_j^2 - \prod_{j=1}^{D} \cos(\frac{x_j}{\sqrt{j}}) + 1$	$[-600, 600]^D$	0

Note: D represents the dimensions of benchmark function, $j \in [1, D]$.

Table 6.2 Experimental parameters setting of algorithms

Experiment	Algorithm	Parameters
Poulation=200	GBest PSO	$c1 = c2 = 2.05$, $\chi = 0.72984$
	LBest PSO	$c1 = c2 = 2.05$, $\chi = 0.72984$
	MCPSO	$rand(r1, r2, r3) \in [0, 1]$
Run Number=30	BBPSO-MC-lbest	None
	BBPSO-DE	None

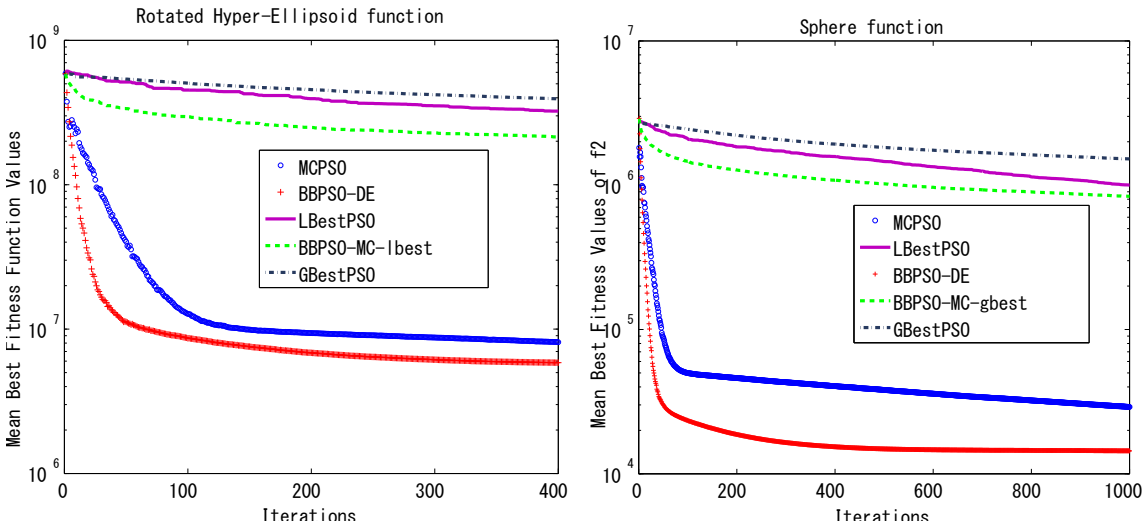


Fig. 6.4 Convergence graphs of f_1 & f_2 on 1000 dimensions

6.3.2 Experimental Results and Discussion

Experimental results including the best optimum error and the mean optimum error over 30 independent trials are displayed in Table 6.3 and Table 6.4. To all the algorithms, the marked result with bold type and down arrow denotes that this algorithm is able to provide the best competitive global optimum compared with others.

The statistical results in Table 6.3 and Table 6.4 show that BBPSO-DE has significantly performance on 300-D, 500-D and 1000-D functions. While, the standard PSO including LBest PSO and GBest PSO have better performance than BBPSO-DE on 30-D and 50-D functions. It reveals that PSO has high search efficiency on relatively low-dimensional problems. However, its performance deteriorates rapidly as the dimensionality of the problem increases. Different with LBest PSO and GBest PSO, BBPSO-DE has shown very good scalability to the search dimension, especially in the high dimensional cases.

In most of cases, BBPSO-MC-lbest shows better performance than LBest PSO and GBest PSO. It seems that DE strategy is able to improve the performance of BBPSO-MC-lbest as the dimensionality of the problem increases. However, the outstanding performance of BBPSO-DE on 500-D and 1000-D test functions should be attributed to the redefined the equations of

Table 6.3 Statistical results of optimization errors on unimodal functions after 30 trials

	Dim.	Iter.	Algorithm	Best	Mean	Std.	Time(s)
f_1	30	400	LBest PSO	1.04e+01	1.70e+01	5.98e+00	3.00e+01
			Gbest PSO	**4.69e-09**↓	1.86e+04	1.63e+04	1.35e+01
			BBPSO-MC-lbest	5.23e+01	6.41e+01	1.61e+01	3.84e+01
			BBPSO-DE	1.93e+00	**3.43e+00**↓	2.17e+00	2.80e+01
f_1	300	400	LBest PSO	1.29e+07	1.37e+07	9.53e+05	1.62e+03
			Gbest PSO	1.30e+07	1.71e+07	3.66e+06	5.64e+02
			BBPSO-MC-lbest	1.02e+07	1.09e+07	9.40e+05	1.69e+03
			BBPSO-DE	**1.76e+05**↓	**2.14e+05**↓	3.38e+04	1.03e+03
f_1	1000	400	LBest PSO	3.07e+08	3.23e+08	1.92e+07	2.34e+04
			Gbest PSO	3.61e+08	3.93e+08	4.62e+07	7.83e+03
			BBPSO-MC-lbest	2.05e+08	2.14e+08	7.62e+06	2.41e+04
			MCPSO	6.68e+06	8.13e+06	1.67e+06	4.03e+02
			BBPSO-DE	**5.57e+06**↓	**5.86e+06**↓	2.61e+05	1.08e+04
f_2	30	400	LBest PSO	9.03e-01	2.66e+00	1.04e+00	3.06e+01
			Gbest PSO	**2.97e-09**↓	**2.36e-08**↓	2.43e-08	1.97e+01
			BBPSO-MC-lbest	4.22e-05	8.23e-05	3.83e-05	1.48e+02
			BBPSO-DE	9.19e-02	1.57e+00	1.64e+00	3.40e+01
f_2	300	1000	LBest PSO	4.60e+04	5.50e+04	5.47e+03	4.76e+02
			Gbest PSO	1.31e+05	1.75e+05	3.09e+04	3.13e+02
			BBPSO-MC-lbest	1.02e+05	1.07e+05	3.00e+03	1.12e+03
			BBPSO-DE	**1.22e+03**↓	**1.62e+03**↓	3.37e+02	1.10e+03
f_2	1000	1000	LBest PSO	9.51e+05	1.00e+06	3.61e+04	1.82e+03
			Gbest PSO	1.33e+06	1.52e+06	9.96e+04	1.18e+03
			BBPSO-MC-lbest	8.11e+05	8.38e+05	1.84e+04	3.67e+03
			MCPSO	2.40e+04	2.91e+04	3.96e+03	1.66e+02
			BBPSO-DE	**1.09e+04**↓	**1.44e+04**↓	4.32e+03	3.65e+03
f_3	30	400	LBest PSO	6.52e+02	2.83e+03	1.93e+03	7.89e+01
			Gbest PSO	**1.70e-01**↓	3.17e+03	2.77e+03	3.31e+01
			BBPSO-MC-lbest	3.98e+03	5.39e+03	8.42e+02	1.00e+02
			BBPSO-DE	2.63e+01	**4.98e+01**↓	2.19e+01	6.54e+01
f_3	300	1000	LBest PSO	1.13e+06	1.32e+06	9.60e+04	3.70e+03
			Gbest PSO	8.72e+05	1.06e+06	1.53e+05	1.26e+03
			BBPSO-MC-lbest	1.07e+06	1.18e+06	9.20e+04	3.84e+03
			BBPSO-DE	**9.81e+03**↓	**1.21e+04**↓	2.05e+03	4.46e+03
f_3	1000	1000	LBest PSO	1.08e+07	1.29e+07	1.14e+06	2.34e+04
			Gbest PSO	3.61e+08	3.93e+08	4.62e+07	5.90e+04
			BBPSO-MC-lbest	1.10e+07	1.28e+07	1.29e+06	5.85e+04
			MCPSO	**1.94e+05**↓	**2.54e+05**↓	4.59e+04	3.38e+03
			BBPSO-DE	5.54e+05	8.66e+05	2741e+05	6.72e+04

Table 6.4 Statistical results of optimization errors on multimodal functions after 30 trials

	Dim.	Iter.	Algorithm	Best	Mean	Std.	Time(s)
f_4	50	1000	LBest PSO	1.19e-01	**5.68e-01**↓	4.66e-01	6.17e+01
			Gbest PSO	**6.81e-08**↓	2.21e+00	3.63e+00	2.96e+01
			BBPSO-MC-lbest	3.56e-01	1.01e+00	4.04e-01	9.46e+01
			BBPSO-DE	6.07e-02	1.32e+00	7.82e-01	9.76e+01
f_4	500	2000	LBest PSO	1.97e+01	2.00e+01	1.02e-01	7.83e+02
			Gbest PSO	1.93e+01	1.95e+01	1.11e-01	3.89e+02
			BBPSO-MC-lbest	1.60e+01	1.65e+01	2.38e-01	1.39e+03
			BBPSO-DE	**4.17e+00**↓	**5.03e+00**↓	6.88e-01	1.34e+03
f_4	1000	2000	LBest PSO	2.00e+01	2.02e+01	1.09e-01	1.86e+03
			Gbest PSO	1.97e+01	1.98e+01	5.30e-02	1.04e+03
			BBPSO-MC-lbest	1.77e+01	1.81e+01	1.30e-01	3.32e+03
			MCPSO	7.04e+00	7.30e+00	1.82e-01	1.57e+03
			BBPSO-DE	**5.13e+00**↓	**6.03e+00**↓	5.98e-01	2.97e+03
f_5	50	1000	LBest PSO	1.45e+02	1.98e+02	2.62e+01	5.54e+01
			Gbest PSO	1.23e+02	2.07e+02	4.80e+01	2.36e+01
			BBPSO-MC-lbest	4.94e+01	6.71e+01	9.55e+00	9.03e+01
			BBPSO-DE	**4.53e-02**↓	**1.41e+00**↓	1.36e+00	8.54e+01
f_5	500	2000	LBest PSO	4.73e+03	5.18e+03	2.29e+02	6.97e+02
			Gbest PSO	4.02e+03	4.26e+03	1.75e+02	3.65e+02
			BBPSO-MC-lbest	2.78e+03	2.98e+03	8.00e+01	1.28e+03
			BBPSO-DE	**5.73e+02**↓	**7.17e+02**↓	1.25e+02	1.19e+03
f_5	1000	2000	LBest PSO	1.13e+04	1.20e+04	3.34e+02	1.57e+03
			Gbest PSO	9.65e+03	1.02e+04	3.50e+02	8.75e+02
			BBPSO-MC-lbest	7.89e+03	8.20e+03	1.39e+02	2.77e+03
			MCPSO	6.89e+03	7.34e+03	3.37e+02	1.48e+03
			BBPSO-DE	**2.28e+03**↓	**2.71e+03**↓	4.30e+02	2.62e+03
f_6	50	1000	LBest PSO	4.28e-02	**8.42e-02**↓	2.18e-02	7.10e+01
			Gbest PSO	**1.22e-15**↓	6.02e+00	2.29e+01	3.23e+01
			BBPSO-MC-lbest	7.23e-01	9.06e-01	7.98e-02	1.07e+02
			BBPSO-DE	2.69e-01	8.32e-01	1.98e-01	1.07e+02
f_6	500	2000	LBest PSO	5.29e+02	5.99e+02	4.31e+01	9.48e+02
			Gbest PSO	2.30e+03	3.31e+03	3.74e+02	4.40e+02
			BBPSO-MC-lbest	1.44e+03	1.59e+03	6.65e+01	1.55e+03
			BBPSO-DE	**2.44e+01**↓	**4.86e+01**↓	2.26e+01	1.52e+03
f_6	1000	2000	LBest PSO	4.34e+03	4.75e+03	2.12e+02	2.24e+03
			Gbest PSO	9.31e+03	1.06e+04	7.33e+02	1.10e+03
			BBPSO-MC-lbest	5.54e+03	5.90e+03	1.72e+02	3.46e+03
			MCPSO	1.91e+02	2.30e+02	1.79e+01	1.79e+03
			BBPSO-DE	**9.38e+01**↓	**1.38e+02**↓	6.53e+01	3.41e+03

