Search Dynamics Analysis and Its Application in Evolutionary Algorithm

by

Xiaosi Li

A dissertation

submitted to the Graduate School of Science and Engineering for Education

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Engineering

University of Toyama

Gofuku 3190, Toyama-shi, Toyama 930-8555 Japan

2020

(Submitted June 9, 2020)

Acknowledgements

I would like to dedicate my paper to all those who have offered me tremendous assistance during the past three years in Tang Lab,University of Toyama. First and foremost, my sincerely thanks go to Prof. Zheng Tang at University of Toyama, my supervisor,who with incomparable patience to help and encouraged me profoundly throughout my academic study.Also I owe many thanks to Prof. Shangce Gao, who has offered me valuable suggestions in the academic studies. In the preparation of this thesis, he has spent much time in reading my draft and provided me with inspiring advice. Without his consistent and illuminating instruction, this thesis could not have reached its present form. Besides, special thanks should go to to all the professors who have taught me during my previous study in this university, such as Prof. Fan who is famous in catalyst field and with his course help me eye-opening and gaining more understanding to engineer is fun spirit. The profit that I gained from them will be of everlasting significance to my future research. Finally, my thanks would go to my beloved family for their loving considerations and great confidence,especially for Axom,my younger son,who always suggest me to build a powerful AI robot which can help me to finish my paper.

Abstract

More and more optimization algorithms have been proposed to solve complex optimization problems in recent years. Most of them have complex dynamic systems including many variable parameters and some features such as rotation characteristics, high dimension, dynamical and hard to measure, and even difficult to make a mathematical model. It is a challenging work for researchers to pay attention to figure out more effective and efficient optimization algorithm. Nature-inspired meta-heuristic (NMH) algorithms are well-known for promising results in combinatorial optimization. Evolutionary algorithms (EAs) as population-based computational intelligence are widely applied to various optimization problems, which show tremendous potential and make great effort in the field of optimization. However, optimization problems are becoming more and more complex, thus a traditional algorithm which has single search mechanism generally suffers from weak search ability or slow convergence speed, which leads to unsatisfying results. In these scenarios, hybridization of algorithms is a powerful method. Every algorithm has its inherent pros and cons. Researchers have been trying to improve their search patterns and search styles in order to figure out their weaknesses or limitations and then combine them with other effective methods. However, the search mode has not been researched deeply. In this paper, we find that search style and individual selection mechanism for interaction are the core problems for a meta-heuristic algorithm. In particular, we focus on search style and have studied the principle of basic search style. Spherical evolution (SE) a spherical search style. Experiments have demonstrated that it is a powerful optimiza-

tion tool for function optimization and real-world problems. However, there are some limitations in SE. SE is sensitive to some initial parameters and the angle of spherical search style influences its search behavior. Thus, although it is a promising algorithm, its performance is limited by its single spherical search. Hypercube search (HS) style has been widely used in various NMH algorithms. By the first order difference, individuals gradually search towards global optimum. In addition, chaos is a common natural phenomenon. Chaotic local search (CLS) makes use of its randomicity and ergodicity. By combining CLS, many traditional NMH algorithms get significant improvement in their performances, such as chaotic BSO, DE combined with CLS, and chaotic GSA. They demonstrate that CLS can greatly enhance their search ability and avoid trapping into premature. Backtracking search optimization algorithm (BSA) ,BSA is similar to evolutionary algorithm, including selection, crossover, mutation and operations. The core idea of BSA is to get guidance from the previous population and search solutions with better fitness. While BSA only evaluates the individuals based on the fitness, which could make the individuals with better exploration prospects be discarded when updating population. The Negatively Correlated Search (NCS) is an improved stochastic optimization inspired by human cooperation. NCS applies a parallel search pattern of multi-agent for modeling based on probability distribution to promote the diversity of search behaviors in an information sharing and cooperation way, which establishes a novel approach for cooperation among individuals. NCS is constituted of multiple search processes that develops into a new individual with the influence of the search results from itself and others. Generally, the relationship among search processes is negatively correlated, which means that the framework of the algorithm tends to form a situation where each search process is distributed in different regions and avoid duplication to prevention of local optimum. Therefore, base on these, I propose several improved search style and verified on thirty CEC2017

benchmark functions and real-world optimization problems . These algorithms are introduced as follows. (1) TDSD: A New Evolutionary Algorithm Based on Triple Distinct Search Dynamics, for the first time we propose a novel algorithm which contains triple distinct search dynamics (TDSD), i.e., SE, HS and CLS. We take advantage of them to implement more sufficient and effective search. Control strategies among three kinds of search styles are conducted to provide a good balance between exploration and exploitation phases. Thus, the performance of TDSD is satisfying. Thirty CEC2017 benchmark functions and three real-world optimization problems are used to evaluate TDSD. Experimental results demonstrate that TDSD has the optimization potential.. (2) NC-GSA: an improved gravitational search algorithm based on negative correlation is proposed with the hypothesis that hybridization of algorithms improves optimization performances and efficiency. For the ease of consistent description, the novel GSA based on negative correlation learning is abbreviated to NC-GSA. As population-based search methods, GSA and NCS exchange information among individuals through fitness and search area, respectively. While GSA conducts exploitation in the search space, NCS fulfills exploration by encouraging discrepant search behaviors to increase the optimization accuracy. By doing so, NC-GSA is expected to well balance the exploitation and exploration, thus possessing a significant better or competitive performance in comparison with its component algorithms. Experimental results based on thirty benchmark optimization functions demonstrate the robustness of NC-GSA, and also indicate that NC-GSA performs better than GSA and NCS in terms of solution quality and convergence speed. Additionally, a contrast experiment is also conducted to verify the influence of the implementation sequence of component algorithms on the performance, and the results suggest that NCS should be implemented prior to GSA, and the reason seems to be that NCS diversifies the distribution of solutions, upon which GSA utilizes the gravitational search to obtain a

fast convergence. (3) BSANCS: backtracking search optimization algorithm (BSA)is new evolutionary algorithm (EA) that to solve global optimization problems. BSA is similar to evolutionary algorithm, including selection, crossover, mutation and operations. The core idea of BSA is to get guidance from the previous population and search solutions with better fitness. While BSA only evaluates the individuals based on the fitness, which could make the individuals with better exploration prospects be discarded when updating population. Thus, in this paper, we proposed a novel way to improve exploration abilities that utilize negative correlation learning enhanced search behavior in BSA to further improve its search ability. We used benchmark function suit CEC2017 to verify the proposed new algorithm and the experiment result indicates the feasibility of this hybridization. The thesis is organized as follows: Chapter 1 gives a brief introduction about the Search Dynamics. Chapter 2 briefly introduces the Search Pattern and Search Style. Chapter 3 introduces Triple Distinct Search Dynamics (TDSD). In Chapter 4, an improved gravitational search algorithm based on negative correlation (NC-GSA) is introduced. The in Chapter 5, negative correlation learning enhanced search behavior in BSA is introduced (BSANCS). Finally, Chapter 6 general conclusions of this thesis and some possible research trends also been pointed out.