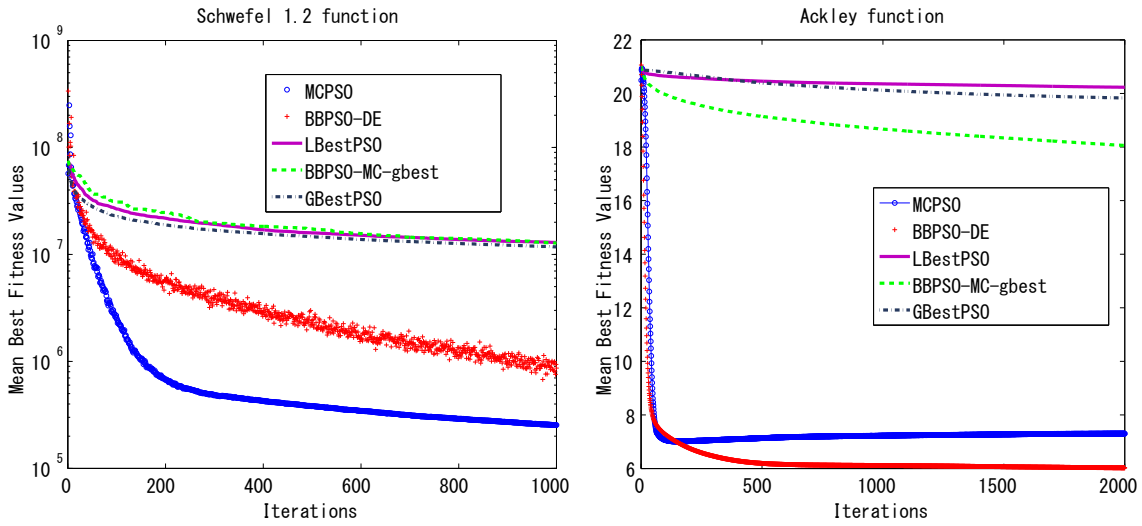


Fig. 6.5 Convergence graphs of f_3 & f_4 on 1000 dimensions

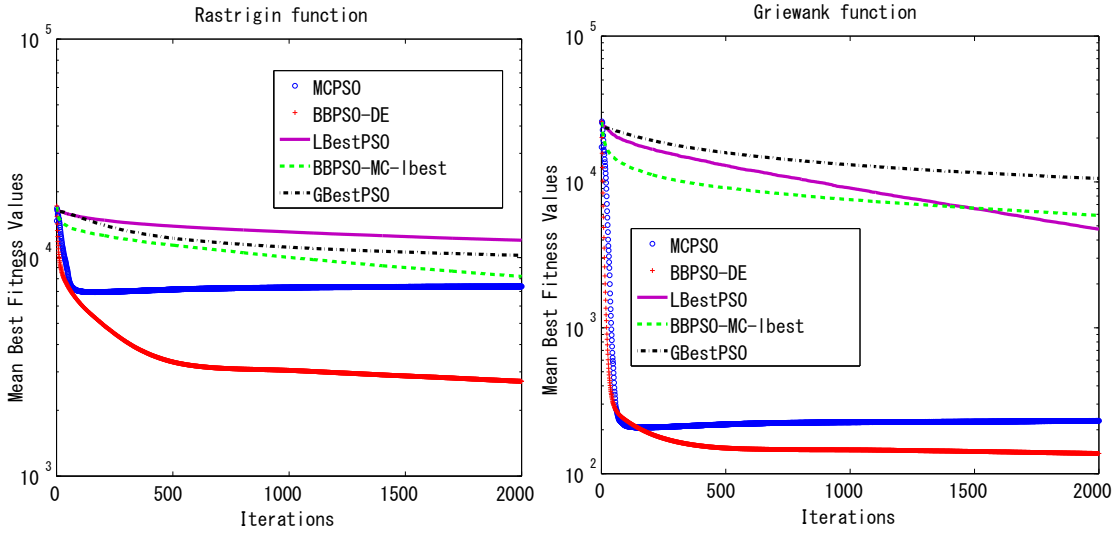


Fig. 6.6 Convergence graphs of f_5 & f_6 on 1000 dimensions

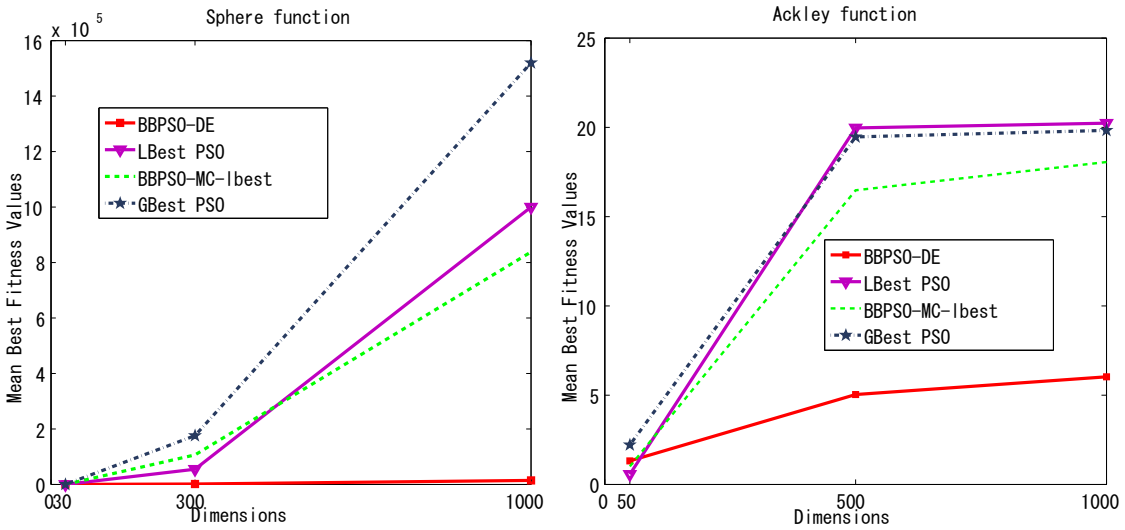


Fig. 6.7 Comparison of mean best optimum errors on various dimensions

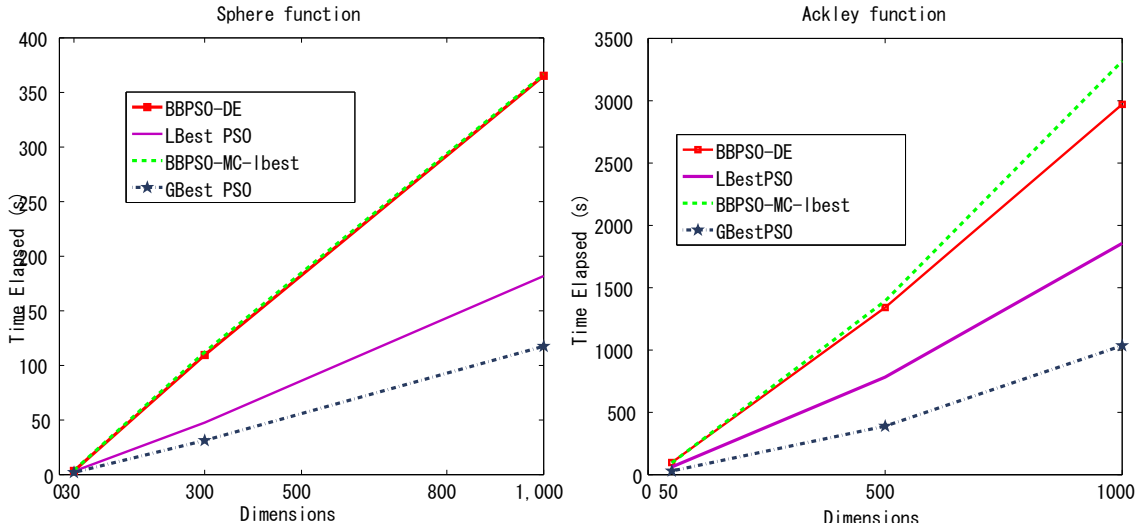


Fig. 6.8 Comparison of computation time on various dimensions

mean μ, standard deviation σ and ring topology. The new standard deviation $\sigma(|lbest_i - \overline{X})$ maintains the swarm diversity by increasing the particle' variance. The ring topology of $memory - swarm$ is more likely to strengthen the global exploration capability of BBPSO-DE and successfully maintain the local exploration ability. Figure 6.4, Fig.6.5 and Fig.6.6 show that both MCPSO and BBPSO-DE have the significant scalability on six test functions when the dimensionality of benchmark function up to 1000. In addition, BBPSO-DE has better performance than MCPSO on most of test functions except Schwefel 1.2 function.

In order to get an overall picture of the scalability of BBPSO-DE, the mean optimum errors obtained by four algorithms on two selected benchmarks on various dimensions are plotted in Fig.6.7. From figure 6.7, it can be noticed that BBPSO-DE shown very good scalability to the search dimension, that is, the performance does not deteriorate seriously as the dimension increases. However, the computation time of BBPSO-DE and BBPSO-MC-lbest shown in Fig.6.8 are greatly increased with high dimensionality. One of reasons is that using a normal distribution for scaling the amplitude of particles' trajectories is high time consumption to BBPSO.

To summarize, experiment results show that the performance of BBPSO-DE doesn't deteriorate seriously as the dimension increases. According to the experimental analysis, DE strategy is able to eliminate the stagnation of swarm and maintain the diversity of swarm as the dimensionality of problem increases. Furthermore, the outstanding performance of BBPSO-DE should be attributed to the redefined equations of μ and σ and the local memory-swarm topology. In conclusion, BBPSO-DE is a very competitive algorithm for solving both unimodal and multimodal large scale problems without any control parameters.

6.4 Summary

In this chapter, a new BBPSO-DE algorithm is proposed to resolve large scale problems. This algorithm is based on the modified BBPSO algorithm and employed Differential Evo-

lution strategy. In BBPSO-DE, several strategies are integrated to alleviate premature convergence. These manipulations are mainly including using ring neighbourhood topology, introducing particle's local best memory ($lbest_i$) and centroid position of all particles ($\overline{X}$) to scale the standard deviation (σ). When a particle's local best memory is equal to its personal best position, mutation and crossover operators of DE strategy is adopted. The motivation of using the ring neighbourhood topology is that it is more likely to strengthen the swarm's global exploration capability while successfully maintains the local exploitation functionality. In BBPSO-DE, we try to redefine the equations of mean (μ) and standard deviation (σ) with $lbest_i$ and $\overline{X}$ instead of $\boldsymbol{p}_i$ and $\boldsymbol{p}_g$. Empirical results demonstrate that the redefined equations are contributions to the outstanding performance of BBPSO-DE for large scale problems. Furthermore, the mutation and crossover operators of DE are employed for updating the local best memory ($lbest_i$), which increases the swarm diversity and enhances the search performance of BBPSO-DE as the dimensionality of problem increases. In addition, one key advantage of the proposed BBPSO-DE is that there is no need to specify any control parameters that are often required in traditional stochastic algorithms.

However, the study of BBPSO-DE has some limitations to be improved in our future research. Firstly, a Gaussian distribution used for BBPSO-DE is time-consumption and the computation time of BBPSO-DE is greatly increased with the increasing dimensionality. Secondly, the comparison against other state-of-the-art algorithms for LSGO should be further explored. In the end, the statistical results of mean optimization errors and standard deviation can be misleading when reporting the results for experiment, the Mann Whitney U test and a modified Bonferroni procedure will be used for our further evaluation analysis.

Chapter 7

Dynamic Heterogeneous Particle Swarm Optimization

The standard PSO and most of its modifications make use of homogeneous swarms where all of the particles follow exactly the same behavior. Models that consider populations of homogeneous individuals are attractive because of their conceptual simplicity. However, heterogeneity is ubiquitous in nature [31]. Modelled with populations of heterogeneous individuals, the optimization algorithm therefore has the ability to maintain an appropriate balance between exploration and exploitation throughout the search process. Recently, the Static Heterogeneous Particle Swarm Optimization (SHPSO) has been studied by more and more researchers. In SHPSO, the different search behaviours assigned to particles do not change during the search process. As a consequence of this, the inappropriate population size of exploratory particles could leave the SHPSO algorithm with great difficulties of escaping local optima. This motivated our attempt to improve the performance of SHPSO by introducing the dynamic heterogeneity. The self-adaptive heterogeneity is able to alter its heterogeneous structure according to some events caused by the behaviour of the swarm. According to the different types of heterogeneity, we propose two kinds of Dynamic Heterogeneous Particle Swarm Optimization (DHPSO) models, namely, DHPSO-d and DHPSO-p.

7.1 Overview and Preliminaries

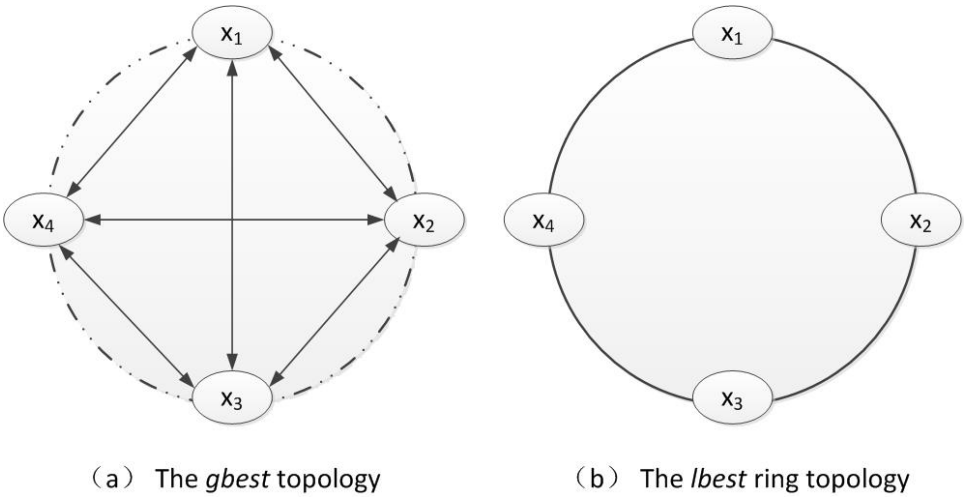
7.1.1 Concept of the Heterogeneous PSO

Particle Swarm Optimization (PSO) [21] is a stochastic, population-based optimization method, which has been successfully applied to a number of applications. In the vast majority of PSO models, it is assumed that the swarm is composed of homogeneous individuals. Models that consider populations of homogeneous individuals are attractive because of their conceptual simplicity. However, heterogeneity is ubiquitous in nature. Recently, heterogeneous systems have drawn the attention of researchers working in different areas of swarm intelligence because designing heterogeneous models more accurately and approximately resembles real circumstances.

In general, a particle swarm is heterogeneous if it has at least a pair of particles that differ in any of the four aspects: the neighbourhood size, the model of influence, the update rules and their parametrization [18]. In this chapter, we only empirically study two types of heterogeneity, update-rule heterogeneity and model-of-influence heterogeneity. If different particles use different rules for updating their position in the search space, then the swarm exhibits update-rule heterogeneity. When particles in a swarm use different mechanisms for choosing their informers, we say that the swarm exhibits model-of-influence heterogeneity. It was shown in [26], [61] that the Heterogeneous PSO (HPSO) model produced significantly better solutions than a selection of homogeneous PSO models.

7.1.2 The Static Heterogeneous PSO Model: HGLPSO

If the heterogeneity type assigned to particles during initialization do not change during the search process, we qualify the resulting heterogeneity as static, otherwise it is said to be dynamic. An example of a static heterogeneity structure is the HGLPSO algorithm, in which



（a） The *gbest* topology　　（b） The *lbest* ring topology

Fig. 7.1 Two typical topologies of SPSO

some of the particles are informed by the *lbest* model, while the others are fully-informed (*gbest* model).

Two kinds of topologies with a population of four particles are shown in Fig.7.1. Figure 7.1(a) shows the full connections with four particles in *gbest* topology. In Fig.7.1(b), the *lbest* ring topology connects each particle to only two other particles in the swarm. More recent research has revealed that *lbest* topology return improved results across many standard problem sets when used in conjunction with other improvements to the algorithm [49]. Many investigations within the particle swarm paradigm [43], [47] have found that the *gbest* type converges quickly on problem solutions but has a weakness for becoming trapped in the local optima, while *lbest* populations are able to escape from local optima, as subpopulations explore different regions. In [44], [48] Kennedy theorized that heterogeneous population structures, with some subsets of the population tightly connected and others relatively isolated, could provide the benefits of both *lbest* and *gbest* sociometries. Following this line of thought, an example of a swarm with heterogeneity, HGLPSO algorithm is proposed in this chapter as a comparative algorithm. HGLPSO is a type of static heterogeneous structure combined with the GBest PSO model and the LBest PSO model, in which some of the particles are informed by the best particle in the sub-swarm, while the others are informed

by the best particle of their local neighbours. Using a heterogeneous swarm thus reduces the risk of using a homogeneous swarm of the wrong type for the problem at hand. However, it hardly outperforms the best performing homogeneous swarm.

7.1.3 Solutions

In population-based algorithms, finding the optimal solution of a problem is based on the right balance between exploration and exploitation of the search space. The premise of the SHPSO model is that a better balance of exploration and exploitation can be achieved by having particles that follow different search behaviours, which should result in more accurate solutions. However, empirical study shows that the nature of different search behaviours between particles is not the only factor that determines the performance of SHPSO [18]. The relative composition of the swarm plays a major role in this respect. In SHPSO, the proportions of particles of different kinds are assigned during initialization, moreover, these assignments do not change during the optimization process. As a consequence of this, the inappropriate composition of the swarm will lead the particles near the local minimum to be trapped in the local optima. In this case, escaping from the local optima becomes difficult and SHPSO suffers from the premature convergence problem.

This motivated our attempt to improve the performance of SHPSO by introducing adaptive particle swarms as a response to some events caused by the behaviour of the swarm, thus "guiding" an appropriate balance between exploration and exploitation in the search space. The dynamic self-adaptive heterogeneity is able to automatically alter its heterogeneity type by the triggering configuration. The proposed triggering configuration keeps track of the frequency of the unchanged best position of the entire swarm (p_g) for a number of iterations. This information is then used to select a new heterogeneous structure when p_g is considered stagnant. According to different types of heterogeneity, we propose two different DHPSO models, namely, DHPSO-d and DHPSO-p [99]. In DHPSO-d, three different update rules

are defined for different particles by the trigger event. Particles dynamically use different rules to update their positions when the global best position is considered stagnant and the triggering event is confirmed. In DHPSO-p, a global *gbest* model and a *pairwise connection* model are proposed to provide the particles different mechanism choosing their informers when the swarm being trapped in the local optimal solution. In order to investigate the scalability of DHPSO-d and DHPSO-p, a series of experiments with four state-of-the-art algorithms are performed on ten well-known optimization problems. The scalability analysis of DHPSO-d and DHPSO-p reveals that the dynamic self-adaptive heterogeneous structure is able to address the exploration-exploitation trade-off problem in PSO, and provide the promising optimal solution of a problem simultaneously.

7.2 Proposal of Dynamic HPSO (DHPSO)

7.2.1 Proposal of DHPSO-d

Conceptually, we say that DHPSO-d exhibits dynamic update-rule heterogeneity. In DHPSO-d ($-r$ represents rule), different particles dynamically use different rules for updating their positions by the triggering configuration in the search space. There are three rules (rule A, rule B and rule C) used in DHPSO-d. To rule A, the velocity and position of each particle are updated by the following equations [7]:

$$\boldsymbol{v}_i = \chi * (\boldsymbol{v}_i + c_1\epsilon_1 * (\boldsymbol{p}_i - \boldsymbol{x}_i) + c_2\epsilon_2 * (\boldsymbol{p}_g - \boldsymbol{x}_i)) \tag{7.1}$$

$$\boldsymbol{x}_i^* = \boldsymbol{x}_i + \boldsymbol{v}_i \tag{7.2}$$

In Eq.(7.1), χ is a constriction factor, ϵ_1 and ϵ_2 are independent random numbers uniquely generated at every update for each individual dimension in the range $[0, 1]$ according to references [7], [78]. In order to ensure convergence, the values $\chi \approx 0.72984$ and $c_1 = c_2 =$

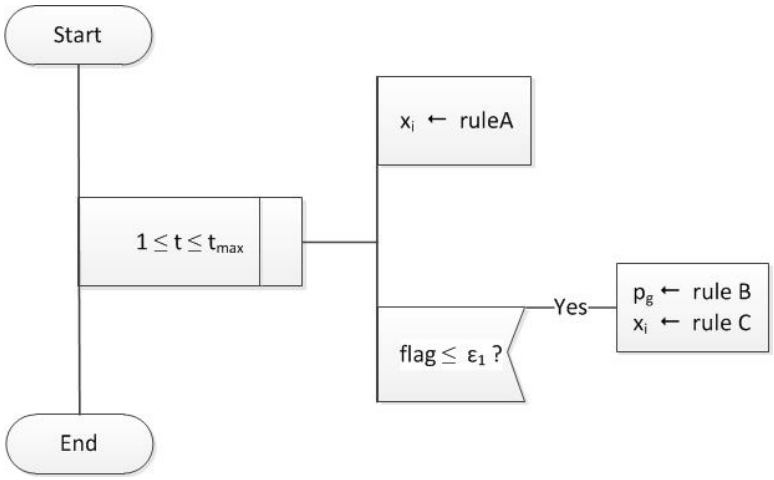


Fig. 7.2 Dynamic process of heterogeneity configuration in DHPSO-d

2.05 are preferred in most cases. p_i is the best position of the particle i, $\boldsymbol{x}_i$ is the current position of the particle i and $\boldsymbol{p}_g$ is the best position of the entire swarm. To rule B, $\boldsymbol{p}_g$ can be reset by:

$$\boldsymbol{p}_g = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{p}_i \tag{7.3}$$

To rule C, all particles in swarm update their positions by the following equations:

$$\boldsymbol{v}_i = \psi_1 * \boldsymbol{v}_i + \psi_2 * (\boldsymbol{p}_i - \boldsymbol{x}_i) \tag{7.4}$$

$$\boldsymbol{x}_i^* = \boldsymbol{x}_i + \boldsymbol{v}_i \tag{7.5}$$

In Eq.(7.4) ψ_1 and ψ_2 are random numbers uniquely generated at every update for each individual dimension in the range [0, 1].

Figure 7.2 demonstrates the dynamic process of heterogeneity configuration in DHPSO-d during the optimization process. Where t is the iteration number of the swarm E. During each iteration, swam E is a homogeneous swarm that all particles use rule A for updating their positions in the search space. When the $\boldsymbol{p}_g$ does not improve for a specified number of iterations (ε_1), triggering event a is confirmed ($flag \ge \varepsilon_1$), then $\boldsymbol{p}_g$ is reset by rule B and all particles in swarm E update their positions by rule C. After that, all particles will

be following up with rule A to the next iteration. To swarm E, the $\boldsymbol{p}_g$ is shared among all particles and three rules are used for the particles by the triggering configuration, the specific update-rule heterogeneity is defined as model Υ in this chapter. The pseudocode of model Υ used in DHPSO-d can be summarized in Algorithm 14.

Algorithm 14 Main function of model Υ used in DHPSO-d

1: Initialization: $c_1 = c_2 = 2.05$, $\chi = 0.7298$, $\epsilon_1 = \lambda_1$, population n
2: Initialize each particle's position $\boldsymbol{x}_i$: $\boldsymbol{x}_i = rand() \in (x_{min}, x_{max})$; $i \in [1..n]$
3: Calculate fitness value of each particle $f(\boldsymbol{x}_i)$
4: Initialize personal best position $\boldsymbol{p}_i$: $\boldsymbol{p}_i = \boldsymbol{x}_i$, $f(\boldsymbol{p}_i) = f(\boldsymbol{x}_i)$
5: Find the swarm's best position $\boldsymbol{p}_g$
6: Initialization: $\boldsymbol{v}_i = 0.5 * rand(); rand() \in (x_{min}, x_{max})$, $flag = 0$, $trap = 0$
7: Repeat
8: **for all** particles **do**
9: Generate: $\psi_1 = rand() \in [0, 1]$, $\psi_2 = rand() \in [0, 1]$
10: Update particle $\boldsymbol{x}_i$ with rule $\boldsymbol{A}$
11: **Calculate fitness value of each updated particle** $f(\boldsymbol{x}_i^*)$
12: **if** $f(\boldsymbol{x}_i^*) < f(\boldsymbol{p}_i)$ **then**
13: $\boldsymbol{p}_i = \boldsymbol{x}_i^*$, $f(\boldsymbol{p}_i) = f(\boldsymbol{x}_i^*)$
14: **end if**
15: **end for**
16: Find the new swarm's best position $\boldsymbol{p}_g^*$
17: **if** $f(\boldsymbol{p}_g^*) < f(\boldsymbol{p}_g)$ **then**
18: $\boldsymbol{p}_g = \boldsymbol{p}_g^*$, $flag = 0$
19: **else**
20: Count the number of unchanged $\boldsymbol{p}_g$: $flag = flag + 1$
21: **end if**
22: **if** $flag \geq \epsilon_1$ **then**
23: Reset $\boldsymbol{p}_g$ with rule $\boldsymbol{B}$
24: Re-initialize each $\boldsymbol{p}_i$: $\boldsymbol{p}_i = rand() \in (x_{min}, x_{max})$
25: Update each particle $\boldsymbol{x}_i$ with rule $\boldsymbol{C}$, $flag = 0$
26: Count the number of $\boldsymbol{p}_g$ trapped in the local optima: trap=trap+1
27: **end if**
28: Until maximum iterations are attained

From Algorithm 14, we can see that DHPSO-d maintains the algorithmic simplicity of GBestPSO. For each iteration, the number of Fitness Evaluations (FEs) is $n * D$ shown in line 10. Where n is the population size and D is the dimension of the problem. Apart from the FEs, which is problem dependent [39], [57], the main computational cost in DHPSO-d

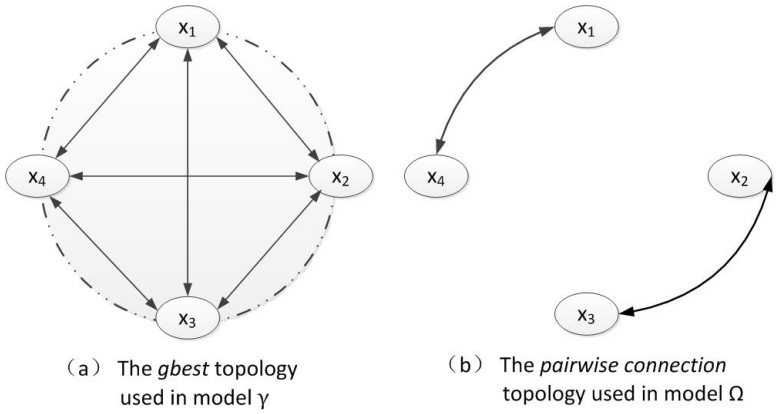


Fig. 7.3 Two kinds of topologies in DHPSO-p

is to find the global best optimum $\boldsymbol{p}_g$ from all of the $\boldsymbol{p}_i$, which is an inevitable operation in most swarm or population based evolutionary algorithms [40], [85]. Consequently, at each iteration, the computational complexity of DHPSO-d is $O(nD)$. In addition, we note that ε_1 is problem dependent and the method of ε_1 selection ($\varepsilon_1 = \lambda_1$) is a trial.

7.2.2 Proposal of DHPSO-p

Conceptually, we say that DHPSO-p is a model-of-influence heterogeneity. In this chapter, DHPSO-p is a dynamic heterogeneous structure combined with the *gbest* topology and the *pairwise connection* topology ($-m$ means model). Figure 7.3(a) shows full connections with four particles in the *gbest* topology, which is the topology type used in model Υ. To model Υ, all particles share the global best experience $\boldsymbol{p}_g$ that can lead the particles to cluster around the global best. If $\boldsymbol{p}_g$ is located near a local optima, escaping from local solution becomes difficult. In this situation, the dynamic update-rule strategy will be performed in model Υ. If the state of $\boldsymbol{p}_g$ trapped in the local optimum couldn't be eliminated only by model Υ, then a new model Ω will be enabled. In Fig.7.3(b), each particle only has one specified neighbourhood, which is the topology type used in model Ω. This pairwise connection mechanism are borrowed from Competitive Swarm Optimization (CSO) [12], a novel swarm intelligence algorithm for large scale optimization. In model Ω, neither the personal best position of each

particle nor the global best position is involved in updating the particles. Instead, all particles are divided into two teams: an elite team and a loser team. The loser team will update their positions by learning from the elite team, while the elite team is directly passed to the swarm of the next iteration. The motivation for us to introduce pairwise competition mechanism in model Ω is to increase swarm diversity and avoid the particles trapped in the local optimum region, which potentially enhances the global exploration capability of DHPSO-p.

Algorithm 15 Main function of DHPSO-p

1: Initialize parameters: $c_1 = c_2 = 2.05$, $\chi = 0.7298$
2: Initialize parameters: $\epsilon_1 = \lambda_1$, $\epsilon_2 = \lambda_2$, $flag = 0, trap = 0$
3: Initialize population n and particles x_i: $\boldsymbol{x}_i = rand() \in (x_{min}, x_{max})$; $i \in [1..n]$
4: Repeat
5: **for all** particles **do**
6: Use model ϒ
7: **end for**
8: **if** $trap \geq \varepsilon_2$ **then**
9: Switch to Model Ω
10: **Break**
11: **end if**
12: Until maximum iterations are attained

Without loss of generality, the population of DHPSO-p is denoted as an even number n. In DHPSO-p, the process of heterogeneity configuration are including of the state of model ϒ and a new state of model Ω triggered by the triggering configuration. To model ϒ, when $\boldsymbol{p}_g$ is considered stagnant and the number of stagnation reaches a specified threshold ε_1, it means that $\boldsymbol{p}_g$ is trapped in the local optima, the dynamic update-rule strategy will be performed in model ϒ. However, if the number of traps is more than a threshold ε_2, the trapped $\boldsymbol{p}_g$ can be addressed by assigning a new model Ω. The main function of DHPSO-p is summarized in Algorithm 15. It should be noted that once the DHPSO-p switches to model Ω, it never be back to model ϒ.

Algorithm 16 demonstrates the fundamental functions of model Ω, which can be seemed as a variant of CSO. Different with the simple serial pairwise competition mechanism in CSO, a paralleled competitive approach is introduced to Model Ω with the aim of decreasing

Algorithm 16 Sub-function of DHPSO-p in model Ω

Initialize each particle's position $\boldsymbol{x}_i$: $\boldsymbol{x}_i = rand() \in (x_{min}, x_{max})$; $i \in [1..n]$
Calculate the fitness value of each particle $f(\boldsymbol{x}_i)$
Initialize personal best position $\boldsymbol{p}_i$: $\boldsymbol{p}_i = \boldsymbol{x}_i$, $f(\boldsymbol{p}_i) = f(\boldsymbol{x}_i)$
Find the swarm's best position $\boldsymbol{p}_g$
Repeat
Create sequence $\{S(i)\}$; $i \in [1..n]$ by the ascending order of $f(\boldsymbol{x}_i)$
Put $S(1)$ into the elite team Te
Put $S(n)$ into the loser team Tl
Randomly select $(n/2 - 1)$ positions from $\{S(i)\}$; $i \in [2..n-2]$
Their next neighbouring positions $S(i+1)$ are put into the loser team
The rest of particles in sequence are assigned to the elite team
The elite team will be unchanged and passed to the next iteration
for all particles **do**
  Generate: $\psi_1 = rand() \in [0, 1]$, $\psi_2 = rand() \in [0, 1]$
  Update the loser with rule **D**
  Calculate fitness value of each updated particle $f(\boldsymbol{x}_i^*)$
  if $Fitness(\boldsymbol{x}_i^*) < Fitness(\boldsymbol{p}_i)$ **then**
    $\boldsymbol{p}_i = \boldsymbol{x}_i^*$, $f(\boldsymbol{p}_i) = f(\boldsymbol{x}_i^*)$
  end if
end for
Update the global best position: $\boldsymbol{p}_g = argmin(f(\boldsymbol{p}_i))$
Until maximum iterations are attained

the operation complexity and maintaining the swarm diversity simultaneously. First of all, all particles are stored in a sorted sequence $S = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_n\}$, $(\{S(i)\};\ i \in [1..n])$ according the ascending order of $Fitness(\boldsymbol{x}_i)$. Secondly, the first particle in $\{S(i)\}$, namely $S(1)$, is definitely denoted as an elite and assigned to the elite team Te, $(\{Te(k)\},\ k \in [1..n/2])$, at the same time, $S(n)$ is put into the loser team Tl, $(\{Tl(l)\},\ l \in [1, n/2])$. Then, randomly select $(n/2 - 1)$ positions from $\{S(i)\}$ in a specified range $i \in [2..n-2]$. Their next neighbouring positions $S(i+1)$ are assigned to the loser team. The rest of particles in $\{S(i)\}$ will be put into the elite team. All particles in the elite team will be passed to the next iteration, while the loser, will update its position by learning from the randomly selected elite with rule D. At each iteration, only half of the particles will be updated, after that, the historical best position of each particle and the global best position of the swarm are recorded. However, it should be pointed out that neither $\boldsymbol{p}_i$ nor $\boldsymbol{p}_g$ is involved in the search process of Model Ω.

From Algorithm 16, we find that model Ω maintains the algorithmic simplicity of model ϒ. The main FEs is to calculate the updated loser particle $\boldsymbol{x}_i^*$, which are shown in line 15 marked with underline. For each iteration, the number of FEs is $0.5 * n * D$, where n is the swarm size and D is the dimensionality of the problem. Apart from the FEs, the main computational cost in model Ω is to create a sequence S of all particles by the ascending order of $Fitness(\boldsymbol{x}_i)$ and update the positions of loser particles by learning from the elite team. Then, at each iteration, the computational complexity of model Ω is $O(nD)$. Combination of model ϒ and model Ω shown in Algorithm 15, at each iteration, the computational complexity of DHPSO-p is $O(nD)$.

To rule D, all loser particles update their positions by the following equations:

$$\begin{cases} \boldsymbol{v}_{loser} = \psi_1 * \boldsymbol{v}_{loser} + \psi_2 * (\boldsymbol{x}_{elite} - \boldsymbol{x}_{loser}) \\ \boldsymbol{x}^*_{loser} = \boldsymbol{x}_{loser} + \boldsymbol{v}_{loser} \end{cases} \tag{7.6}$$

Table 7.1 Evaluation test environment

OS	Windows 7 Professional SP1
Processor	Intel(R) $Core^{TM}$i7 860@2.80GHz
Memory (RAM)	8.00GB
System type	64-bit operation system
Tool	MATLAB R2013b

Where, $\boldsymbol{x}_{elite}$ is the position of elite particle, $\boldsymbol{x}_{loser}$ is the position of loser particle and $\boldsymbol{x}^*_{loser}$ is the new updated position of loser. ψ_1 and ψ_2 are random numbers uniquely generated at every update for each individual dimension in the range [0, 1].

7.3 Experiments

7.3.1 Evaluation Method

The experiments are implemented on a PC with Windows operating system, and all algorithms are written in Matlab language. The detail information of the test environment is demonstrated in Table 7.1. The bounds and global minimums of ten benchmark functions are shown in Table 7.2. These benchmarks were chosen for their variety. Functions f_1--f_6 are highly complex multimodal problems with many local minima, f_7--f_{10} are continuous, convex and unimodal test problems.

The four state-of-the-art algorithms are also chosen for their variety. First of all, the proposed DHPSO-d and DHPSO-p are based on GBestPSO [7]. So the GBestPSO algorithm is used to compare the performance of our proposed algorithms. Secondly, LBestPSO [7] and Firefly Algorithm (FA) [101] are well-known algorithms for solving the highly complex multimmodal problems. The global searching capability of DHPSO-d and DHPSO-p for multimmodal problems can be confirmed by empirical comparison with LBestPSO and FA. The last comparative algorithm, HGLPSO, is a type of SHPSO, where particles will be

Table 7.2 The bounds and global minimums of benchmark functions.

Equation	Bounds	Optimum
Rosenbrock $f_1(x) = \sum_{j=1}^{D}(100 * (x_{j+1} - x_j^2)^2 + (x_j - 1)^2)$	$(-100, 100)^D$	$0, (x^* = 1.0^D)$
Rastrigin $f_2(x) = \sum_{j=1}^{D}(x_j^2 - 10 * \cos(2\pi x_j)) + 10 * D$	$(-5.12, 5.12)^D$	$0, (x^* = 0.0^D)$
Griewank $f_3(x) = \frac{1}{4000}\sum_{j=1}^{D} x_j^2 - \prod_{j=1}^{D}\cos(\frac{x_j}{\sqrt{j}}) + 1$	$(-5, 10)^D$	$0, (x^* = 0.0^D)$
Ackley $f_4(x) = -a * \exp(-0.02 * (D^{-1}\sum_{j=1}^{D-1} x_j^2)\frac{1}{2})$	$(-32, 32)^D$	0
$-\exp(D^{-1}\sum_{j=1}^{D}\cos(2\pi x_j)) + a + \exp, a = 20$		$(x^* = 0.0^D)$
Levy $f_5(x) = sin^2(\pi\varpi_1) + (\varpi_D - 1)^2[1 + sin^2(2\pi\varpi_D)]$	$(-10, 10)^D$	0
$+\sum_{j=1}^{D-1}(\varpi_j - 1)^2[1 + 10sin^2(\pi\varpi_j + 1)], \varpi_j = 1 + \frac{x_j - 1}{4}$		$(x^* = 1.0^D)$
Schwefel $f_6(x) = 418.9829 * D - \sum_{j=1}^{D} x_j sin(\sqrt{\lvert x_j \rvert})$	$(-500, 500)^D$	0
		$(x^* = 420.9687^D)$
Sphere $f_7(x) = \sum_{j=1}^{D} x_j^2$	$(-5.12, 5.12)^D$	$0, (x^* = 0.0^D)$
Rotated hyper-ellipsoid $f_8(x) = \sum_{j=1}^{D}\sum_{k=1}^{j} x_k^2$	$(-65.536, 65.536)^D$	$0, (x^* = 0.0^D)$
Sum of different powers $f_9(x) = \sum_{j=1}^{D} \lvert x_j \rvert^{j+1}$	$(-1, 1)^D$	$0, (x^* = 0.0^D)$
Zakharov $f_{10}(x) = \sum_{j=1}^{D} x_j^2 + (\sum_{j=1}^{D} 0.5jx_j)^2 + (\sum_{j=1}^{D} 0.5jx_j)^4$	$(-1, 1)^D$	$0, (x^* = 0.0^D)$

Table 7.3 The control parameters setting of algorithms.

Algorithm	Control Parameters
GBest PSO	$c1 = c2 = 2.05$, $\chi = 0.7298$
LBest PSO	$c1 = c2 = 2.05$, $\chi = 0.7298$
FA	$\alpha = 0.25$, $\gamma = 1$, $\beta_{min} = 0.20$, $\beta_0 = 1$
HGLPSO	$c1 = c2 = 2.05$, $\chi = 0.7298$
DHPSO-d	$c1 = c2 = 2.05$, $\chi = 0.7298$, $\lambda_1 = 80$
DHPSO-p	$c1 = c2 = 2.05$, $\chi = 0.7298$, $\lambda_1 = 80$, $\lambda_2 = 2$

allocated different search behaviours by selecting different topologies from GBestPSO and LBestPSO. The control parameters of all algorithms are shown in Table 7.3.

7.3.2 Evaluation Results

The collecting experimental results are including the best optimum solution for the test functions, averaged optimum solution and running time of each algorithm after 30 times independent trials. These experimental results are displayed in Table 7.4 and Table 7.5. We analyse these empirical data with six steps as follows:

* To multimodal functions: f_1--f_6

Step 1 Pairwise comparisons between DHPSO-d and four compared algorithms by *t*-test.

Step 2 Pairwise comparisons between DHPSO-p and five compared algorithms including DHPSO-d by *t*-test.

* To unimodal functions: f_7--f_{10}

Step 3 Pairwise comparisons between DHPSO-d and four compared algorithms by *t*-test.

Step 4 Pairwise comparisons between DHPSO-p and five compared algorithms by *t*-test.

* To all functions: f_1--f_{10}

Step 5 A modified Bonferroni procedure for DHPSO-d versus HGLPSO.

Step 6 A modified Bonferroni procedure for DHPSO-p versus HGLPSO.