Contents

Acknowledgements **ii**

Abstract **iii**

1 Introduction **1**

1.1 Evolutionary Computation 1

1.2 Evolutionary Algorithms 2

1.3 Search Dynamics 3

2 Search Pattern and Search Style **5**

2.1 TDSD Search 6

2.2 Negatively Correlated Search 6

2.3 Backtracking Search 7

3 Triple Distinct Search Dynamics (TDSD) **10**

3.1 Introduction 10

3.2 Spherical evolution 13

3.3 New algorithm TDSD 17

3.3.1 Brief introduction of CLS 17

3.3.2 Principle of TDSD 18

3.4 Experiments 22

3.4.1 Experimental setup 22

3.4.2 Experimental results 23
3.4.3 Analysis of Parameters 28
3.4.4 Three real-world optimization problems 29
3.4.5 Comparison with champion algorithms 30
3.5 Conclusion 31

4 An improved gravitational search algorithm based on negative correlation(NC-GSA) 38

4.1 Introduction 38
4.2 Negative Correlation Gravitational Search Algorithm 42
4.3 Experimental evaluation 48
4.4 Conclusions 51

5 Negative correlation learning enhanced search behavior in BSA(BSANCS) 53

5.1 Introduction 53
5.2 Hybrid BSANCS 55
5.3 Experimental Results 59
5.4 Conclusion 61

6 Conclusions and Remarks 65

Bibliography 67

List of Figures

3.1 Illustration of hypercube and spherical search styles in two-dimension space. 15

3.2 The flowchart of TDSD. 20

3.3 The box-and-whisker diagrams of optimal solutions obtained by seven algorithms on F1, F4, F17 and F26. 26

3.4 The convergence graphs of average best-so-far solutions obtained by seven algorithms on F1, F4, F17 and F26. 26

3.5 The search graphs of TDSD on F10 with 2 dimensions where indicates the best-so-far solution. 28

4.1 The gravitational phenomena used in GSA. 40

4.2 Generic architectures of hybrid algorithms. 42

4.3 Flowchart of (a) NC-GSA and (b) GSA-NC. 44

4.4 Box-and-whisker plot of the obtained 30 final solutions by all compared algorithms for the benchmark function F30. 50

4.5 Convergence graph of all compared algorithms for the benchmark function F30. 50

5.1 Convergence graph of F8. 60

5.2 Convergence graph of F16. 60

5.3 Convergence graph of F7. 61

5.4 Convergence graph of F28. 61

5.5 Box-and-whisker graph of F8. 62
5.6 Box-and-whisker graph of F16. 62
5.7 Box-and-whisker graph of F7. 62
5.8 Box-and-whisker graph of F28. 64

List of Tables

3.1 Brief summary of meta-heuristic hybridization and their applications. . 12

3.2 Parameter settings of TDSD and other algorithms. 23

3.3 Experimental results of TDSD versus three NMH algorithms on CEC2017 benchmark functions (1). 24

3.4 Experimental results of TDSD versus three NMH algorithms on CEC2017 benchmark functions (2). 25

3.5 Experimental results of different $nCLS$. 32

3.6 Experimental results of different settings of F_0 and σ_0. 33

3.7 Statistical results of different settings of F_0 and σ_0. 34

3.8 Experimental results of eight algorithms on FMSWP. 35

3.9 Experimental results of eight algorithms on SSRPPCD. 35

3.10 Experimental results of eight algorithms on NLSTRP. 35

3.11 Experimental results of TDSD versus four champion algorithms on CEC2017 benchmark functions. 36

3.12 Experimental results of five algorithms on FMSWP. 37

3.13 Experimental results of five algorithms on SSRPPCD. 37

3.14 Experimental results of five algorithms on NLSTRP. 37

4.1 The objective evaluations of different methods 48

4.2 Results obtained by the Friedman and Wilcoxon tests for the control algorithm NC-GSA. 51

5.1 Experiment results of BSANCS and BSA on CEC2017. 63

5.2 Results obtained by the Wilcoxon test. 64

Chapter 1

Introduction

1.1 Evolutionary Computation

In the field of computational intelligence, evolutionary computing is a family of global optimization algorithms inspired by biological evolution. It is also a subfield of artificial intelligence and soft computing that studies these algorithms. Technically, they are a series of population-based trial and error solvers, which with meta-heuristic or random optimization characteristics..

In evolutionary computation, the general process is that an initial set of candidate solutions will be generated and updated iteratively. Then a new generation of products was created by randomly deleting less desired ideal solutions and introducing small random changes. In biological terms, a group of populations undergoes natural selection (or artificial selection) and mutation. As a result, the population will gradually evolve to increase the fitness. In this case, the fitness function of the algorithm is selected.

Evolutionary computation technology can produce highly optimized solutions to various problems, making them very popular in computer science. Because of there are many variations and extensions, they are suitable for more specific questions and data structure series. Evolutionary computing is sometimes used as a computer simulation experiment program for evolutionary biology to study common aspects of

general evolutionary processes.

1.2 Evolutionary Algorithms

Evolutionary algorithms constitute a subset of evolutionary computing because they usually only involve technologies that implement mechanisms inspired by biological evolution, such as replication, mutation, reorganization, natural selection, and survival of the fittest. Candidate solutions to optimization problems play an individual role in the population, and the cost function determines the environment in which the solution will "survive" (see also fitness function). Then, after repeatedly applying the above operators, the population will evolve.

In this process, there are two main forces that form the basis of the evolutionary system: recombination mutations and crossovers create the necessary diversity, thereby promoting novelty, and choice becomes a major means of improving quality.

Many aspects of this evolutionary process are random. On the one hand, the information changed by reorganization and mutation is randomly selected. On the other hand, the selection operator can be deterministic or random. In the latter case, individuals with higher fitness have a higher chance of being selected than individuals with lower fitness, but usually even individuals with low fitness have the opportunity to become parents or survive.

A good balance between exploration and exploitation is the key to a evolutionary algorithms. However,the search mode has not been researched deeply. In this paper,we deeply study the search dynamics, and find that search style and individual selection mechanism for interaction are the core problems for a meta-heuristic algorithm.

1.3 Search Dynamics

In these years, Nature-inspired meta-heuristic (NMH) algorithms are more and more well-known for promising results in combinatorial optimization. Evolutionary algorithms (EAs) as population-based computational intelligence are widely applied to various optimization problems, which show tremendous potential and make great effort in the field of optimization.However, optimization problems are becoming more and more complex, thus a traditional algorithm which has single search mechanism generally suffers from weak search ability or slow convergence speed, which leads to unsatisfying results. So it is necessary to study search dynamics to find what is the key aspect to improve the search ability in evolutionary algorithms. Optimization usually aims to find the best point in the search space, that is, finding the global minimum of objective function. However, the objective function may be non-linear, complex or non-differentiable. When the objective function is non-linear, complex or non-differentiable, evolutionary algorithm (EA) [1, 2] is usually used to solve the global optimal problem. EA is a mature global optimization method with high robustness and wide applicability. EAs can solve different types of problems. Therefore, EAs have be used by various industries to solve the global optimum problem [3, 4]. In the family of EAs, many algorithms exhibit great search abilities. For example, the particle swarm optimization algorithm (PSO) [5] simulates the social behavior of creatures, such as bird flock or fish schooling. The ant colony optimization algorithm (ACO) [6] is stochastic search algorithm that simulates the process of natural ants seeking source of food. The artificial bee colony algorithm (ABC) [7–9] is an optimization algorithm that simulates the foraging behavior of honey bees. The genetic algorithm (GA) [10] is a meta-heuristic algorithm inspired by the process of natural selection strategy. The differential evolution algorithm (DE) [11–14] is a method to optimize problems by alliteratively trying to improve a candidate solution with regard

to a given measure of quality. Among these kind of EAs,which can categorized by their different search pattern and search style,which will discuss later.