Table 7.4 The statistical results of optimization errors on multimodal functions with a population of 100 and 200 after 30 trials of 1×10^4 function evaluations

Algorithm	50-D	f_1	f_2	f_3	f_4	f_5	f_6
GBestPSO	Best	2.07e+01	1.30e+02	9.86e-03	0	4.90e+00	4.35e+03
	Mean	1.16e+05	2.25e+02	7.44e-01	4.77e+00	2.65e+01	6.48e+03
	Time(s)	1.59e+02	1.53e+02	3.71e+02	1.54e+02	2.33e+02	1.86e+02
LBestPSO	Best	1.41e+02	1.05e+02	0	0	0	5.74e+03
	Mean	1.81e+02	1.90e+02	0	0	4.60e-01	7.00e+03
	Time(s)	3.82e+02	3.97e+02	4.26e+02	3.64e+02	4.83e+02	4.77e+02
FA	Best	4.70e+01	4.97e+00	0	1.61e-06	0	7.82e+03
	Mean	8.96e+02	3.90e+01	5.21e-07	4.70e-03	7.56e-06	9.07e+03
	Time(s)	9.34e+03	2.88e+05	9.73e+03	9.40e+03	6.02e+04	9.41e+03
HGLPSO	Best	3.07e-04	1.61e+02	0	0	0	5.12e+03
	Mean	2.76e+01	2.13e+02	2.47e-04	3.66e-01	5.73e+00	6.69e+03
	Time(s)	1.99e+02	2.62e+02	2.99e+02	8.50e+01	3.41e+02	3.13e+02
DHPSO-d	Best	2.31e+01	5.37e+01	0	0	0	2.61e+03
	Mean	8.86e+04	1.35e+02	1.43e-01	8.51e-01	0	3.92e+03
	Time(s)	2.05e+02	1.78e+02	2.07e+02	1.90e+02	2.74e+02	1.96e+02
DHPSO-p	Best	1.05e+02	8.95e+00	0	7.38e-07	0	2.57e+03
	Mean	2.01e+03	1.77e+01	4.35e-06	2.86e-01	2.44e-01	3.93e+03
	Time(s)	2.50e+02	3.04e+02	3.41e+02	3.04e+02	3.71e+02	3.25e+02
Algorithm	100-D	f_1	f_2	f_3	f_4	f_5	f_6
GBestPSO	Best	4.56e+01	4.30e+02	5.79e-01	3.24e+00	7.26e+01	1.15e+04
	Mean	2.00e+09	5.82e+02	1.00e+00	1.38e+01	1.20e+02	1.50e+04
	Time(s)	4.08e+02	4.42e+02	5.07e+02	4.71e+02	5.77e+02	4.96e+02
LBestPSO	Best	4.03e+02	4.55e+02	0	1.21e+02	3.18e+01	1.41e+04
	Mean	5.63e+02	5.21e+02	0	2.41e+02	5.85e+01	1.58e+04
	Time(s)	8.41e+02	9.32e+02	8.98e+02	8.79e+02	1.20e+03	1.14e+03
FA	Best	1.01e+02	4.88e+01	1.69e-07	3.20e-03	4.98e-05	2.01e+04
	Mean	8.96e+02	9.05e+01	4.36e-06	1.16e-02	9.30e-05	2.16e+04
	Time(s)	4.48e+04	4.23e+05	5.22e+05	6.29e+04	3.67e+05	1.39e+05
HGLPSO	Best	3.85e+01	4.46e+02	0	1.27e+00	4.32e+01	1.29e+04
	Mean	1.49e+02	5.39e+02	0	2.63e+00	6.49e+01	1.53e+04
	Time(s)	5.90e+02	8.22e+02	8.60e+02	8.12e+02	9.28e+02	9.57e+02
DHPSO-d	Best	7.34e+01	3.89e+02	5.09e-01	0	2.14e+01	7.74e+03
	Mean	1.66e+09	4.68e+02	8.96e-01	1.20e+01	6.04e+01	1.10e+04
	Time(s)	4.82e+02	4.92e+02	5.59e+02	5.29e+02	6.46e+02	5.09e+02
DHPSO-p	Best	1.71e+02	1.39e+01	0	0	0	5.27e+03
	Mean	4.10e+02	2.20e+01	0	8.79e-03	1.59e-01	7.26e+03
	Time(s)	5.71e+02	6.75e+02	7.25e+02	6.23e+02	8.98e+02	7.42e+02

Table 7.5 The statistical results of optimization errors on unimodal functions with a population of 100 and 200 after 30 trials of 1×10^4 function evaluations.

Algorithm	50-D	f_7	f_8	f_9	f_{10}
GBestPSO	Best	0	0	0	2.55e+01
	Mean	5.24e+00	6.10e+04	0	2.40e+02
	Time(s)	1.25e+02	3.46e+02	3.32e+02	1.61e+02
LBestPSO	Best	0	0	0	4.03e-05
	Mean	0	0	0	8.17e-05
	Mean	1.80e+02	7.98e+02	9.10e+02	3.22e+02
FA	Best	6.57e-07	4.79e-02	0	0
	Mean	4.21e-06	1.15e-01	0	1.15e-06
	Time(s)	9.50e+03	1.66e+04	2.73e+03	1.00e+04
HGLPSO	Best	0	0	0	1.27e+01
	Mean	0	0	0	2.15e+02
	Time(s)	1.50e+02	5.82e+02	6.43e+02	2.13e+02
DHPSO-d	Best	0	0	0	0
	Mean	0	1.05e+04	0	1.39e+02
	Time(s)	1.45e+02	4.01e+02	4.03e+02	2.23e+02
DHPSO-p	Best	0	0	0	0
	Mean	1.17e-04	2.22e-01	0	2.15e-04
	Time(s)	1.99e+02	4.93e+02	3.54e+02	2.00e+02
Algorithm	100-D	f_7	f_8	f_9	f_{10}
GBestPSO	Best	0	8.59e+04	0	4.81e+02
	Mean	4.54e+01	4.77e+05	0	1.04e+03
	Time(s)	3.13e+02	1.49e+03	8.81e+02	4.00e+02
LBestPSO	Best	0	0	0	1.29e+00
	Mean	0	0	0	2.23e+00
	Time(s)	4.72e+02	3.85e+03	2.23e+03	6.31e+02
FA	Best	3.23e-05	4.41e-01	0	4.33e-05
	Mean	5.29e-05	9.77e+00	0	6.42e-04
	Time(s)	3.70e+05	6.26e+04	3.37e+03	6.78e+04
HGLPSO	Best	0	0	0	2.76e+01
	Mean	0	0	0	3.45e+02
	Time(s)	4.22e+02	1.18e+03	2.67e+03	5.39e+02
DHPSO-d	Best	0	1.72e+04	0	2.61e+02
	Mean	8.74e+00	2.03e+05	0	9.85e+02
	Time(s)	3.82e+02	1.87e+03	1.08e+03	5.55e+02
DHPSO-p	Best	0	0	0	1.29e-04
	Mean	0	2.20e-05	0	4.53e-04
	Time(s)	4.42e+02	2.22e+03	9.42e+02	5.45e+02

Table 7.6 p values of t-test between DHPSO-d and four comparative algorithms on multimodal functions with a significance level of $\alpha = 0.05$ after 30 trials

Dim	p value	f_1	f_2	f_3	f_4	f_5	f_6
50	VS GBPSO	1.15e-01	1.13e-12	9.91e-14	1.40e-01	1.38e-01	1.88e-26
	VS LBPSO	9.69e-09	2.51e-07	4.94e-04	1.40e-01	1.38e-01	1.88e-26
	VS FA	1.18e-08	1.21e-04	4.94e-04	1.42e-01	6.99e-02	5.69e-05
	VS HGLPSO	9.35e-09	6.07e-11	5.03e-04	4.34e-01	5.51e-04	2.98e-19
100	VS GBPSO	1.17e-02	6.37e-38	2.48e-03	9.58e-02	2.59e-10	7.16e-12
	VS LBPSO	2.26e-02	3.62e-06	3.97e-24	1.52e-12	7.31e-01	8.59e-15
	VS FA	2.26e-02	9.64e-32	3.97e-24	2.31e-14	1.60e-12	1.58e-06
	VS HGLPSO	2.26e-02	1.68e-08	3.97e-24	2.93e-12	4.14e-01	3.32e-13

Table 7.7 Performance comparison between DHPSO-d and four comparative algorithms on f_1--f_6 by t-test with a significance level of $\alpha = 0.05$.

Dim	DHPSO-d VS	f_1	f_2	f_3	f_4	f_5	f_6	+/≈/−
50-D	VS GBestPSO	≈	+	+	≈	≈	+	3/3/0
	VS LBestPSO	−	+	+	≈	≈	+	2/2/2
	VS FA	−	−	−	≈	≈	+	1/2/3
	VS HGLPSO	−	+	−	≈	+	+	3/1/2
100-D	VS GBestPSO	+	+	+	≈	+	+	5/1/0
	VS LBestPSO	−	+	−	−	≈	+	2/1/3
	VS FA	−	−	−	−	−	+	1/0/5
	VS HGLPSO	−	+	−	−	≈	+	2/1/3

Analysis of Results on Multimodal Functions f_1--f_6

Table 7.6 shows the p values of t-test on pairs of groups of statistical results between DHPSO-d and four comparative algorithms on six multimodal functions. These tests give a *p-value* which is compared to a constant called α to determine whether a difference is significant or not. If $p < \alpha$, then a test is reported as significant, else the test is reported as nonsignificant.

Table 7.7 shows the results of pairwise comparisons between DHPSO-d and four compared algorithms obtained by the t-test results of table 7.6. In Table 7.7, the symbol (+) represents that the performance obtained by proposed DHPSO-d is significantly superior to the compared algorithm, the symbol (≈) represents that there is no significant difference between DHPSO-d and the compared algorithm, and the symbol (−) represents that the

Table 7.8 Performance comparison between DHPSO-p and five compared algorithms on f_1--f_6 by t-test with a significance level of $\alpha = 0.05$.

Dim	DHPSO-p VS	f_1	f_2	f_3	f_4	f_5	f_6	+/≈/−
50-D	VS GBestPSO	+	+	+	+	+	+	6/0/0
	VS LBestPSO	≈	+	≈	−	≈	+	2/3/1
	VS FA	≈	+	≈	−	−	+	2/2/2
	VS HGLPSO	≈	+	≈	≈	+	+	3/3/0
	VS DHPSO-d	+	+	+	≈	−	≈	3/2/1
100-D	VS GBestPSO	+	+	+	+	+	+	6/0/0
	VS LBestPSO	+	+	≈	+	+	+	5/1/0
	VS FA	≈	+	+	≈	−	+	3/2/1
	VS HGLPSO	−	+	≈	+	+	+	4/1/1
	VS DHPSO-d	+	+	+	+	+	+	6/0/0

performance of DHPSO-d is poorer than the compared algorithm. The symbols (+/≈/−) presented in the last column means that DHPSO-d wins in (+) functions, ties in (≈) functions and loses in (−) functions. Similarly, the symbols shown in the following tables also have the same definitions.

From Table 7.7, it can be seen that DHPSO-d demonstrates the superior performance compared with other algorithms except FA on test functions with dimensions of 50. However, with higher dimensions of 100, the performance of DHPSO-d is poorer than the other compared algorithms except GBestPSO.

Similarly, according to Table 7.8, it can be observed that DHPSO-p demonstrates the superior performance over four algorithms except FA with the dimensions of 50. There is no significant difference between DHPSO-p and FA. It is worth noting that, with the dimensions of 100, DHPSO-p exhibits the best scalability compared with all algorithms.

Analysis of Results on Unimodal Functions f_7--f_{10}

Table 7.9 shows the p values of t-test on pairs of groups of statistical results between DHPSO-d and four comparative algorithms on three unimodal functions. These tests give a *p-value* which is compared to a constant called α to determine whether a difference is significant or

Table 7.9 p values of t-test between DHPSO-d and four comparative algorithms on unimodal functions with a significance level of $\alpha = 0.05$ after 30 trials

Dim	p value	f_7	f_8	f_{10}
50-D	VS GBestPSO	1.17e-02	7.33e-08	1.15e-05
	VS LBestPSO	2.00e-01	4.24e-04	1.70e-07
	VS FA	5.50e-02	4.24e-04	1.70e-07
	VS HGLPSO	2.00e-01	4.24e-04	6.81e-01
100-D	VS GBestPSO	6.63e-06	9.25e-05	5.41e-01
	VS LBestPSO	2.31e-03	6.37e-09	4.49e-14
	VS FA	2.31e-03	6.38e-09	4.24e-14
	VS HGLPSO	2.31e-03	6.37e-09	8.95e-04

Table 7.10 Performance comparison between DHPSO-d and four comparative algorithms on f_7--f_{10} by t-test with a significance level of $\alpha = 0.05$.

Dim	DHPSO-d VS	f_7	f_8	f_9	f_{10}	+/≈/−
50-D	VS GBestPSO	+	+	≈	+	3/1/0
	VS LBestPSO	≈	−	≈	−	0/2/2
	VS FA	≈	−	≈	−	0/2/2
	VS HGLPSO	≈	−	≈	≈	0/3/1
100-D	VS GBestPSO	+	+	≈	≈	2/2/0
	VS LBestPSO	−	−	≈	−	0/1/3
	VS FA	−	−	≈	−	0/1/3
	VS HGLPSO	−	−	≈	−	0/1/3

not. If $p < \alpha$, then a test is reported as significant. If $p > \alpha$, then the test is reported as nonsignificant. It should be noted that all six algorithms could find the optimum solution and have the same performance on unimodal function f_9. In this case, f_9 is excluded from the t-test of table 7.9 and table 7.11.

Table 7.10 shows the significant performance comparisons between DHPSO-d and four algorithms on four unimodal functions with the dimensions of 50 and 100 respectively. As described comparisons in Table 7.10, DHPSO-d performs better than GBestPSO on all unimodal functions with dimensions of 50 and 100. Compared with LBestPSO, FA and HGLPSO, the performance of DHPSO-d is poorer than them. It reveals that DHPSO-d could not perform the superior performance for unimodal problems.

Table 7.11 p values of t-test between DHPSO-p and five comparative algorithms on unimodal functions with a significance level of $\alpha = 0.05$ after 30 trials

Dim	p value	f_7	f_8	f_{10}
50-D	DHPSO-p VS GBestPSO	1.17e-02	7.33e-08	1.15e-05
	DHPSO-p VS LBestPSO	2.61e-01	2.80e-01	4.32e-01
	DHPSO-p VS FA	2.78e-01	6.01e-01	2.11e-01
	DHPSO-p VS HGLPSO	2.61e-01	2.80e-01	2.48e-01
	DHPSO-p VS DHPSO-d	2.61e-01	4.24e-04	1.70e-07
100-D	DHPSO-p VS GBestPSO	1.17e-07	4.33e-09	4.02e-17
	DHPSO-p VS LBestPSO	3.24e-01	3.26e-01	7.56e-23
	DHPSO-p VS FA	3.13e-03	2.78e-01	5.57e-01
	DHPSO-p VS HGLPSO	3.24e-01	3.26e-01	4.28e-02
	DHPSO-p VS DHPSO-d	2.31e-03	6.37e-09	4.24e-14

Table 7.12 Performance comparison between DHPSO-p and five comparative algorithms on f_7--f_{10} by t-test with a significance level of $\alpha = 0.05$.

Dim	DHPSO-p VS	f_7	f_8	f_9	f_{10}	+/≈/−
50-D	VS GBestPSO	+	+	≈	+	3/1/1
	VS LBestPSO	≈	−	≈	≈	0/3/1
	VS FA	≈	≈	≈	≈	0/4/0
	VS HGLPSO	≈	−	≈	+	1/2/1
	VS DHPSO-d	≈	+	≈	+	2/2/0
100-D	VS GBestPSO	+	+	≈	+	3/1/0
	VS LBestPSO	≈	≈	≈	+	1/3/0
	VS FA	+	+	≈	≈	2/2/0
	VS HGLPSO	≈	≈	≈	+	1/3/0
	VS DHPSO-d	+	+	≈	+	3/1/0

Table 7.13 A modified Bonferroni procedure for DHPSO-d VS HGLPSO.

50-D	p-value	Rank	Rank'	New α	Significant
$f6$	2.89e-19	1	9	5.56e-03	Yes (+)
$f2$	6.07e-11	2	8	6.25e-03	Yes (+)
$f1$	9.35e-09	3	7	7.14e-03	Yes (−)
$f8$	4.24e-04	4	6	8.33e-03	Yes (−)
$f3$	5.03e-04	5	5	1.00e-02	Yes (−)
$f5$	5.51e-04	6	4	1.25e-02	Yes (+)
$f7$	2.00e-01	7	3	1.67e-02	No (≈)
$f4$	4.34e-01	8	2	2.50e-02	No (≈)
$f10$	6.81e-01	9	1	5.00e-02	No (≈)
+/≈/−					3/3/3
100-D	p-value	Rank	Rank'	New α	Significant
$f3$	3.97e-24	1	9	5.56e-03	Yes (−)
$f6$	3.32e-13	2	8	6.25e-03	Yes (+)
$f4$	2.93e-12	3	7	7.14e-03	Yes (−)
$f8$	6.37e-09	4	6	8.33e-03	Yes (−)
$f2$	1.68e-08	5	5	1.00e-02	Yes (+)
$f10$	8.95e-04	6	4	1.25e-02	Yes (−)
$f7$	2.31e-03	7	3	1.67e-02	Yes (−)
$f1$	2.26e-02	8	2	2.50e-02	Yes (−)
$f5$	4.14e-01	9	1	5.00e-02	No (≈)
+/≈/−					2/1/6

Table 7.12 presents the comparisons of scalability of DHPSO-p and five algorithms on four unimodal functions with dimensions of 50 and 100 respectively. In Table 7.12, only to LBestPSO, the performance of DHPSO-p is poorer than the compared algorithm on f_8 function with the dimensions of 50. Specifically, when the dimensionality of the problem is increased to 100, DHPSO-p successfully demonstrates the superior scalability over all compared algorithms. It reveals that DHPSO-p shows significantly superior performance over the compared algorithms on unimodal problems.

Analysis of Results on All Functions f_1--f_{10}

In this part, the scalability analysis is performed in the comparison between DHPSO models (DHPSO-d and DHPSO-p) and SHPSO model (HGLPSO) to all test functions $f1$--$f10$.

Table 7.14 A modified Bonferroni procedure for DHPSO-p VS HGLPSO.

50-D	p-value	Rank	Rank'	New α	Significant
$f2$	6.18e-25	1	9	5.56e-03	Yes (+)
$f6$	2.49e-14	2	8	6.25e-03	Yes (+)
$f5$	8.67e-04	3	7	7.14e-03	Yes (+)
$f10$	2.48e-02	4	6	8.33e-03	No ($\approx$)
$f8$	2.80e-02	5	5	1.00e-02	No ($\approx$)
$f1$	1.05e-02	6	4	1.25e-02	No ($\approx$)
$f7$	2.61e-01	7	3	1.67e-02	No ($\approx$)
$f3$	3.34e-01	8	2	2.50e-02	No ($\approx$)
$f4$	7.64e-01	9	1	5.00e-02	No ($\approx$)
$+/\approx/-$					3/6/0
100-D	p-value	Rank	Rank'	New α	Significant
$f2$	1.20e-34	1	9	5.56e-03	Yes (+)
$f6$	6.38e-31	2	8	6.25e-03	Yes (+)
$f5$	1.32e-24	3	7	7.14e-03	Yes (+)
$f4$	5.72e-05	4	6	8.33e-03	Yes (+)
$f1$	8.78e-05	5	5	1.00e-02	Yes ($-$)
$f10$	4.28e-02	6	4	1.25e-02	No ($\approx$)
$f3$	1.68e-01	7	3	1.67e-02	No ($\approx$)
$f7$	3.24e-01	8	2	2.50e-02	No ($\approx$)
$f8$	6.81e-01	9	1	5.00e-02	No ($\approx$)
$+/\approx/-$					4/4/1

There is a problem in conducting multiple significance tests. Because they are probabilistic, it is possible that some results are easy to chance, even random data generated from the same distribution will differ significantly sometimes. This problem can be addressed by a modified Bonferroni procedure which manipulates the α value in a way that protects against capitalization on chance [34]. It should be noted that the test function f_9 is excluded from the modified Bonferroni procedure in Table 7.13 and Table 7.14, because all six algorithms could find the optimum solution and have the same performances on f_9.

In Table 7.13, on the dimensions of 50, the number of (3/3/3) shown in the last column means that there is no significantly difference in performance between DHPSO-d and HGLPSO to all test functions. On the high dimensions of 100, 2/1/6 means that the performance of DHPSO-d is poorer than HGLPSO.

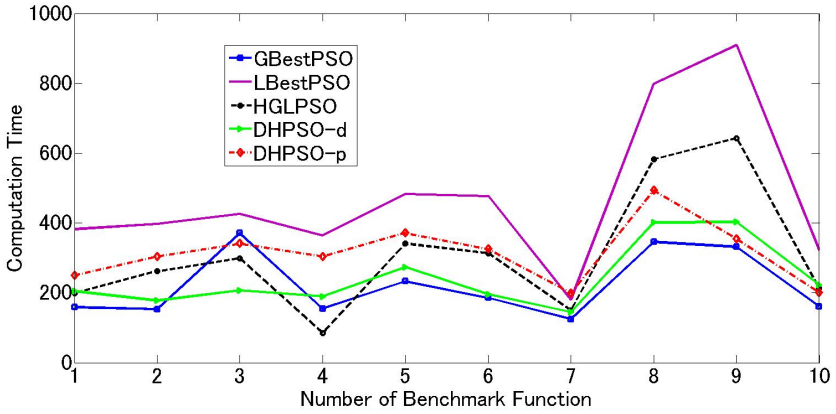


Fig. 7.4 Computation time of different algorithms on benchmark functions with dimensions of 50 after 30 times of trails.

In Table 7.14, on the dimensions of 50, 3/6/0 means that DHPSO-p is significantly superior on 3 test functions. On the dimensions of 100, 4/4/1 means that DHPSO-p is significantly superior on 4 functions, while HGLPSO is only significantly superior on 1 function (f_1). It is clear that in many of the test cases, DHPSO-p has better performance than the HGLPSO, especially in the high dimensional cases.

7.3.3 Discussion

Based on the previous empirical analysis, the scalability of proposed DHPSO-d and DHPSO-p will be examined from a practical point of view, namely, by evaluating the optimum performance and running time on each algorithm. In general, if there is no significantly difference in optimal solution, we will prefer the algorithm with less running time. Following this viewpoint, an overall picture of the scalability of DHPSO-d and DHPSO-p is depicted as follows.

▷ To the multimodal functions: f_1--f_6

From Fig.7.4 it can be observed that the computation time of DHPSO-d is less than the most of algorithms except GBestPSO. As shown in Fig.7.4 and Fig.7.5, the computation time of DHPSO-p is less than LBestPSO and FA. To summarize, on 50 dimensional

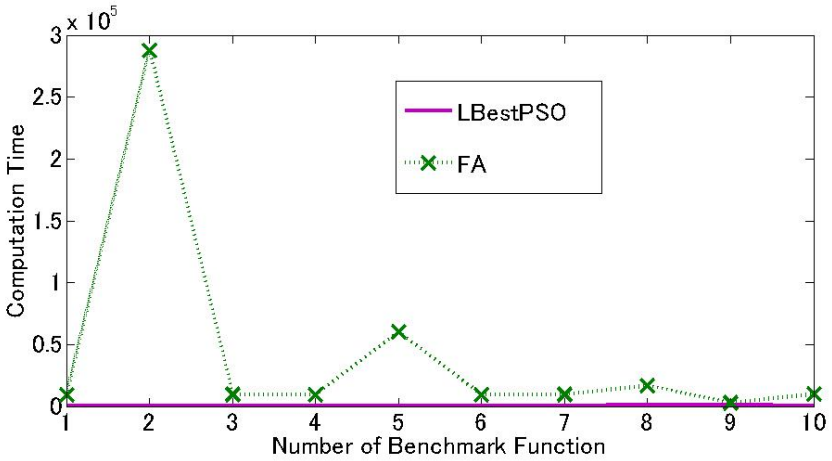


Fig. 7.5 Computation time of LBestPSO and FA on benchmark functions with dimensions of 50 after 30 times of trails.

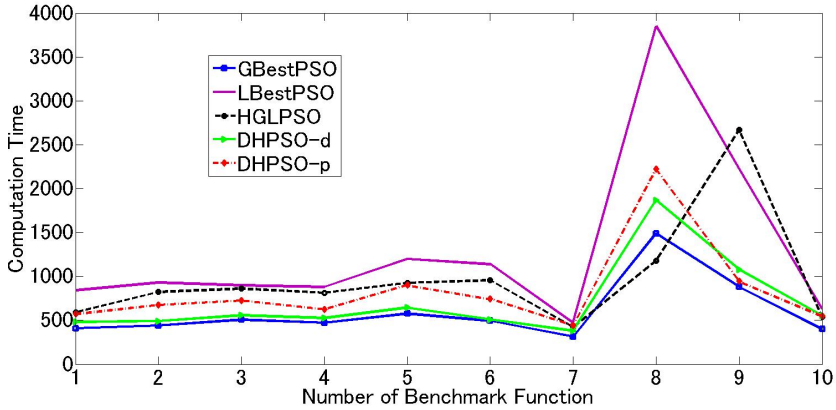


Fig. 7.6 Computation time of different algorithms on benchmark functions with dimensions of 100 after 30 times of trails.

functions, DHPSO-d exhibits the superior solution accuracy and faster convergence speed over two homogeneous PSO algorithms (GBestPSO and LBestPSO) and one SHPSO algorithm (HGLPSO). This implies that the method of different particles dynamically using different rules for updating their positions is beneficial to the solution improvement during the optimization process. However, DHPSO-d did show some deterioration in performance for 100 dimensions on test functions, which illustrates that the stagnation of the best-so-far solution couldn't be eliminated only by the dynamic update-rule heterogeneity. On the other hand, we note that DHPSO-p has the best scalability over all five algorithms on multimodal functions, even in the high dimensional cases. It reveals that the strategy of dynamic model-of-influence heterogeneity in DHPSO-p could successfully improve the performance of DHPSO-d for multimodal function, especially in the high dimensional cases.

$\triangleright$ To the unimodal functions: f_7--f_{10}

Table 7.10 reveals that DHPSO-d could not demonstrate the desirable performance for the unimodal functions. Although dynamic update-rule heterogeneity is efficient for searching the global optimum of multimodal problems, tuning the algorithm to find the optimal solution on unimodal problem is ineffective. Different with DHPSO-d, DHPSO-p exhibits the superior scalability over all algorithms except LBestPSO. These results point towards the idea that mechanisms allowing the swarm or sub-swarms to choose their informers during the optimization process could be more beneficial to address the exploration and exploitation trade-off problem than designing sophisticated update rules to make particles capable of doing both things. This means that dynamic heterogeneous structure enables us to move design complications from the individual level to the swarm level. The results show that DHPSO-p has the best scalability over all five algorithms as the dimensionality of unimodal problem increased to 100.

$\triangleright$ To all functions: f_1--f_{10}

According to Fig.7.4 and Fig.7.6, it can be seen that the computation time of DHPSO-d is less than HGLPSO, and there is no significantly difference in computation time between DHPSO-p and HGLPSO. In terms of solution quality and computation time, DHPSO-d outperforms HGLPSO on 50 dimensional functions, while HGLPSO has better scalability than DHPSO-d on the high dimensions of 100. According to the previous analysis of DHPSO-d, the stagnation of optimal solution can't be eliminated only by the dynamic update-rule heterogeneity as the dimensionality of the problem is increased to 100. Another result of the experimental analysis of DHPSO-d is that the strategy of dynamic model-of-influence heterogeneity is more beneficial to solution improvement than designing sophisticated update rules. We note that HGLPSO is a static model-of-influence heterogeneity combined with the $gbest$ model and the $lbest$ model. In conclusion, the results presented in this study have shown that the model-of-influence heterogeneity algorithm is more scalable to higher dimensions compared to the update-rule heterogeneity algorithm. In addition, DHPSO-p has better scalability than HGLPSO on all functions with dimensions of 50 and 100. Our results show that the dynamic self-adaptive heterogeneity is able to address the exploration-exploitation trade-off problem, and provide the excellent optimal solution of a problem.

7.4 Summary

In this chapter, we propose two DHPSO models, DHPSO-d with dynamic update-rule heterogeneity and DHPSO-p with dynamic model-of-influence heterogeneity. In DHPSO-d, particles dynamically use different rules for updating their position when the triggering events are confirmed. In DHPSO-p, a global gbest model and a pairwise connection model are automatically selected by the triggering configuration. The proposed triggering events are confirmed by keeping track of the frequency of the unchanged global best position ($\boldsymbol{p}_g$) for a number of iterations. This information is then used to select a new heterogeneous structure when $\boldsymbol{p}_g$ is considered stagnant. In order to investigate the scalability of DHPSO-d and

DHPSO-p, a series of experiments with four state-of-the-art algorithms are performed on ten well-known optimization problems. One of the results of DHPSO-d analysis is that the method of dynamic update-rule heterogeneity is beneficial to the solution improvement during the optimization process. Another result analysis of DHPSO-d shows that the stagnation of optimal solution can't be eliminated only by the dynamic update-rule heterogeneity as the dimensionality of the problem increases. Moreover, empirical study of DHPSO-p shows the adaptive model-of-influence heterogeneity algorithm is more scalable to higher dimensions than dynamic update-rule heterogeneity algorithm. In conclusion, the dynamic self-adaptive heterogeneous structure is able to effectively address the exploration-exploitation trade-off problem, and provides the excellent optimal solution of a problem simultaneously.

Chapter 8

Conclusion and Future Work

Two major problems encountered by many swarm intelligence algorithms when handling global optimization, and especially in large scale global optimization are the premature convergence and exploration-exploitation trade-off problems. Particle Swarm Optimization (PSO) despite being an efficient method to solve a variety of global optimal problems, also suffers from these major problems. To handle these challenges, several variants of PSO have been developed in this thesis, each of which implementing the biological metaphor in their own particular way. Basically, there are two main categories of our proposed variants, the first consisting of five variants of homogeneous PSO which is used for multimodal optimization and large scale optimization problems, and lastly two variants of dynamic heterogeneous PSO which is used for complex real-world problems.

8.1 The Knowledge Gained from This Study

Particle Swarm Optimization (PSO) is a population based and intelligent method for solving a wide range of optimization problems, which is inspired by the social behaviour of biological organisms, like birds flocking while searching for a food source. Despite being an efficient method, it also suffers from the premature convergence problem, especially when solving

the multimodal problems. For instance, all particles share its swarm's best experience (the global best) that can lead the particles to cluster around the global best. If the global best is located near a local minimum, escaping from the local optimum becomes difficult and the swarm suffers diversity loss near the local minimum. In order to address this problem, several variants of PSO are proposed in this thesis.

First of all, in developed Fitness Predator Optimization (FPO), the individual competition method is proposed to increase the dispersion of particles (swarm diversity), which greatly reduces the possibility of all of the individuals moving toward the same position. Secondly, the elite team generated by roulette wheel selection also is an effective approach to maintain the swarm diversity by stimulating the competition among the elite team, strengthening the ability of the elite to find the best global optimal solution. In addition, the elite team proposed in FPO also provides a potential parallel computation mechanism, which is a promising method for FPO to decrease the computation time for solving large scale optimization. To enhance the global exploration capability of the FPO algorithm for the high multimodal problem, a modified paralleled virtual team approach is developed for FPO, namely DFPO. In DFPO, each particle is self-organized a virtual team with several nearest neighbouring particles. The main function of this dynamic virtual team is to build a paralleled information-exchange system. The potential global best position can be widely spread by this paralleled information-exchange system based on these virtual teams. It is able to strengthen the swarm's global searching effectiveness. On the other hand, the individual competition generated by the elite team is able to provide excellent exploitation solutions and utilizing local minima avoidance. In order to prevent the particles from cluster around the local optimum, a random team size selection strategy is defined in DFPO named as DFPO-r, which based on the fact that a dynamic virtual team with a higher degree of population diversity is able to help DFPO-r alleviate the premature convergence and strengthen on the global exploration simultaneously. Experimental results demonstrate that DFPO-r has

a more robust performance in most cases compared with DFPO. In conclusion, the strategies of an elite team stimulated by the individual competition, a paralleled information-exchange system based on the dynamic virtual teams and the random virtual team size selection are the important contributions to the improved global exploration capability and local minima avoidance for FPO.

In the vast majority of PSO models, the swarm is composed of homogeneous individuals. However, heterogeneity is ubiquitous in nature. Recently, the Static Heterogeneous Particle Swarm Optimization (SHPSO) has been studied by more and more researchers. In SHPSO, the different search behaviours assigned to particles during initialization do not change during the search process. As a consequence of this, the inappropriate population size of exploratory particles could leave the SHPSO with great difficulties of escaping local optima. This motivated our attempt to improve the performance of SHPSO by introducing the dynamic heterogeneity. In this thesis, two variants of dynamic Heterogeneous PSO, namely DHPSO-d with differential update-rule heterogeneity and DHPSO-p with dynamic model-of-influence heterogeneity are proposed for complex real-world problems. In DHPSO-d, particles dynamically use different rules for updating their position when the triggering events are confirmed. The experiment of DHPSO-d illustrates that the method of different particles dynamically using different rules for updating their positions is beneficial to the solution improvement during the optimization process. However the stagnation of optimal solution can't be eliminated only by the dynamic update-rule heterogeneity as the dimensionality of the problem increases. Unlike DHPSO-d, in DHPSO-p, two proposed types of topology models are proposed which gives the particle models different mechanisms of choosing their informers when the swarm being trapped in the local optimal solution. Experimental results show that the strategy of dynamic model-of-influence heterogeneity in DHPSO-p could successfully improve the performance of DHPSO-d for multimodal function, especially in the high dimensional cases. These results point towards the idea that mechanisms allowing the

swarm or sub-swarms to choose their informers during the optimization process could be more beneficial to address the exploration and exploitation trade-off problem than designing sophisticated update rules to make particles capable of doing both things. It means that dynamic heterogeneous structure enables us to move design complications from the individual level to the swarm level.

8.2 Conclusion

In this study, five variants of homogeneous PSO have been developed for multimodal optimization and large scale optimization problems, and two variants of dynamic Heterogeneous PSO for complex real-world problems. First of all, an individual competition strategy is proposed to the new variant of PSO algorithm, namely FPO, for multimodal problems. The development of individual competition plays an important role of the diversity conservation in the population, which is crucial for preventing premature convergence in multimodal optimization. Experimental results show that FPO is able to provide excellent performance of global exploration and local minima avoidance simultaneously. However, to the higher dimensionality of multimodal problem, the slow convergence speed becomes the bottleneck of FPO. Consequently, a dynamic team model is utilized in FPO named DFPO to accelerate an early convergence rate. The main function of this team model is to build a paralleled information-exchange system, which is able to enhance the global optimal capability and speed up convergence rate during the process of searching. Furthermore, the strategy of team size selection is also presented in DFPO which is based on the idea that a team size with a higher population diversity is able to prevent solutions from clustering too tightly in the local search space. Then DFPO with dynamic team size selection strategy is provided as DFPO-r. The excellent global optimizing abilities of DFPO and DFPO-r show that the paralleled exchanging information system spread by the dynamic virtual teams could help to accelerate the early convergence rate and improve the global searching capability.