Chapter 2

Search Pattern and Search Style

The computational model of evolutionary computation is used as a key element in solving problems. Many evolutionary computing models have been proposed. We call them evolutionary algorithms. They share a common conceptual basis which can simulate the evolution of natural animals through nature selection and regeneration processes. The first evolutionary algorithms proposed can be went back to 1950s years (e.g., Fraser, 1957; Box, 1957). To More simplicity, we won't focus on the earlier work. However,we normally will talk about more detail the following three methodologies which have been proposed in the past years: "evolutionary programming" (et al Fogel. (1966), "evolutionary strategy" (Rechenberg, 1973) and "genetic algorithm" (Holland, 1975).Although each of these variants is implemented in a different way. Evolutionary algorithms have been applied in many applications. The examples include data mining, pattern recognition, and complex functions. In this case, many functions have been added to EA, which can improve the quality of the solution,as a result the EAs look very different from one to another,however they can be divided into different search mode. As for the computational search mode,which is majorly about its search pattern and search style. Search pattern indicates individuals' search mechanism. It guides how individuals search. Search style denotes individuals' search operator, implying individuals' evolutionary method. To be specific, search pattern defines which search style is used. Also, it can contain various search styles.In addi-

tion, search pattern it gives the canonical form of individuals' search.The following is three major search mode we have studied.

2.1 TDSD Search

TDSD is a self-adaptive search algorithm. It can adaptively adjust search strategies according to current search situation. SE has a large search range, hence it can extensively search the space and avoid trapping into local optima. HS is a directional search. Based on new found solutions, it can guide individuals towards more promising area. Thus, it improves the exploration ability. CLS implements local search to enhance the exploitation ability. TDSD adaptively switches these three search strategies to reinforce its overall search ability. In six NMH algorithms, individuals only have one kind of search strategy. In other words, they have the same search behaviors in each algorithm. However, in TDSD, each individual executes different search strategies according to own search situation. Their search diversity is remarkably improved. Consequently, they can find better solutions than six NMH algorithms on numerous functions.

2.2 Negatively Correlated Search

the negatively correlated search (NCS) [15] is an improved stochastic optimization inspired by human cooperation. NCS applies a parallel search pattern of multi-agent for modeling based on probability distribution to promote the diversity of search behaviors in an information sharing and cooperation way, which establishes a novel approach for cooperation among individuals. NCS is constituted of multiple search processes that develops into a new individual with the influence of the search results from itself and others. Generally, the relationship among search processes is negatively correlated, which means that the framework of the algorithm tends to form a situation

where each search process is distributed in different regions and avoid duplication to prevention of local optimum.

NCS is highlighted by the consideration of both the fitness of individuals as quality measurement and individual differences. In a steep solution space, the global optimal solution where an individual with poor fitness comprising a piece of excellent gene, in all probability, might be missed or eliminated in the early stage. As a solution for preserving misinterpreted solution, NCS selects candidate solutions according to search behavior (probability distribution of new individuals) rather than distance between individuals. Based on the information of population distribution and transformation of population distribution, NCS leads to sub population division to simplify the solution space of the original problem through information transmission among individuals. Compared with the traditional evolutionary algorithm, NCS has the advantages of high efficiency and the characteristics of robustness. The scaling factor A is positive control group evolutionary rate selection. Although there is no upper limit on the A, the RMS value is less than "1", and is a random number between 0 and 1.

2.3 Backtracking Search

Backtracking search optimization algorithm (BSA) [16] is a new EA that to solve global optimization problems. BSA is based on basic genetic rules. Therefore,it includes five processes: initialization, selection-I, mutation, crossover and selection-II. In initialization process, the population (pop) is randomly generated in the search space. In selection-I stage, BSA possesses a memory to store historical population ($pop_{historical}$) that to be used for calculating the search direction. In mutation process,

BSA generates a mutant population ($pop_{mutation}$) using Eq. (5.1):

$$pop_{mutation} = pop + F \cdot (pop_{historical} - pop), \tag{2.1}$$

where F is a parameter that controls the amplitude of the search direction matrix ($pop_{historical}$). The F is a standard normal distribution, which is formulated as $F = 3 \cdot randn$.

Then, in crossover stage, trial population ($offsprings$) is generated by crossover process using Eq. (5.2):

$$offsprings = pop + (map \cdot F) \cdot (pop_{historical} - pop), \tag{2.2}$$

where map is binary integer value matrix of size $Popsize \cdot Dim$ ($Popsize$ is size of population. Dim is population dimension) that guides crossover operation. Finally, in selection-II process, population (pop) is updated by trial population ($offsprings$) with the better fitness value. If the fitness value of the best population (P_{best}) is better than the currently global minimum value, the global minimum population is updated to be P_{best}, the global minimum value is updated to the fitness value of P_{best}.

Due to BSA only evaluates the individuals based on the fitness, individuals with promising exploration prospects may be discarded when updating population. Therefore, we should take these individuals into account when executing selection operation. Now, many ways of optimizing algorithms are made by hybridizing two or more algorithms [17–20]. Inspired by this, we combine BSA with negatively correlated search (NCS) [15] to enhance search behavior backtracking search mechanism. We call this hybridized algorithm BSANCS. The NCS models the search behavior of individual search process as a probability distribution, and calculates the correlation among different individual via a variable called Bhattacharyya distance. The NCS maintains search behaviors by encouraging differences among the probability distributions. The

main property of NCS is that it selects individual with better fitness value and larger probability distributions into the next generation. NCS can population and promote negatively correlated search by sharing information to improve exploration abilities of BSA.

Chapter 3

Triple Distinct Search Dynamics (TDSD)

3.1 Introduction

More and more optimization algorithms have been proposed to solve complex optimization problems in recent years [21–23]. Most of them have complex dynamic systems including many variable parameters and some features such as rotation characteristics, high dimension, dynamical and hard to measure, and even difficult to make a mathematical model. It is a challenging work for researchers to pay attention to figure out more effective and efficient optimization algorithm.

Nature-inspired meta-heuristic (NMH) algorithms are well-known for promising results in combinatorial optimization. Evolutionary algorithms (EAs) as population-based computational intelligence are widely applied to various optimization problems, which show tremendous potential and make great effort in the field of optimization [24, 25]. However, optimization problems are becoming more and more complex, thus a traditional algorithm which has single search mechanism generally suffers from weak search ability or slow convergence speed, which leads to unsatisfying results. In these scenarios, hybridization of algorithms is a powerful method. Every algorithm has its inherent pros and cons. Researchers have been trying to improve their search patterns and search styles in order to figure out their weaknesses or limitations and

then combine them with other effective methods [26–28].

Differential evolution (DE) [29] is a heuristic random search algorithm. Wang et al. have embedded DE with multi-objective sorting-based mutation operators [30]. Its fitness and diversity information are simultaneously considered for selecting parents. As a result, a good balance between exploration and exploitation can be achieved to greatly enhance the performance of DE. Gravitational search algorithm (GSA) [31] is an adaptive search technology based on Newtonian gravity. In GSA, candidate solutions of a population are modeled as swarm objects. These objects tend to move towards the heaviest object. Yin et al. introduce a hybridization of K-harmonic mean method into GSA to solve clustering problems [32]. Particle swarm optimization (PSO) [5] is well-known for simulating social behaviors of wild animals such as birds' flocking and fishes' schooling. Tian et al. have successfully combined PSO with DE to arrange rescue vehicles to extinguish forest fire problem [33]. Brain storm optimization (BSO) [34] is inspired by the human brainstorming process. CBSO [35] and ASBSO [36] are two kinds of hybridization of BSO with chaotic local search and memory-based selection. At present, there are plenty of researches demonstrating that hybrid methods gain the great success in resolving complex optimization problems. Some of them are listed in Table 3.1.