Secondly, the combination of FPO and FCM (FPO-FCM) algorithm is proposed to avoid the premature convergence for clustering optimization. In FPO-FCM, the position of each particle represents a set of clustering centroids, a number of particles composed of a swarm of FPO-FCM. The objective function of FCM is used for evaluating the generalized solutions. An experiment of five benchmark data sets show that FPO-FCM has a better global optimization ability and the local optimum avoidance for clustering optimization. The use of hybrid algorithms to deal with specific problems proves that hybridization is a powerful tool far beyond the individual algorithms.

To handle the large scale global optimization problem, a variant of the modified BBPSO algorithm incorporating the Differential Evolution (DE) approach, namely BBPSO-DE is developed to improve the swarm's global search capability as the search space's dimensionality increases. Three strategies are proposed for BBPSO-DE, specifically introducing ring topology of memory-swarm, integrating differential evolution in particle's local best position $lbest_i$ and introducing the centroid position of all particles to the particle's update formula. The outstanding performance of BBPSO-DE should be attributed to these strategies by means of strengthening the swarm's global exploration while successfully maintenance of the local exploitation.

In population-based algorithms, finding the global optimal solution of a problem is based on two cornerstones; namely exploring all over the search space to find promising regions, and exploiting the identified promising regions to tune the search for the global optimum. It is worth of noting that modelled with populations of heterogeneous individuals, the optimization algorithm has a ability to maintain an appropriate balance between exploration and exploitation throughout the search process. Experiment of DHPSO-r show that the method of dynamic update-rule heterogeneity is beneficial to the solution improvement during the optimization process. Empirical study of DHPSO-p shows the adaptive model-of-influence heterogeneity algorithm is more scalable to the higher dimensions than dynamic update-rule

heterogeneity algorithm. In conclusion, the dynamic self-adaptive heterogeneous structure is able to effectively address the exploration-exploitation trade-off problem, and provides an excellent optimal solution of a problem.

8.3 Future Work

In our previous work, the assessment done on presented variants of PSO algorithms is based on experimental comparisons. In this case, the use of descriptive statistics, such as the sample mean,the standard deviation and the *t*-test, is not sufficient. To ensure fair and meaningful comparisons of the presented algorithms, different statistical tests should be carried out to analyze and compare the algorithms. For instance, the Mann Whitney U test and a modified Bonferroni procedure will be used for our further evaluation analysis. There are also many commercial (e.g., SAS/STAT, XPSS) and free softwares (e.g., R, MacAnova) which are available for conducting such an analysis.

With the low price and easy access, the Graphic Processing Unit (GPU) has gained much popularity in general purpose computing. Accelerating swarm intelligence algorithms for solving large scale complex problems on GPU platform has attracted the attention of many researchers due to their applicability to many engineering and scientific problems. With improvements in its programmability and the emergence of more handy development toolkits, more and more swarm intelligence algorithms are implanted in the GPU hardware to leverage the rapidly increasing performance of GPU. In this thesis, Gaussian distribution used for BBPSO-DE is time-consumption and the computation time of BBPSO-DE is greatly increased with the increasing dimensionality of LSGO problems. Accelerating BBPSO-DE for solving complex large scale problems on GPU platform may lead to a more efficient algorithm and our future work is focusing on this problem. Furthermore, comparison with other state-of-the-art algorithms for LSGO problems should be further explored.

One of the most interesting trends in the last years is on hybrid optimization methods. Indeed, more and more papers are published on the hybridization of Swarm Intelligence (SI) algorithms with other techniques for optimization. Hybridization is not restricted to the combination of different SI algorithms but includes the use of hybrid algorithms that combine local search or exact algorithms. Moreover, the combination of concepts from different metaheuristics algorithm and different research areas can lead to interesting new approaches, such as FPO-FCM, which combines fuzzy logic and several optimization techniques. Such hybridizations can be used to take advantage of strengths from each algorithm in order to improve algorithms' performance for more effective and efficient problem-solving. And our future research is to expand this approach for more fuzzy clustering optimization problems.

Besides, for some more complicated applications, for example, dynamic systems, higher level intelligence should be integrated into the SI algorithm. Environment detection and response mechanism are new intelligence methods. This is population-level intelligence, in this level, we can integrate more human knowledge and intelligence to the algorithm to accommodate complex systems.

Bibliography

[1] Alatas, B., Akin, E., and Ozer, A. B. (2009). Chaos embedded particle swarm optimization algorithms. *Chaos, Solitons & Fractals*, 40(4):1715--1734.

[2] Anusuya, S. and Parthiban, L. (2014). Efficient hybridized fuzzy clustering with fcm-iqpso for biomedical datasets. *Annual Review & Research in Biology*, 4(17).

[3] Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

[4] Bellman, R. E. (2015). *Adaptive control processes: a guided tour*. Princeton university press.

[5] Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers.

[6] Bezdek, J. C. (2011). Fuzzy c-means cluster analysis. *Scholarpedia*, 6(7):2057.

[7] Bratton, D. and Kennedy, J. (2007). Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium*, pages 120--127. IEEE.

[8] Brooks, S. P. and Morgan, B. J. (1995). Optimization using simulated annealing. *The Statistician*, pages 241--257.

[9] Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1--27.

[10] Chen, C. P. and Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314--347.

[11] Chen, Y.-P., Peng, W.-C., and Jian, M.-C. (2007). Particle swarm optimization with recombination and dynamic linkage discovery. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(6):1460--1470.

[12] Cheng, R. and Jin, Y. (2015). A competitive swarm optimizer for large scale optimization. *Cybernetics, IEEE Transactions on*, 45(2):191--204.

[13] Cheng, S., Shi, Y., Qin, Q., Ting, T., and Bai, R. (2014a). Maintaining population diversity in brain storm optimization algorithm. In *IEEE Congress on Evolutionary Computation*, pages 3230--3237. IEEE.

[14] Cheng, S., Shi, Y., Qin, Q., Zhang, Q., and Bai, R. (2014b). Population diversity maintenance in brain storm optimization algorithm. *Journal of Artificial Intelligence and Soft Computing Research*, 4(2):83--97.

[15] Clerc, M. (2010). *Particle swarm optimization*, volume 93. John Wiley & Sons.

[16] Clerc, M. and Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6(1):58--73.

[17] Črepinšek, M., Liu, S.-H., and Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 45(3):35.

[18] De Oca, M. A. M., Peña, J., Stützle, T., Pinciroli, C., and Dorigo, M. (2009). Heterogeneous particle swarm optimizers. In *IEEE Congress on Evolutionary Computation*, pages 698--705. IEEE.

[19] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29--41.

[20] Eberhart, R., Simpson, P., and Dobbins, R. (1996). *Computational intelligence PC tools*. Academic Press Professional, Inc.

[21] Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. *Proc. 6th International Symposium on Micro Machine and Human Science*, pages 39--43.

[22] Egrioglu, E., Aladag, C. H., and Yolcu, U. (2013). Fuzzy time series forecasting with a novel hybrid approach combining fuzzy c-means and neural networks. *Expert Systems with Applications*, 40(3):854--857.

[23] Eiben, A. E. and Schippers, C. A. (1998). On evolutionary exploration and exploitation. *Fundamenta Informaticae*, 35(1-4):35--50.

[24] El-Dib, A. A., Youssef, H. K., El-Metwally, M., and Osman, Z. (2006). Maximum loadability of power systems using hybrid particle swarm optimization. *Electric Power Systems Research*, 76(6):485--492.

[25] Engelbrecht, A. P. (2006). *Fundamentals of computational swarm intelligence*. John Wiley & Sons.

[26] Engelbrecht, A. P. (2010). Heterogeneous particle swarm optimization. In *Swarm Intelligence*, pages 191--202. Springer.

[27] Friedrich, T., Oliveto, P. S., Sudholt, D., and Witt, C. (2009). Analysis of diversity-preserving mechanisms for global exploration. *Evolutionary Computation*, 17(4):455--476.

[28] Gandomi, A. H. and Alavi, A. H. (2012). Krill herd: a new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, 17(12):4831--4845.

[29] Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2):95--99.

[30] Gupta, K. and Shrivastava, M. (2010). Web usage mining clustering using hybrid fcm with ga. *International Journal of Advanced Computer Research*.

[31] Hansell, M. (2007). *Built by animals: the natural history of animal architecture*. OUP Oxford.

[32] Hathaway, R. J. and Bezdek, J. C. (1995). Optimization of clustering criteria by reformulation. *Fuzzy Systems, IEEE Transactions on*, 3(2):241--245.

[33] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504--507.

[34] Hommel, G. (1989). A comparison of two modified bonferroni procedures. *Biometrika*, 76(3):624--625.

[35] Hsieh, S.-T., Sun, T.-Y., Liu, C.-C., and Tsai, S.-J. (2008). Solving large scale global optimization using improved particle swarm optimizer. In *IEEE Congress on Evolutionary Computation*, pages 1777--1784. IEEE.

[36] Hsieh, S.-T., Sun, T.-Y., Liu, C.-C., and Tsai, S.-J. (2009). Efficient population utilization strategy for particle swarm optimizer. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):444--456.

[37] Hu, J., Wang, Y., Zhou, E., Fu, M. C., and Marcus, S. I. (2012). A survey of some model-based methods for global optimization. In *Optimization, Control, and Applications of Stochastic Systems*, pages 157--179. Springer.

[38] Jamil, M. and Yang, X.-S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150--194.

[39] Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1):3--12.

[40] Karaboga, D. and Akay, B. (2009). A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, 31(1-4):61--85.

[41] Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459--471.

[42] Kenndy, J. and Eberhart, R. C. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942--1948. IEEE.

[43] Kennedy, J. (1998). The behavior of particles. In *Evolutionary Programming VII*, pages 579--589. Springer.

[44] Kennedy, J. (1999). Small worlds and mega minds: effects of neighborhood topology on particle swarm performance. In *IEEE Congress on Evolutionary Computation*, pages 1931--1938. IEEE.

[45] Kennedy, J. (2003). Bare bones particle swarms. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, pages 80--87. IEEE.

[46] Kennedy, J. and Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 5, pages 4104--4108. IEEE.

[47] Kennedy, J., Kennedy, J. F., and Eberhart, R. C. (2001). *Swarm intelligence*. Morgan Kaufmann.

[48] Kennedy, J. and Mendes, R. (2002). Population structure and particle swarm performance. In *IEEE Congress on Evolutionary Computation*. IEEE.

[49] Kennedy, J. and Mendes, R. (2006). Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. *IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews*, 36(4):515--519.

[50] Khan, A., Jaffar, M. A., and Choi, T.-S. (2013). Som and fuzzy based color image segmentation. *Multimedia tools and applications*, 64(2):331--344.

[51] Khanesar, M. A., Teshnehlab, M., and Shoorehdeli, M. A. (2007). A novel binary particle swarm optimization. In *Control & Automation, 2007. MED'07. Mediterranean Conference on*, pages 1--6. IEEE.

[52] Larranaga, P. and Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer Science & Business Media.

[53] Li, X. (2003). A new intelligent optimization-artificial fish swarm algorithm. *Doctor thesis*.

[54] Li, X. (2010). Niching without niching parameters: particle swarm optimization using a ring topology. *Evolutionary Computation, IEEE Transactions on*, 14(1):150--169.

[55] Liang, J. and Suganthan, P. N. (2005). Dynamic multi-swarm particle swarm optimizer. In *IEEE Swarm Intelligence Symposium*, pages 124--129. IEEE.

[56] Liang, J. J., Qin, A. K., Suganthan, P. N., and Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE transactions on evolutionary computation*, 10(3):281--295.

[57] Lim, D., Jin, Y., Ong, Y.-S., and Sendhoff, B. (2010). Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 14(3):329--355.

[58] Liu, B., Wang, L., Jin, Y.-H., Tang, F., and Huang, D.-X. (2006). Directing orbits of chaotic systems by particle swarm optimization. *Chaos, Solitons & Fractals*, 29(2):454--461.

[59] Liu, H.-C., Yih, J.-M., Wu, D.-B., and Liu, S.-W. (2008). Fuzzy c-mean clustering algorithms based on picard iteration and particle swarm optimization. In *Education Technology and Training, 2008. and 2008 International Workshop on Geoscience and Remote Sensing. ETT and GRS 2008. International Workshop on*, volume 2, pages 838--842. IEEE.

[60] Liu, L.-Z., Wu, F.-X., and Zhang, W.-J. (2012). Inference of biological s-system using the separable estimation method and the genetic algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(4):955--965.

[61] Lynn, N. and Suganthan, P. N. (2015). Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. *Swarm and Evolutionary Computation*, 24:11--24.

[62] Mahdavi, S., Shiri, M. E., and Rahnamayan, S. (2015). Metaheuristics in large-scale global continues optimization: a survey. *Information Sciences*, 295:407--428.

[63] Mendes, R., Kennedy, J., and Neves, J. (2004). The fully informed particle swarm: simpler, maybe better. *IEEE transactions on evolutionary computation*, 8(3):204--210.

[64] Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Adv. Eng. Softw.*, 69:46--61.

[65] Mühlenbein, H. and Paass, G. (1996). From recombination of genes to the estimation of distributions i. binary parameters. In *International Conference on Parallel Problem Solving from Nature*, pages 178--187. Springer.

[66] Naka, S., Genji, T., Yura, T., and Fukuyama, Y. (2003). A hybrid particle swarm optimization for distribution state estimation. *IEEE Transactions on Power systems*, 18(1):60--68.

[67] Noman, N. and Iba, H. (2007). Inferring gene regulatory networks using differential evolution with local search heuristics. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 4(4):634--647.

[68] Omran, M. G., Engelbrecht, A. P., and Salman, A. (2007). Differential evolution based particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 112--119. IEEE.

[69] Oo, N. W. (2008). A comparison study on particle swarm and evolutionary particle swarm optimization using capacitor placement problem. In *Power and Energy Conference, 2008. PECon 2008. IEEE 2nd International*, pages 1208--1211. IEEE.

[70] Ortiz, A., Palacio, A. A., Górriz, J. M., Ramírez, J., and Salas-González, D. (2013). Segmentation of brain mri using som-fcm-based method and 3d statistical descriptors. *Computational and mathematical methods in medicine*, 2013.

[71] Pan, Q.-K., Tasgetiren, M. F., and Liang, Y.-C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research*, 35(9):2807--2839.

[72] Pan, Q.-K. and Wang, L. (2008). No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. *The International Journal of Advanced Manufacturing Technology*, 39(7-8):796--807.

[73] Pandey, H. M., Chaudhary, A., and Mehrotra, D. (2014). A comparative review of approaches to prevent premature convergence in ga. *Applied Soft Computing*, 24:1047--1077.

[74] Parsopoulos, K. E. and Vrahatis, M. N. (2002). Recent approaches to global optimization problems through particle swarm optimization. *Natural computing*, 1(2-3):235--306.

[75] Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008:3.

[76] Potter, M. A. and De Jong, K. A. (1994). A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 249--257. Springer.

[77] Runkler, T. A. and Katz, C. (2006). Fuzzy clustering by particle swarm optimization. In *Fuzzy Systems, 2006 IEEE International Conference on*, pages 601--608. IEEE.

[78] Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. In *IEEE Congress on Evolutionary Computation*, pages 69--73. IEEE.

[79] Shi, Y. and Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE.

[80] Shiqin, Y., Jianjun, J., and Guangxing, Y. (2009). A dolphin partner optimization. In *Proceedings of the 2009 WRI Global Congress on Intelligent Systems - Volume 01*, GCIS '09, pages 124--128.

[81] Shiqin Yang, Jia Guo, Y. S. (2015). Validity index mixed pseudo f for a new fuzzy cluster analysis algorithm. pages 506--509. International Symposium on Artificial Life and Robotics (ISAROB).

[82] Siddique, N. and Adeli, H. (2013). *Computational intelligence: synergies of fuzzy logic, neural networks and evolutionary computing*. John Wiley & Sons.

[83] Silva, A., Neves, A., and Costa, E. (2002). Chasing the swarm: a predator prey approach to function optimisation. In *Proceedings of the MENDEL2002--8th International Conference on Soft Computing. Brno, Czech Republic*.

[84] Spencer, H. (1864). *The Principles of Biology*. Number v. 1 in A system of synthetic philosophy. vol. II-[III]. William and Norgate.

[85] Storn, R. and Price, K. (1997). Differential evolution--a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341--359.

[86] Sun, J., Feng, B., and Xu, W. (2004). Particle swarm optimization with particles having quantum behavior. In *IEEE Congress on Evolutionary Computation*.

[Surjanovic and Bingham] Surjanovic, S. and Bingham, D. Virtual library of simulation experiments: Test functions and datasets. Retrieved September 17, 2016, from http://www.sfu.ca/~ssurjano.

[88] Tasgetiren, M. F., Liang, Y.-C., Sevkli, M., and Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research*, 177(3):1930--1947.

[89] Tenne, Y. and Goh, C.-K. (2010). *Computational intelligence in expensive optimization problems*, volume 2. Springer Science & Business Media.

[90] Wang, B., Tai, N.-l., Zhai, H.-q., Ye, J., Zhu, J.-d., and Qi, L.-b. (2008). A new armax model based on evolutionary algorithm and particle swarm optimization for short-term load forecasting. *Electric Power Systems Research*, 78(10):1679--1685.

[91] Wang, H. and Qian, F. (2008). An improved particle swarm optimizer with behavior-distance models and its application in soft-sensor. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 4473--4478. IEEE.

[92] Wang, H., Yang, S., Xu, W., and Sun, J. (2007). Scalability of hybrid fuzzy c-means algorithm based on quantum-behaved pso. In *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*, volume 2, pages 261--265. IEEE.

[93] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *nature*, 393(6684):440--442.

[94] XUE, J., SUN, J., YANG, G., SUN, Y., and HE, H. (2006). Fuzzy cluster analysis algorithm based on emergency events response. *Computer Engineering*, 1:069.

[95] Yang, F., Sun, T., and Zhang, C. (2009). An efficient hybrid data clustering method based on k-harmonic means and particle swarm optimization. *Expert Systems with Applications*, 36(6):9847--9852.