Table 3.1 has shown diversity, flexibility and effectiveness of meta-heuristic hybridization in their applications. However, to take insight into the mechanism of hybridization, there is no quintessential difference among them. The only difference is search operators. Two main factors can be considered: search pattern and search style. Researchers study advantages and disadvantages of certain algorithm to reinforce its search ability in order to get an excellent hybrid algorithm. From state-of-the-art hybrid algorithms, a spiral structure can help to realize an effective strategy in some scenarios, such as sine cosine algorithm (SCA) [48], artificial algae

Table 3.1: Brief summary of meta-heuristic hybridization and their applications.

NMH hybridization	Applications
GA+SA [37]	Functions
GSA+K-harmonic means [32]	Clustering problems
PSO+DE [33]	Scheduling problems
GA+ACO [38]	Traveling salesman problems
GA+PSO [39]	Traveling salesman problems
SA+TS [40]	Traveling salesman problems
GRASP+PSO [41]	Traveling salesman problems
IA+ACO [42]	Traveling salesman problems
ABC+ACO [43]	Traveling salesman problems
PSO+ACO [44]	Traveling salesman problems
GA+ACO+PSO [45]	Traveling salesman problems
PSO+ACO+3-opt [46]	Traveling salesman problems
GA+SA+ACO+PSO [47]	Traveling salesman problems
BSO+flexible search [36]	Functions and real-world problems
DE+sorting-based mutation [30]	Functions and real-world problems

algorithm (AAA) [49], and so on [50].

Recently, a novel NMH algorithm named spherical evolution (SE) [51] has been proposed for solving continuous optimization problems. It has a spherical search style. Experiments have demonstrated that it is a powerful optimization tool for function optimization and real-world problems. However, there are some limitations in SE. SE is sensitive to some initial parameters and the angle of spherical search style influences its search behavior. Thus, although it is a promising algorithm, its performance is limited by its single spherical search.

Hypercube search (HS) style has been widely used in various NMH algorithms [5, 29, 34]. By the first order difference, individuals gradually search towards global optimum. In addition, chaos is a common natural phenomenon. Chaotic local search (CLS) makes use of its randomicity and ergodicity. By combining CLS, many traditional NMH algorithms get significant improvement in their performances, such as chaotic BSO [35], DE combined with CLS [52, 53], and chaotic GSA [54]. They demonstrate that CLS can greatly enhance their search ability and avoid trapping into premature.

In this paper, for the first time we propose a novel algorithm which contains triple

distinct search dynamics (TDSD), i.e., SE, HS and CLS. We take advantage of them to implement more sufficient and effective search. Control strategies among three kinds of search styles are conducted to provide a good balance between exploration and exploitation phases. Thus, the performance of TDSD is satisfying. Thirty CEC2017 benchmark functions and three real-world optimization problems are used to evaluate TDSD. Experimental results demonstrate that TDSD has the optimization potential.

The main contributions of this paper can be summarized as follows: (1) We make first attempt to well organize SE, HS and CLS. Triple distinct search dynamics greatly improve the search ability of algorithm. (2) We implement good control strategies for balancing these three search mechanisms. (3) Sufficient experiments are conducted to verify the performance of TDSD.

The remainder of this paper is organized as follows. Section 3.2 gives brief introduction of SE. Section 3.3 proposes TDSD in details. Section 3.4 presents simulations and analyses of results. Section 3.5 summarizes the research and gives future work.

3.2 Spherical evolution

Search operators play important roles in NMH algorithms. Researches have pointed out that these NMH algorithms are very different [51]. However, their mechanisms can be summarized as two main characteristics. One is search pattern and the other is search style. Search pattern indicates individuals' search mechanism. It guides how individuals search. Search style denotes individuals' search operator, implying individuals' evolutionary method. To be specific, search pattern defines which search style is used. Also, it can contain various search styles. Thus, search pattern can be represented as Eq. (3.1).

$$X_{i,d}^{new} = X_{\gamma,d} + \sum_{k=1}^{n} S(X_{\alpha,d}^{k}, X_{\beta,d}^{k}), \tag{3.1}$$

where $X_{i,d}^{new}$ indicates the new ith solution in the dth dimension. X_α, X_β and X_γ are three definite solutions selected by a certain strategy. $S()$ represents updating units in the search operator and n is the number of updating units.

Hypercube search is popularly used in many well-known NMH algorithms owing to its simple description and multi-dimension extension, described as Eq. (3.2).

$$S(X_{\alpha,d}^k, X_{\beta,d}^k) = SF_1() \cdot (SF_2() \cdot X_{\alpha,d}^k - SF_3() \cdot X_{\beta,d}^k), \tag{3.2}$$

where $SF_1()$, $SF_2()$ and $SF_3()$ denote three scaling factor functions which tune the scale of difference between $X_{\alpha,d}^k$ and $X_{\beta,d}^k$.

SE is one of recently proposed NMH algorithms [51]. The essential difference of search operators between SE and other algorithms is that SE adopts a spherical search style whereas others use a hypercube search style. When algorithms are limited to two dimensions, SE presents a circular search style whereas others conduct a rectangular one. Fig. 3.1 shows their search styles.

As shown in Fig. 3.1, in two dimensions, a hypercube search trajectory is a rectangle whereas a spherical search explores one circle with center O and radius OF. Dotted lines with black arrow $\vec{SD}$, $\vec{SC}$, $\vec{SE}$ and $\vec{SG}$ denote two solution vectors for hypercube search and spherical search, respectively. In hypercube search, its search area is determined by DA and DB. In spherical search, its angle changes from 0 to 2π. When its radius OF equals to rectangular diagonal DC, it is obvious that the spherical search has larger search space than hypercube search, which means it has better exploration ability. This advantage can help spherical search get more possibility to avoid trapping into local optima. When dimension is increased, spherical search works according to Euclidean distance. Its principle in one, two and high

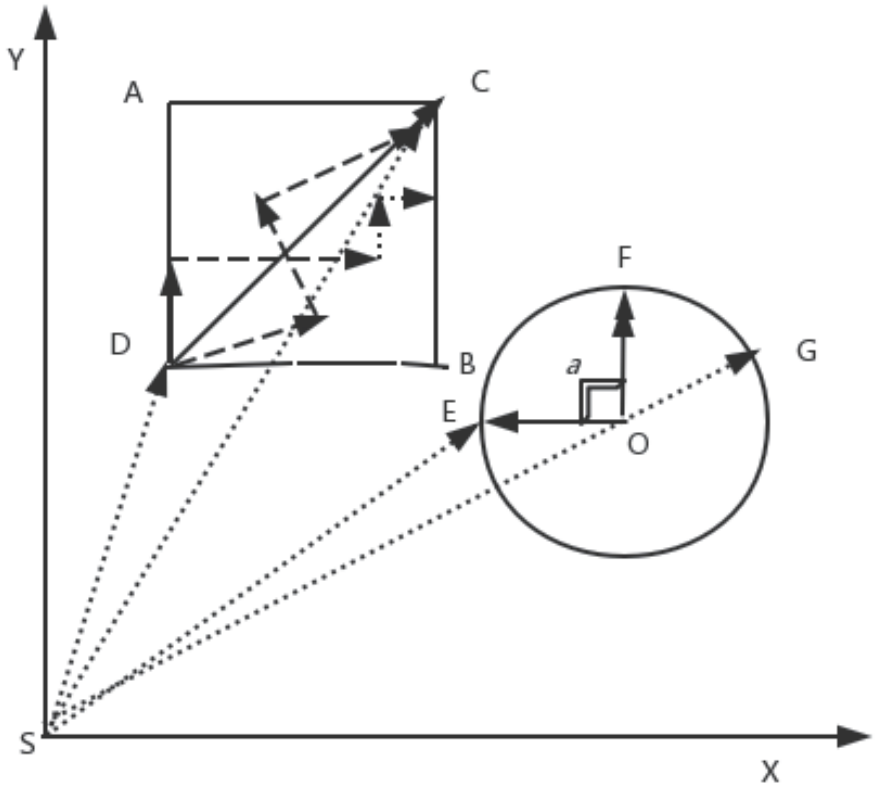

Figure 3.1: Illustration of hypercube and spherical search styles in two-dimension space.

dimensions is described as Eqs. (3.3)-(3.8).

$$S_1(X_{\alpha,d}, X_{\beta,d}) = SF() \cdot |X_{\alpha,d} - X_{\beta,d}| \cdot cos\theta, \tag{3.3}$$

$$S_2(X_{\alpha,d}, X_{\beta,d}) = SF() \cdot \|X_{\alpha,*} - X_{\beta,*}\|_2 \cdot sin\theta, \tag{3.4}$$

$$S_2(X_{\alpha,d}, X_{\beta,d}) = SF() \cdot \|X_{\alpha,*} - X_{\beta,*}\|_2 \cdot cos\theta, \tag{3.5}$$

$$S_3(X_{\alpha,d}, X_{\beta,d}) = SF() \cdot \|X_{\alpha,*} - X_{\beta,*}\|_2 \cdot \Pi_{d=1}^{dim-1} sin\theta_d, \tag{3.6}$$

$$\begin{aligned} S_3(X_{\alpha,d}, X_{\beta,d}) =& SF() \cdot \|X_{\alpha,*} - X_{\beta,*}\|_2 \cdot cos\theta_{d-1} \\ & \cdot \Pi_{d=2}^{dim-1} sin\theta_d, \end{aligned} \tag{3.7}$$

$$S_3(X_{\alpha,d}, X_{\beta,d}) = SF() \cdot \|X_{\alpha,*} - X_{\beta,*}\|_2 \cdot cos\theta_{dim-1}, \tag{3.8}$$

where $|X_{\alpha,d} - X_{\beta,d}|$ represents the absolute value of distance between $X_{\alpha,d}$ and $X_{\beta,d}$ in one dimension. $\|X_{\alpha,*} - X_{\beta,*}\|_2$ indicates the Euclidean distance between $X_{\alpha,*}$ and $X_{\beta,*}$ in high dimensions. θ is a random number in $[0, 2\pi]$ and denotes the angle between $X_{\alpha,*}$ and $X_{\beta,*}$.

According to Eqs. (3.3)-(3.8), seven search operators are proposed as follows:

$$X_{i,d}^{new} = X_{i,d} + S_m(X_{i,d}, X_{g,d}) + S_m(X_{r1,d}, X_{r2,d}), \tag{3.9}$$

$$X_{i,d}^{new} = X_{g,d} + S_m(X_{r1,d}, X_{r2,d}), \tag{3.10}$$

$$X_{i,d}^{new} = X_{g,d} + S_m(X_{r1,d}, X_{r2,d}) + S_m(X_{r3,d}, X_{r4,d}), \tag{3.11}$$

$$X_{i,d}^{new} = X_{r1,d} + S_m(X_{r2,d}, X_{r3,d}), \tag{3.12}$$

$$X_{i,d}^{new} = X_{r1,d} + S_m(X_{r2,d}, X_{r3,d}) + S_m(X_{r4,d}, X_{r5,d}), \tag{3.13}$$

$$X_{i,d}^{new} = X_{i,d} + S_m(X_{r2,d}, X_{r3,d}), \tag{3.14}$$

$$X_{i,d}^{new} = X_{i,d} + S_m(X_{r2,d}, X_{r3,d}) + S_m(X_{r4,d}, X_{r5,d}), \tag{3.15}$$

where $m \in \{1, 2, 3\}$ and X_g indicates global best individual. $X_{ri}, i \in \{1, ..., 5\}$ denote one randomly selected individual. According to Tang's research [51], among these search operators, Eq. (3.12) has the best search behavior.

Since SE has a simple mechanism and large search range, it is an efficient heuristic approach. Its search angle is distributed in $[0, 2\pi]$, thus its search trajectory is directionless, which can provide more diverse evolutionary path. However, it may generate opposite directions to result in slow convergence.

3.3 New algorithm TDSD

3.3.1 Brief introduction of CLS

CLS plays an important role and has achieved great success in memetic algorithms which combine evolutionary algorithms with local search [55]. Its search behavior can improve the quality of individuals and accelerate convergence speed due to its ergodicity and randomicity. Dynamic property of chaos ensures that algorithms can avoid sticking into stagnation in the exploitation phase. At present, CLS has been greatly developed to improve various algorithms, such as PSO [56], DE [57], krill herd algorithm [58] and so on [59, 60].

For CLS, chaotic maps are used to generate chaotic sequences. Gao et al. apply CLS to GSA [54]. Mirjalili et al. embed ten chaotic maps to tune the gravitational constant [61]. Yang et al. employ twelve chaotic maps to greatly increase the diversity of search mechanism in BSO [35]. In this paper, we adopt one popular and traditional chaotic map, namely Logistic map, described as follows:

$$Z_{k+1} = \mu Z_k(1 - Z_k), \tag{3.16}$$

where Z_k is the kth chaotic number. $\mu = 4$ and $Z_0 = 0.152$.

According to chaotic sequences of Logistic map, CLS searches all dimensions of each individual X_i to form new one X_i^{new} as follows:

$$X_i^{new} = X_i + r(U_b - L_b)(Z_k - 0.5), \tag{3.17}$$

$$r = \rho r, \tag{3.18}$$

where $r \in (0, 1)$ is chaotic search radius. U_b and L_b indicate upper and lower bounds of X_i. $\rho = 0.988$ is a shrinking parameter.

3.3.2 Principle of TDSD

SE is a simple and efficient global optimization algorithm with large search space and undirected search trajectory, however, its convergence speed is unsatisfying. HS is popularly adopted because of its high search efficiency. CLS is utilized to help individuals escape from local optima. These three kinds of search styles have own characteristics and advantages, thus we combine them to propose new algorithm TDSD, which makes individuals achieve better performances.

In TDSD, we adopt Eq. (3.14) as SE. Although SE with Eq. (3.12) has the best performance in [51], we aim to precisely control each individual's search. Thus, we select Eq. (3.14) similar to Eq. (3.12) to avoid individuals' randomness. HS is used to generate new individuals as follows:

$$X_i^{new} = X_i + F_i \cdot EP_i, \tag{3.19}$$

where F_i is a scale factor and EP_i stands for an effective path which makes X_i better than its parent. CLS adopts Eq. (3.17) to enhance individuals' exploitation abilities.

Since there are three kinds of search styles in TDSD, good control strategies among

them are devised to help individuals continuously and effectively search. For them, control strategies are described as follows:

(1) Control strategy of SE: If the offspring of SE is worse than its parent, SE should search other directions by itself. Otherwise, current direction needs to be further explored. Thus, HS should be used in next iteration.