[96] Yang, S. and Sato, Y. (2014). Fuzzy clustering with fitness predator optimizer for multivariate data problems. In *Simulated Evolution and Learning*, pages 155--166. Springer.

[97] Yang, S. and Sato, Y. (2015). Paralleled fitness predator optimizer with modified dynamic virtual team for multimodal problems. In *IEEE Congress on Evolutionary Computation*, pages 1551--1558. IEEE.

[98] Yang, S. and Sato, Y. (2016). Modified bare bones with differential evolution for large scale optimization. In *2016 IEEE World Congress on Computational Intelligence*, pages 2760--2767. IEEE.

[99] Yang, S. and Sato, Y. (2017a). Heterogeneous particle swarm optimization. *IEICE Transactions on Information and Systems*, 100(2):247--255.

[100] Yang, S. and Sato, Y. (2017b). Swarm intelligence algorithm based on competitive predators with dynamic virtual teams. *Journal of Artificial Intelligence and Soft Computing Research*, 7(2):87--101.

[101] Yang, X.-S. (2010a). *Nature-inspired metaheuristic algorithms*. Luniver press.

[102] Yang, X.-S. (2010b). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65--74. Springer.

[103] Zhan, Z.-H., Zhang, J., Li, Y., and Chung, H. S.-H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6):1362--1381.

[104] Zhan, Z.-H., Zhang, J., Li, Y., and Shi, Y.-H. (2011). Orthogonal learning particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 15(6):832--847.

[105] Zhang, H., Fernández-Vargas, J. A., Rangaiah, G. P., Bonilla-Petriciolet, A., and Segovia-Hernández, J. G. (2011a). Evaluation of integrated differential evolution and unified bare-bones particle swarm optimization for phase equilibrium and stability problems. *Fluid Phase Equilibria*, 310(1):129--141.

[106] Zhang, H., Kennedy, D. D., Rangaiah, G. P., and Bonilla-Petriciolet, A. (2011b). Novel bare-bones particle swarm optimization and its performance for modeling vapor--liquid equilibrium data. *Fluid Phase Equilibria*, 301(1):33--45.

[107] Zhong, W.-h., Zhang, J., and Chen, W.-n. (2007). A novel discrete particle swarm optimization to solve traveling salesman problem. In *2007 IEEE Congress on Evolutionary Computation*, pages 3283--3287. IEEE.

[108] Zhou, Y. and Tan, Y. (2009). Gpu-based parallel particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, pages 1493--1500. IEEE.

Acknowledgements

I want to express the deepest appreciation to my supervisor, Professor Yuji Sato, for his constant guidance and encouragement. He has been actively interested in my work and has always been available to advise me. I am very grateful for his patience, motivation, enthusiasm, and immense knowledge. Without his guidance and persistent help this dissertation would not have been possible.

I would also genuinely acknowledge professor Runhe Huang, who introduced me to Hosei University and gave great effort to help me improve my research skills during the first three years as my PhD advisor.

I appreciate the professors of CIS who have provided me with kind help for my research work and daily life.

I would like to thank Dr.Mads, providing many great suggestions and paper proofreading for my doctoral dissertation.

I am also very grateful to the present and past members in the lab, who give me helpful and disinterested assistance during my research period.

I greatly appreciate the support received through the staff in the administration office, for their kind assistance and strong support, which helped me much in the living and research.

Last but not the least, I would like to thank my family for their understanding and support.

Appendix A

List of Research Papers

Refereed Journal Papers

First Author

[1] Yang, S. and Sato, Y. (2017). Swarm intelligence algorithm based on competitive predators with dynamic virtual teams. *Journal of Artificial and Soft Computing Research (JAISCR)*, 7(2):87--101.

[2] Yang, S. and Sato, Y. (2017). Dynamic heterogeneous particle swarm optimization. *IEICE Transactions on Information and Systems*, 100(2):247--255.

Refereed Conference Papers

First Author

[3] Yang, S. and Sato, Y. (2014). Fitness predator optimizer to avoid premature convergence for multimodal problems. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 258--263. IEEE.

[4] Yang, S. and Sato, Y. (2014). Fuzzy clustering with fitness predator optimizer for multivariate data problems. In *Simulated Evolution and Learning*, pages 155--166. Springer.

[5] Yang, S., Guo, J. and Sato, Y. (2015). Validity index mixed pseudo f for a new fuzzy cluster analysis algorithm. pages 506--509. *International Symposium on Artificial Life and Robotics (ISAROB)*.

[6] Yang, S. and Sato, Y. (2015). Paralleled fitness predator optimizer with modified dynamic virtual team for multimodal problems. In *IEEE Congress on Evolutionary Computation*, pages 1551--1558. IEEE.

[7] Yang, S. and Sato, Y. (2016). Modified Bare Bones with Differential Evolution for Large Scale Optimization. In *IEEE World Congress on Computational Intelligence (WCCI)*, pages 2760--2767. IEEE.

Corresponding Chapters with Papers

Chapter 1. Introduction

Chapter 2. Standard Particle Swarm Optimization (SPSO)

Chapter 3. Fitness Predator Optimizer for Multimodal Problems

[3] Yang, S. and Sato, Y. (2014). Fitness predator optimizer to avoid premature convergence for multimodal problems. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 258--263. IEEE.

Chapter 4. A New Hybrid Fuzzy Clustering Algorithm for Multivariate Data

[4] Yang, S. and Sato, Y. (2014). Fuzzy clustering with fitness predator optimizer for multivariate data problems. In *Simulated Evolution and Learning*, pages 155--166. Springer.

[5] Shiqin Yang, Jia Guo, Y. S. (2015). Validity index mixed pseudo f for a new fuzzy cluster analysis algorithm. pages 506--509. *International Symposium on Artificial Life and Robotics (ISAROB)*.

Chapter 5. Dynamic Virtual Teams for Fitness Predator Optimizer

[6] Yang, S. and Sato, Y. (2015). Paralleled fitness predator optimizer with modified dynamic virtual team for multimodal problems. In *IEEE Congress on Evolutionary Computation*, pages 1551--1558. IEEE.

[1] Yang, S. and Sato, Y. (2017). Swarm intelligence algorithm based on competitive predators with dynamic virtual teams. *Journal of Artificial and Soft Computing Research (JAISCR)*, 7(2):87--101.

Chapter 6. Modified Bare Bones Particle Swarms for Large Scale Optimization

[7] Yang, S. and Sato, Y. (2016). Modified Bare Bones with Differential Evolution for Large Scale Optimization. In *IEEE World Congress on Computational Intelligence (WCCI)*, pages 2760--2767. IEEE.

Chapter 7. Dynamic Heterogeneous Particle Swarm Optimization

[2] Yang, S. and Sato, Y. (2017). Dynamic heterogeneous particle swarm optimization. *IEICE Transactions on Information and Systems*, 100(2):247--255.

Chapter 8. Conclusion and Future Work

Appendix B

Definitions of the Benchmark Functions

Function Definitions

All of test benchmark functions are defined as following:

D: number of variables

j: j-th variable, $j \in [1..D]$

x: the position vector, $x = (x_1, x_2, ..., x_D)$

$f(x)$: the objective function

Rosenbrock function

$$f(x) = \sum_{j=1}^{D} (100 * (x_{j+1} - x_j^2)^2 + (x_j - 1)^2)$$

$x \in [-100, 100]^D$, Global optimum $x^* = (0, 0, ..., 0)$, $f(x^*) = 0$

Matlab code of the Rosenbrock function

```
function fitness=rosenbrock(chrom)
[Row,Dim]=size(chrom);
a=100;
```

```
P1=chrom(1:Row,1:Dim-1);
P2=chrom(1:Row,2:Dim);
fitness = sum((a*(P2-P1.^2).^2+(1-P1).^2),2);
end
```

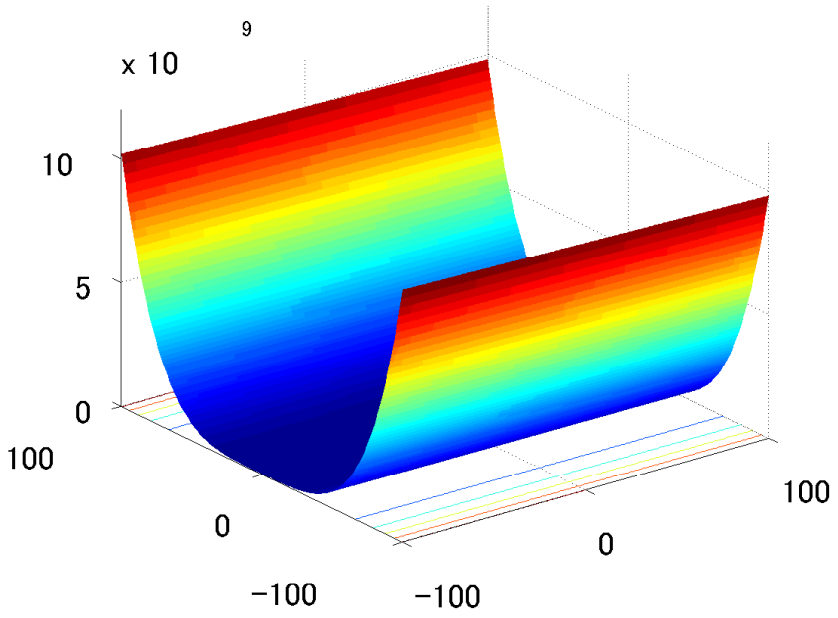


Fig. B.1 3-D map for 2-d Rosenbrock function

Rastrigin function

$$f(x) = \sum_{j=1}^{D}(x_j^2 - 10 * \cos(2\pi x_j)) + 10 * D$$

$x \in [-5.12, 5.12]^D$, Global optimum $x^* = (0, 0, ..., 0)$, $f(x^*) = 0$

Matlab code of the Rastrigin function

```
function fitness=rastrigin(chrom)
Dim=size(chrom,2);
a=10;
```

```
theta=2*pi;
fit=sum((chrom.^2-a*cos(theta*chrom)),2);
fitness=fit+a*Dim;
```

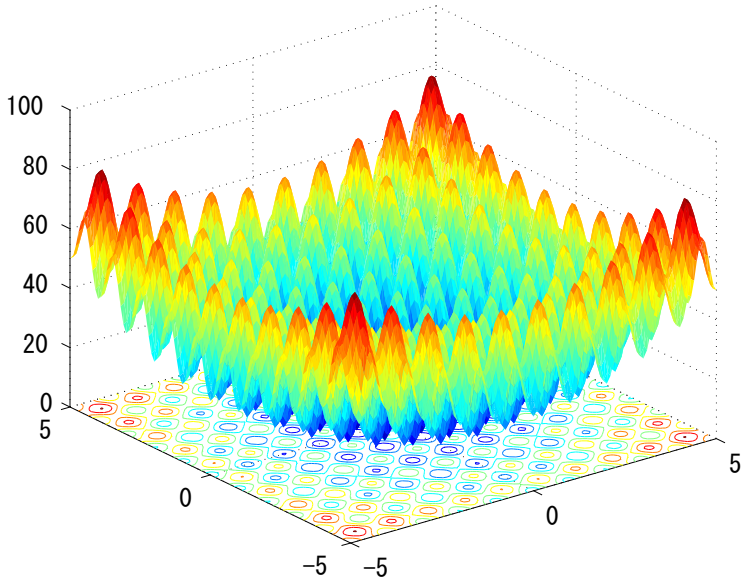


Fig. B.2 3-D map for 2-d Rastrigin function

Griewank function

$$f(x) = \frac{1}{4000}\sum_{j=1}^{D} x_j^2 - \prod_{j=1}^{D} \cos(\frac{x_j}{\sqrt{j}}) + 1$$

$x \in [-5, 10]^D$, Global optimum $x^* = (0, 0, ..., 0)$, $f(x^*) = 0$

Matlab code of the Griewank function

```
function    fitness=griewank(chrom)
[Row,  Dim]=size(chrom);
a=4000.^(-1);
b=repmat(1:Dim,Row,1);
```

```
fit1=sum((a*(chrom).^2),2);
fit2=prod((cos(chrom./sqrt(b))),2);
fitness=fit1-fit2+1;
end
```

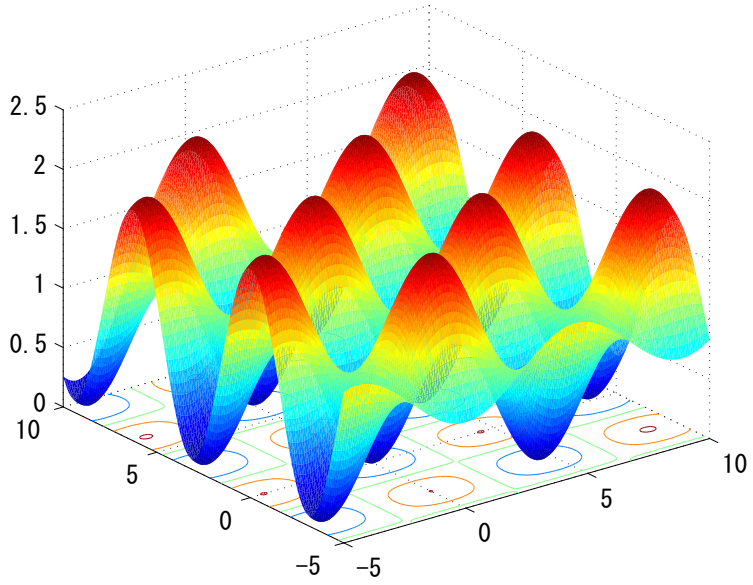


Fig. B.3 3-D map for 2-d Griewank function

Ackley function

$$f(x) = -a * \exp(-0.02 * (D^{-1} \sum_{i=1}^{D-1} x_i^2)^{\frac{1}{2}}) - \exp(D^{-1} \sum_{j=1}^{D} \cos(2\pi x_j)) + a + \exp,\ a = 20$$

$x \in [-15, 30]^D$, Global optimum $x^* = (0, 0, ..., 0)$, $f(x^*) = 0$

Matlab code of the Ackley function

```
function fitness=ackley(chrom)
```

```
Dim=size(chrom,2);
a=-20;
b=exp(1);
c=-0.2;
theta=2*pi;
fit1=sum(chrom.^2,2);
fit2=sum(cos(theta*chrom),2);
fitness=a*exp(c*sqrt(Dim.^(-1)*fit1))-exp(Dim.^(-1)*fit2)-a+b;
end
```

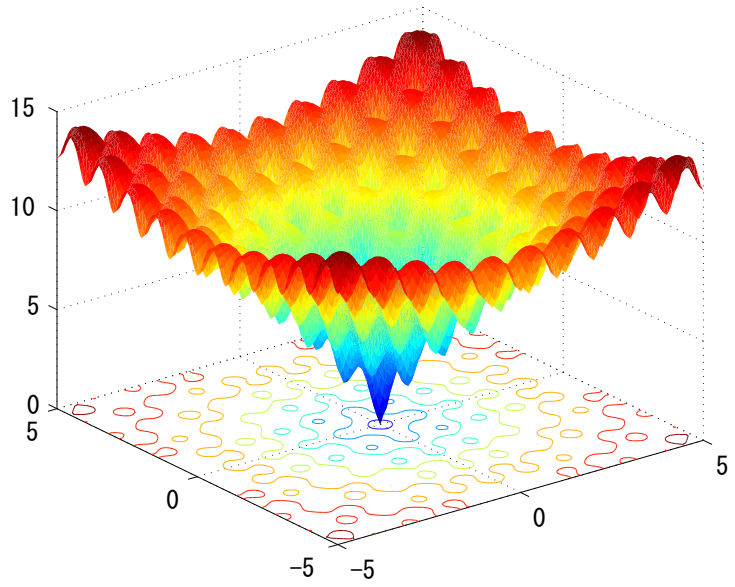


Fig. B.4 3-D map for 2-d Ackley function

Michalewicz function

$$f(x) = -\sum_{j=1}^{D} sin(x_j)[sin(\frac{j \cdot x_j^2}{\pi})]^{20}$$

$x \in [0, \pi]^2$, Global optimum $x^* = (2.20, 1.57)$, $f(x^*) = -1.8013$

Matlab code of the Michalewicz function

```
function fitness = michalewicz(x)
%
% Michalewicz function
% The default value of n =2.
%
n = 2;
m = 10;
s = 0;
for i = 1:n;
    s = s+sin(x(i))*(sin(i*x(i)^2/pi))^(2*m);
end
fitness = -s;
```

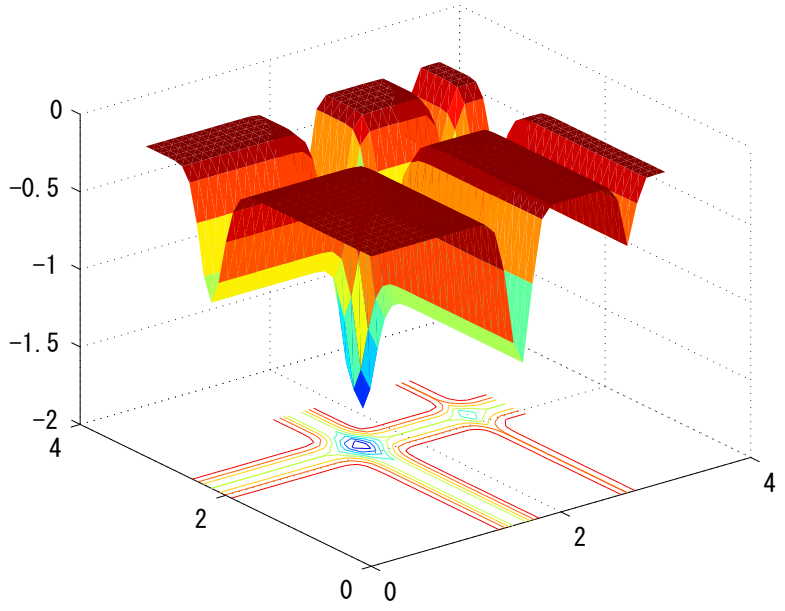


Fig. B.5 3-D map for 2-d Michalewicz function

Levy function

$$f(x) = \sum_{j=1}^{D-1} (\varpi_j - 1)^2[1 + 10sin^2(\pi\varpi_j + 1)] + sin^2(\pi\varpi_1) + (\varpi_D - 1)^2[1 + sin^2(2\pi\varpi_D)]$$