(2) Control strategy of HS: If the offspring of HS is better than its parent, HS should continue to search current direction. Otherwise, the offspring should be further exploited. Thus, CLS is used in next iteration.

(3) Control strategy of CLS: When HS cannot find a better offspring, CLS carries out 50 times to search local areas of the offspring. If CLS finds a better offspring within 50 times, HS is used to explore the direction of this better offspring in next iteration. Otherwise, SE is applied to expand the search direction of the offspring of HS in next iteration.

The main procedures of TDSD are given as follows: 1) TDSD randomly generates a population and starts from SE. 2) When the offspring of SE is worse than its parent, SE continues to search other directions. 3) When the offspring of SE is better than its parent, HS is used. 4) When the offspring of HS is better than its parent, the same direction continues to be searched by HS. 5) When the offspring of HS is worse than its parent, CLS is executed. 6) Within 50 times, if the offspring of CLS is better, switch to HS. Otherwise, switch to SE. 7) Repeat 2)-6) until TDSD is terminated. The whole procedures can be depicted by Fig. 3.2.

Pseudocode of TDSD is shown in Algorithm 1. $U(0, 0.1)$ indicates a uniform distribution and σ is the variance of Gaussian distribution. We use a variable σ to adjust the Gaussian distribution in order to tune scale factor. In TDSD, SE, HS and CLS are executed in lines 8, 10 and 27, respectively. In each iteration, one kind of search style is selected and used to improve the search performance according to the

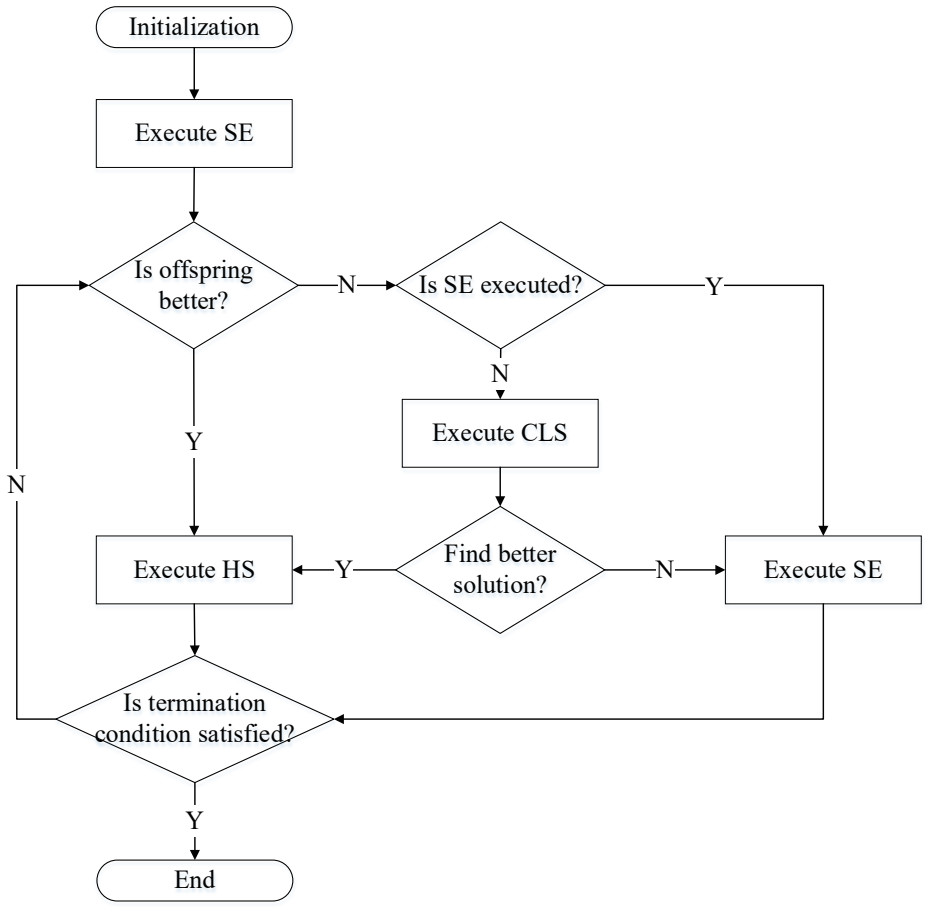

Figure 3.2: The flowchart of TDSD.

Algorithm 1: TDSD

Input: Parameters F_0, σ_0, μ, Z_0, r, ρ
Output: Optimal solution
Initialization: Randomly initialize N individuals and evaluate $f(x_i)$;
while *Termination criterion is not satisfied* **do**
  Scale factor (SF) tuning with $N(0, \sigma)$;
  Dimension selection factor (DSF) tuning;
  for *i = 1 to N* **do**
    Randomly select two individuals x_{r1} and x_{r2}, $i \neq r_1 \neq r_2$;
    if $flag_i == 0$ **then**
      $x_{i,d}^{new} = x_{i,d} + S_m(x_{r1,d}, x_{r2,d})$;
    else
      $x_i^{new} = x_i + F_i \cdot EP_i$;
    Evaluate $f(x_i^{new})$;
    if $f(x_i^{new}) < f(x_i)$ **then**
      if $flag_i == 1$ **then**
        $F_i = F_i + U(0, 0.1)$;
      else
        $flag_i = 1$;
        $F_i = F_i - U(0, 0.1)$;
        $\sigma_i = \sigma_0$;
      $EP_i = x_i^{new} - x_i$;
      $x_i = x_i^{new}$;
    else
      if $flag_i == 1$ **then**
        $F_i = F_0$;
        $flag_{suc} = 0$;
        for *k = 1 to 50* **do**
          $Z_{k+1} = \mu Z_k(1 - Z_k)$;
          $x_i^{new} = x_i + r(U_b - L_b)(Z_{k+1} - 0.5)$;
          Evaluate $f(x_i^{new})$;
          if $f(x_i^{new}) < f(x_i)$ **then**
            $flag_{suc} = 1$;
            $EP_i = x_i^{new} - x_i$;
            $x_i = x_i^{new}$;
            break;
        if $flag_{suc} == 1$ **then**
          $flag_i = 1$;
        else
          $flag_i = 0$;
        $\sigma_i = \sigma_i - U(0, 0.1)$;
      else
        $\sigma_i = \sigma_i + U(0, 0.1)$;

difference between offspring and its parent. Based on this principle, TDSD effectively integrates triple distinct search dynamics.

To illustrate the efficiency of TDSD, its time complexity is calculated where N is population size, as follows: (1) Initialization process is $O(N)$. (2) SF and DSF tuning are $O(N)$. (3) SE takes $O(N^2)$ in the worst case. (4) HS needs $O(N)$ in the worst case. (5) CLS costs $O(N^2)$ in the worst case. Thus, time complexity of TDSD can be regarded as $O(N^2)$.

3.4 Experiments

3.4.1 Experimental setup

We adopt thirty CEC2017 benchmark functions to test the performance of TDSD. F1-F3 are unimodal functions. F4-F10 are multimodal functions. F11-F20 are hybrid functions. F21-F30 are composition functions. Six state-of-the-art NMH algorithms including SE [51], HGSA [62], CLPSO [63], NCS [15], BSO [34] and DE [64] are used to compare with TDSD. The population size N is 100 and the dimension of function is 30. The running time is 30 in order to reduce random errors. The maximum number of iteration is 3000. The testing environment is PC with 3.10GHz Intel(R) Core(TM) i5-4440 CPU and 8GB run time memory by using Matlab R2013b. Parameters are used following the information provided by related literatures, shown in Table 3.2.

Mean and standard deviation (Std Dev) of optimization error between obtained solution and global optimal solution of seven algorithms are listed in Tables 3.3 and 3.4. The best result on each function is highlighted by boldface. Wilxocon rank-sum test at a significant level of $\alpha = 0.05$ is conducted to make statistical analyses where signs $+$, $\approx$ and $-$ indicate that TDSD is better, equal and worse than comparative algorithms on each function.

Table 3.2: Parameter settings of TDSD and other algorithms.

Algorithm	Parameters
TDSD	$F_0 = 2.5$, $\sigma_0 = 0.5$, $\mu = 4$, $Z_0 = 0.152$, $\rho = 0.988$, $r = 0.05$
SE	$\sigma_0 = 0.1$, using Eq. (3.12)
HGSA	$G_0 = 100$, $L = 100$, $w_1(t) = 1 - t^6/T^6$, $w_2(t) = t^6/T^6$
CLPSO	$\omega_0 = 0.9$, $\omega_1 = 0.4$, $c = 1.49445$, $m = 7$, $p_c = 0.05 \sim 0.5$
NCS	$r = 0.99$, $epoch = 10$
BSO	$m = 5$, $p_{5a} = 0.2$, $p_{6b} = 0.8$, $p_{6biii} = 0.4$, $p_{6c} = 0.5$
DE	$F = 0.5$, $CR = 0.9$

3.4.2 Experimental results

From Tables 3.3 and 3.4, we can see that TDSD obtains more better results than its peers. To be specific, TDSD is superior to SE, HGSA, CLPSO, NCS, BSO and DE on 20, 17, 18, 28, 23, 26 functions, respectively. These results demonstrate good search performance of TDSD. In six NMH algorithms, SE uses spherical search and the others adopt HS according to their principles. Comparison between TDSD and SE illustrates that three kinds of search styles are better than single spherical search. HS and CLS further improve SE. Likewise, comparison between TDSD and the others indicate that triple distinct search dynamics implement better performance than single HS.

TDSD is a self-adaptive search algorithm. It can adaptively adjust search strategies according to current search situation. SE has a large search range, hence it can extensively search the space and avoid trapping into local optima. HS is a directional search. Based on new found solutions, it can guide individuals towards more promising area. Thus, it improves the exploration ability. CLS implements local search to enhance the exploitation ability. TDSD adaptively switches these three search strategies to reinforce its overall search ability. In six NMH algorithms, individuals only have one kind of search strategy. In other words, they have the same search behaviors in each algorithm. However, in TDSD, each individual executes different search

Table 3.3: Experimental results of TDSD versus three NMH algorithms on CEC2017 benchmark functions (1).

Fun.	TDSD Mean ± Std Dev	SE Mean ±Std Dev	HGSA Mean ±Std Dev	CLPSO Mean ±Std Dev
F1	**1.7446E+03 ± 9.0697E+02**	2.2251E+03 ± 1.9277E+03 ≈	2.5825E+03 ± 2.4958E+03 ≈	2.9592E+03 ± 1.0811E+03 +
F2	1.3076E+12 ± 2.3022E+12	1.8895E+17 ± 2.5016E+17 +	4.1639E+05 ± 2.0000E+06 −	1.4791E+16 ± 2.1065E+16 +
F3	4.0435E+04 ± 9.5356E+03	5.6122E+04 ± 8.0543E+03 +	4.3258E+04 ± 5.4921E+03 ≈	2.5897E+04 ± 4.8598E+03 −
F4	**2.0235E+01 ± 2.0113E+01**	1.0810E+02 ± 7.4898E+00 +	1.1915E+02 ± 2.6272E+00 +	9.6200E+01 ± 1.0917E+01 +
F5	7.9872E+01 ± 1.1533E+01	**7.3308E+01 ± 8.3327E+00** −	1.5299E+02 ± 1.2839E+01 +	9.1924E+01 ± 1.4460E+01 +
F6	3.2773E+00 ± 6.7748E-01	**1.5916E-13 ± 5.6647E-14** −	8.1917E+00 ± 4.5442E+00 +	4.3493E-06 ± 2.2016E-06 −
F7	1.3181E+02 ± 1.3996E+01	1.1504E+02 ± 7.0235E+00 −	**4.0761E+01 ± 3.0080E+00** −	1.5252E+02 ± 1.9497E+01 +
F8	8.3313E+01 ± 8.1652E+00	**8.0863E+01 ± 6.5922E+00** ≈	9.9794E+01 ± 9.0330E+00 +	9.0900E+01 ± 1.3795E+01 +
F9	1.7129E+03 ± 3.4730E+02	1.1989E+00 ± 9.8020E-01 −	**7.9581E-14 ± 5.2988E-14** −	2.3782E+01 ± 1.1816E+01 −
F10	**2.1866E+03 ± 2.2587E+02**	3.7695E+03 ± 2.3024E+02 +	3.2067E+03 ± 2.9282E+02 +	3.6270E+03 ± 6.5767E+02 +
F11	**6.8555E+01 ± 2.4028E+01**	1.1818E+02 ± 2.1471E+01 +	9.7308E+01 ± 2.9810E+01 +	7.1063E+01 ± 1.1606E+01 ≈
F12	2.9161E+05 ± 1.7235E+05	1.9639E+06 ± 6.7720E+05 +	**1.2777E+05 ± 8.1524E+04** −	5.2864E+05 ± 2.1390E+05 +
F13	**6.9559E+02 ± 3.3086E+02**	6.0232E+04 ± 3.2488E+04 +	1.3349E+04 ± 5.3217E+03 +	9.6227E+03 ± 4.6658E+03 +
F14	8.4226E+03 ± 6.0815E+03	5.3949E+04 ± 3.2298E+04 +	5.3218E+03 ± 3.0488E+03 −	1.0750E+04 ± 7.3693E+03 ≈
F15	3.5640E+02 ± 2.2676E+02	2.1429E+04 ± 8.9177E+03 +	7.0068E+02 ± 7.2068E+02 ≈	9.6215E+02 ± 3.8530E+02 +
F16	4.8592E+02 ± 1.1375E+02	6.0382E+02 ± 1.2553E+02 +	1.2338E+03 ± 2.3248E+02 +	**4.4267E+02 ± 1.0845E+02** ≈
F17	**9.3752E+01 ± 4.0179E+01**	1.2674E+02 ± 3.8258E+01 +	1.0655E+03 ± 1.9931E+02 +	1.0658E+02 ± 2.3862E+01 +
F18	8.1465E+04 ± 3.6777E+04	2.5746E+05 ± 1.1435E+05 +	5.9831E+04 ± 1.4748E+04 −	1.2723E+05 ± 4.1387E+04 +
F19	1.5314E+02 ± 1.0351E+02	2.3332E+04 ± 1.1535E+04 +	3.5238E+03 ± 1.2483E+03 +	3.7730E+02 ± 2.7881E+02 +
F20	**1.3811E+02 ± 5.4266E+01**	1.7656E+02 ± 6.4731E+01 +	8.5598E+02 ± 2.2429E+02 +	1.5515E+02 ± 5.4485E+01 ≈
F21	2.2212E+02 ± 8.1525E+01	2.7758E+02 ± 1.2908E+01 ≈	3.0985E+02 ± 5.9021E+01 +	2.5388E+02 ± 5.6592E+01 ≈
F22	1.1038E+02 ± 1.8762E+00	1.3649E+02 ± 1.3178E+01 +	**1.0000E+02 ± 3.9053E-09** −	1.2256E+02 ± 9.0168E+00 +
F23	**4.5389E+02 ± 1.7290E+02**	4.2519E+02 ± 9.3929E+00 +	4.5683E+02 ± 1.3328E+02 +	4.1710E+02 ± 9.9420E+00 ≈
F24	**4.2415E+02 ± 1.8954E+02**	5.1770E+02 ± 9.9031E+00 −	5.2286E+02 ± 3.5773E+01 ≈	4.9208E+02 ± 3.6518E+01 −
F25	**3.8263E+02 ± 1.1444E+01**	3.8727E+02 ± 1.4479E-01 +	3.8995E+02 ± 7.5869E+00 +	3.8701E+02 ± 4.6671E-01 +
F26	**2.2014E+02 ± 5.5468E+00**	1.4859E+03 ± 2.5408E+02 +	2.5333E+02 ± 5.0742E+01 ≈	7.3689E+02 ± 1.5348E+02 +
F27	5.2013E+02 ± 6.1588E+00	5.1789E+02 ± 4.1957E+00 −	5.5121E+02 ± 2.0816E+01 +	**5.1100E+02 ± 3.4847E+00** −
F28	4.0840E+02 ± 1.4862E+01	4.2873E+02 ± 8.1138E+00 +	**3.0725E+02 ± 2.8167E+01** −	4.0830E+02 ± 2.8767E+00 −
F29	**5.8901E+02 ± 4.5694E+01**	5.9054E+02 ± 5.9965E+01 ≈	1.1500E+03 ± 1.8828E+02 +	6.1388E+02 ± 5.2553E+01 +
F30	**5.0400E+03 ± 8.3171E+02**	2.5010E+04 ± 9.1420E+03 +	7.9894E+03 ± 2.6002E+03 +	1.8097E+04 ± 5.0441E+03 +
+/≈/-		20/ 4 /6	17/ 5 /8	18/ 6 /6