$x \in [-10, 10]^{30}$, Global optimum $x^* = (1, 1, ..., 1)$, $f(x^*) = 0$

Matlab code of the Levy function

```
function fitness = levy(x)
% Levy function
% The default value of n =2.
n = length(x);
for i = 1:n; z(i) = 1+(x(i)-1)/4; end
s = sin(pi*z(1))^2;
for i = 1:n-1
    s = s+(z(i)-1)^2*(1+10*(sin(pi*z(i)+1))^2);
end
fitness = s+(z(n)-1)^2*(1+(sin(2*pi*z(n)))^2);
```

Branin function

$$f(x) = a(x_2 - bx_1^2 + cx_1 - 6)^2 + g(1 - h)\cos(x_1) + 10,$$

$a = 1, b = 1.25\pi^{-2}, c = 5\pi^{-1}, g = 10, h = 0.125\pi^{-1}, x_1 \in [-5, 10], x_2 \in [0, 15]$

Global optimum $x^* = (-\pi, 12.275), (\pi, 2.275) and (9.42478, 2.475)$, $f(x^*) = 0$

Matlab code of the Branin function

```
function fitness = branin(x)
% Branin function
% The number of variables n = 2.
fitness = (x(2)-(5.1/(4*pi^2))*x(1)^2+5*x(1)/pi-6)^2
```

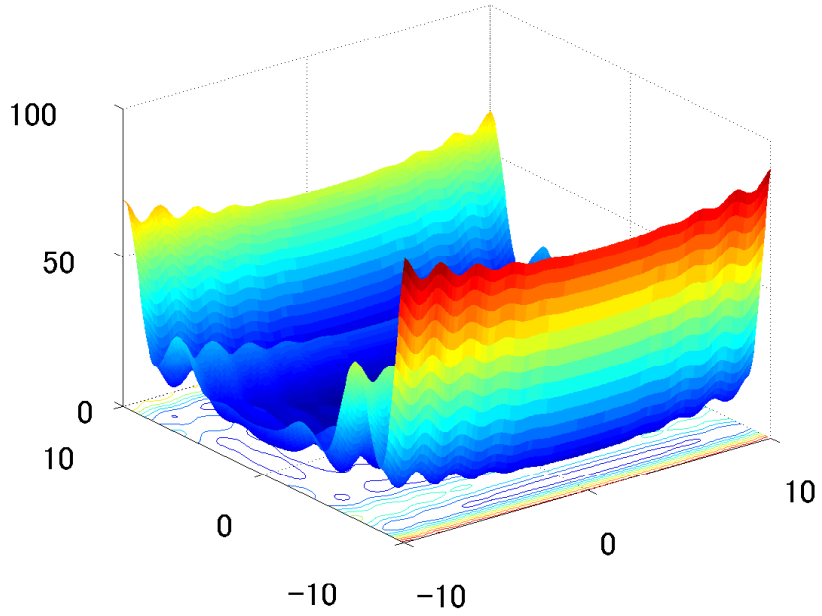


Fig. B.6 3-D map for 2-d Levy function

```
+10*(1-1/(8*pi))*cos(x(1))+10;
```

Shekel function

$$f(x) = -\sum_{i=1}^{m}(\sum_{k=1}^{4}(x_k - C_{ki})^2 + \beta_i)^{-1}$$

$$m = 10, \beta = \frac{1}{10}(1,2,2,4,4,6,3,7,5,5)^T$$

$$C = \begin{pmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3 \\ 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3 \end{pmatrix}$$

$x \in [0,10]^4$, Global optimum $x^* = (4,4,4,4)$, $f(x^*) = -10.5364$

Matlab code of the Shekel function

```
function fitness = shekel(x)
```

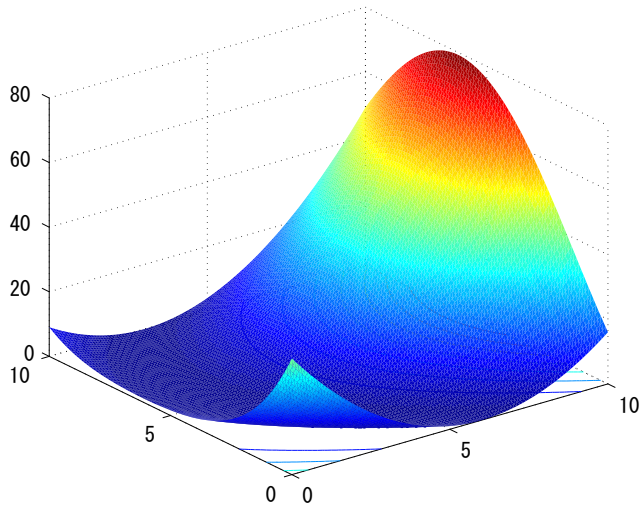


Fig. B.7 3-D map for 2-d Branin function

```
% Shekel function
% The number of variables n = 4
% The parameter m should be adjusted m = 5,7,10.
% The default value of m = 10.
%
m = 10;
a = ones(10,4);
a(1,:) = 4.0*a(1,:);
a(2,:) = 1.0*a(2,:);
a(3,:) = 8.0*a(3,:);
a(4,:) = 6.0*a(4,:);
for j = 1:2;
    a(5,2*j-1) = 3.0; a(5,2*j) = 7.0;
    a(6,2*j-1) = 2.0; a(6,2*j) = 9.0;
    a(7,j)     = 5.0; a(7,j+2) = 3.0;
```

```
    a(8,2*j-1) = 8.0; a(8,2*j) = 1.0;
    a(9,2*j-1) = 6.0; a(9,2*j) = 2.0;
    a(10,2*j-1)= 7.0; a(10,2*j)= 3.6;
end
c(1) = 0.1; c(2) = 0.2; c(3) = 0.2; c(4) = 0.4; c(5) = 0.4;
c(6) = 0.6; c(7) = 0.3; c(8) = 0.7; c(9) = 0.5; c(10)= 0.5;
s = 0;
for j = 1:m;
    p = 0;
    for i = 1:4
        p = p+(x(i)-a(j,i))^2;
    end
    s = s+1/(p+c(j));
end
fitness = -s;
```

Shaffers N.2 function

$$f(x) = 0.5 + \frac{sin^2(x^2 - y^2) - 0.5}{(1 + 0.001 * (x^2 + y^2))^2}$$

$x \in [-100, 100]^2$, Global optimum $x^* = (0, 0)$, $f(x^*) = 0$

Matlab code of the Shaffers N.2 function

```
function fitness = shaffers(x)
% Shaffers N.2 function
% The number of variables n=2
a1=sqrt(x(1)^2+x(2)^2);
a=sin(a1);
```

```
b=((x(1)^2+x(2)^2)*0.001+1)^2;
fitness=0.5+(a^2-0.5)/b;
```

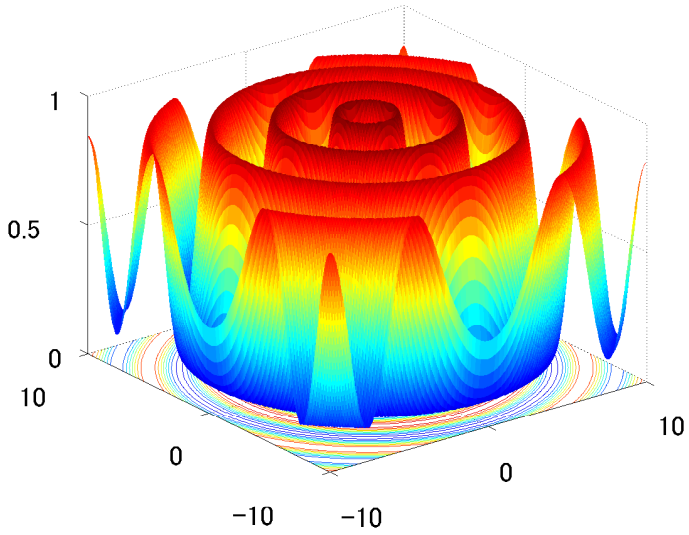


Fig. B.8 3-D map for 2-d Shaffers N.2 function

Schwefel function

$$f(x) = 418.9829 * D - \sum_{j=1}^{D} x_j sin(\sqrt{|x_j|})$$

$x \in [-500, 500]^D$, Global optimum $x^* = (420.9687, 420.9687, ..., 420.9687)$, $f(x^*) = 0$

Matlab code of the Schwefel function

```
function    fitness=schwefel(chrom)
[Row,  Dim]=size(chrom);
a=418.9829*Dim;
b=chrom.*sin(sqrt(abs(chrom)));

fitness=a-sum(b')';
```

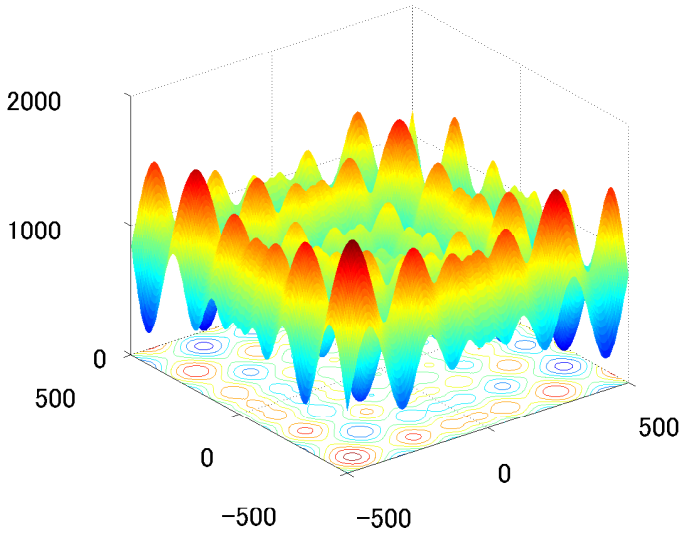


Fig. B.9 3-D map for 2-d Schwefel function

Rotated hyper-ellipsoid function

$$f(x) = \sum_{j=1}^{D} \sum_{k=1}^{j} x_k^2$$

$x \in [-65.536, 65.536]^D$, Global optimum $x^* = (0, 0, ..., 0)$, $f(x^*) = 0$

Matlab code of the Rotated hyper-ellipsoid function

```
function   fitness=rothyper(chrom)
% Rotated hyper-ellipsoid function
[Row, Dim]=size(chrom);
new_chrom=zeros(Row,Dim);
chrom=chrom.^2;
new_chrom(:,1)=chrom(:,1);
for i=2:Dim
    new_chrom(:,i)=sum(chrom(:,1:i),2);
end
```

```
fitness=sum(new_chrom,2);
end
```

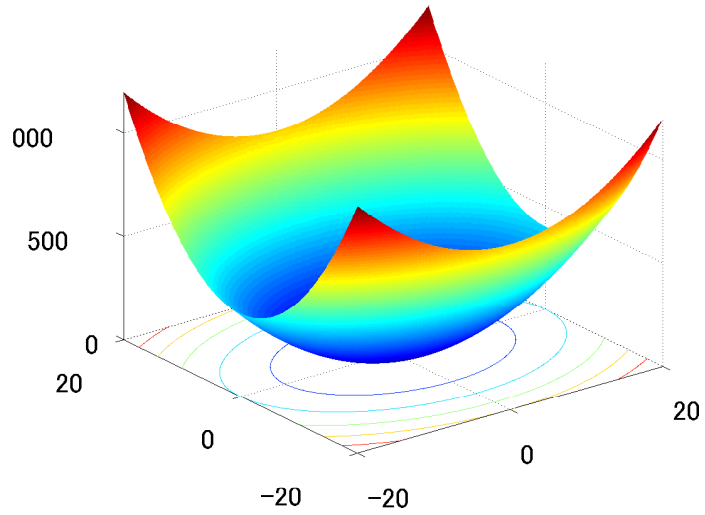


Fig. B.10 3-D map for 2-d Rotated hyper-ellipsoid function

Sphere function

$$f(x) = \sum_{j=1}^{D} x_i^2$$

$x \in [-100, 100]^D$, Global optimum $x^* = (0, 0, ..., 0)$, $f(x^*) = 0$

Matlab code of the Sphere function

```
function fitness=Sphere(Chrom)
Colony=Chrom.^2;
fitness=sum(Colony,2);
end
```

Zakharov function

$$f(x) = \sum_{j=1}^{D} x_j^2 + (\sum_{j=1}^{D} 0.5jx_j)^2 + (\sum_{j=1}^{D} 0.5jx_j)^4$$

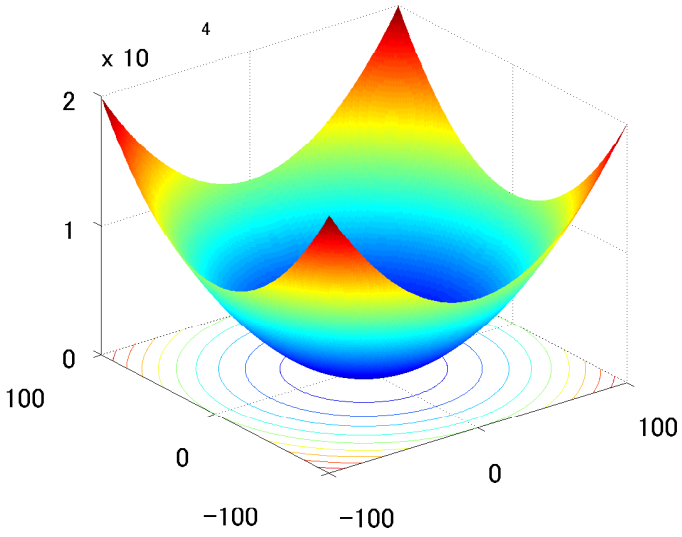


Fig. B.11 3-D map for 2-d Sphere function

$x \in [-1, 1]^D$, Global optimum $x^* = (0, 0, ..., 0)$, $f(x^*) = 0$

Matlab code of the Zakharov function

```
function fitness = zakharov(xx)
d = length(xx);
sum1 = 0;
sum2 = 0;
for ii = 1:d
        xi = xx(ii);
        sum1 = sum1 + xi^2;
        sum2 = sum2 + 0.5*ii*xi;
end
fitness = sum1 + sum2^2 + sum2^4;
```

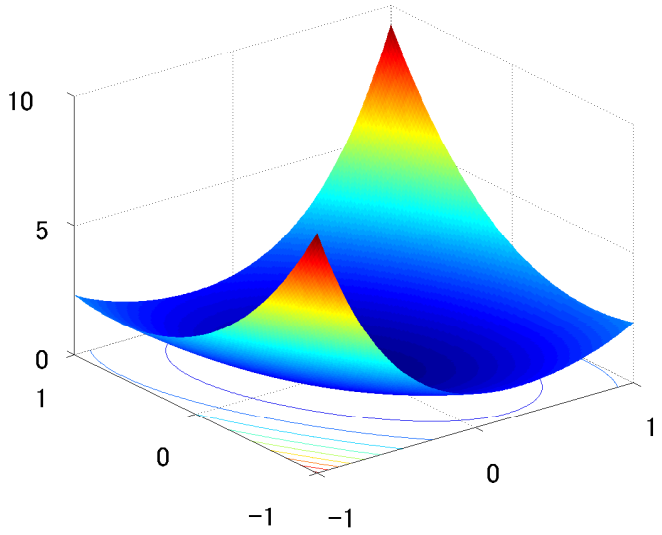


Fig. B.12 3-D map for 2-d Zakharov function

Sum of different powers function

$$f(x) = \sum_{j=1}^{D} |x_j|^{j+1}$$

$x \in [-1, 1]^D$, Global optimum $x^* = (0, 0, ..., 0)$, $f(x^*) = 0$

Matlab code of the Sum of different powers function

```
function fitness = sumpow(chrom)
% Sum of different powers function
d = length(chrom);
sum = 0;
for ii = 1:d
    xi = chrom(ii);
    new = (abs(xi))^(ii+1);
    sum = sum + new;
end
fitness = sum;
```

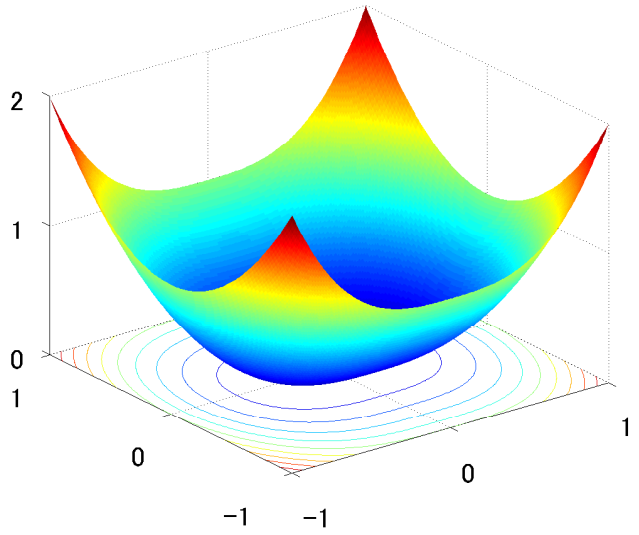


Fig. B.13 3-D map for 2-d Sum of different powers function