Table 3.4: Experimental results of TDSD versus three NMH algorithms on CEC2017 benchmark functions (2).

Fun.	NCS Mean ±Std Dev	BSO Mean ±Std Dev	DE Mean ±Std Dev
F1	9.3488E+07 ± 1.2200E+07 +	2.3695E+03 ± 1.9534E+03 ≈	1.0624E+07 ± 1.7674E+06 +
F2	1.6886E+14 ± 1.8752E+14 +	**8.5863E+02 ± 1.3057E+03** −	4.0866E+24 ± 5.0206E+24 +
F3	6.3579E+04 ± 1.1372E+04 +	**2.3356E+02 ± 2.6618E+02** −	4.8272E+04 ± 4.8272E+03 +
F4	1.1523E+02 ± 1.8073E+01 +	6.7337E+01 ± 2.1877E+01 +	2.3535E+02 ± 1.6436E+01 +
F5	3.3676E+02 ± 2.9773E+01 +	1.8655E+02 ± 4.0450E+01 +	2.4717E+02 ± 1.4400E+01 +
F6	7.8200E+01 ± 6.0094E+00 +	5.1884E+01 ± 7.0118E+00 +	3.6621E+01 ± 2.4160E+00 +
F7	5.9672E+02 ± 7.4377E+01 +	4.5445E+02 ± 9.6493E+01 +	3.2612E+02 ± 1.2082E+01 +
F8	2.8193E+02 ± 2.2344E+01 +	1.4679E+02 ± 2.8204E+01 +	2.4457E+02 ± 1.1535E+01 +
F9	1.7586E+04 ± 2.4106E+03 +	3.0811E+03 ± 6.9519E+02 +	2.8307E+03 ± 3.4700E+02 +
F10	3.8919E+03 ± 2.3180E+02 +	4.2966E+03 ± 5.1641E+02 +	7.1640E+03 ± 2.2593E+02 +
F11	2.6308E+02 ± 4.6974E+01 +	1.2943E+02 ± 4.0497E+01 +	1.7590E+02 ± 1.6468E+01 +
F12	1.7824E+07 ± 4.2028E+06 +	1.7701E+06 ± 1.2480E+06 +	1.8462E+06 ± 3.0375E+05 +
F13	2.3996E+06 ± 5.6699E+05 +	5.2301E+04 ± 2.8456E+04 +	7.4878E+02 ± 6.7608E+01 +
F14	8.9670E+03 ± 4.7464E+03 ≈	5.0009E+03 ± 4.6624E+03 −	**9.1439E+01 ± 5.1645E+00** −
F15	2.3404E+05 ± 8.6920E+04 +	2.7999E+04 ± 1.6069E+04 +	**1.3895E+02 ± 1.3121E+01** −
F16	1.1738E+03 ± 1.6706E+02 +	1.6047E+03 ± 4.2387E+02 +	1.6492E+03 ± 1.5272E+02 +
F17	3.6643E+02 ± 8.9316E+01 +	7.8457E+02 ± 2.5562E+02 +	5.3564E+02 ± 7.4791E+01 +
F18	1.5173E+05 ± 4.8336E+04 +	1.1951E+05 ± 1.0252E+05 ≈	**1.1820E+02 ± 8.8527E+00** −
F19	9.8129E+05 ± 3.6206E+05 +	1.4961E+05 ± 6.4292E+04 +	**4.5806E+01 ± 3.6850E+00** −
F20	5.9343E+02 ± 1.0448E+02 +	6.6740E+02 ± 1.7372E+02 +	6.0216E+02 ± 1.0326E+02 +
F21	**1.8971E+02 ± 1.3784E+02** −	4.0121E+02 ± 4.4805E+01 +	4.3440E+02 ± 1.1808E+01 +
F22	3.3036E+02 ± 7.3901E+02 +	3.8332E+03 ± 1.7705E+03 +	2.4706E+02 ± 3.8870E+01 +
F23	6.5009E+02 ± 1.2460E+02 +	9.8581E+02 ± 1.2670E+02 +	5.7713E+02 ± 6.0703E+01 +
F24	5.1806E+02 ± 2.9403E+02 +	1.1014E+03 ± 1.1344E+02 +	6.8755E+02 ± 1.4046E+01 +
F25	4.1605E+02 ± 1.5412E+01 +	3.8517E+02 ± 1.4585E+01 −	4.5099E+02 ± 6.7690E+00 +
F26	4.1034E+02 ± 1.0258E+02 +	5.5626E+03 ± 1.5683E+03 +	1.7050E+03 ± 1.2513E+02 +
F27	5.9316E+02 ± 2.0310E+01 +	1.1230E+03 ± 2.9051E+02 +	6.4745E+02 ± 1.3231E+01 +
F28	4.6738E+02 ± 2.0039E+01 +	4.0525E+02 ± 2.5740E+01 −	5.5008E+02 ± 1.6626E+01 +
F29	1.2794E+03 ± 8.8581E+01 +	1.4810E+03 ± 2.7931E+02 +	1.4098E+03 ± 1.2188E+02 +
F30	1.8006E+06 ± 4.4061E+05 +	5.7103E+05 ± 3.5006E+05 +	7.4916E+04 ± 2.0269E+04 +
+/≈/-	28 / 1 /1	23/ 2 /5	26/ 0 /4

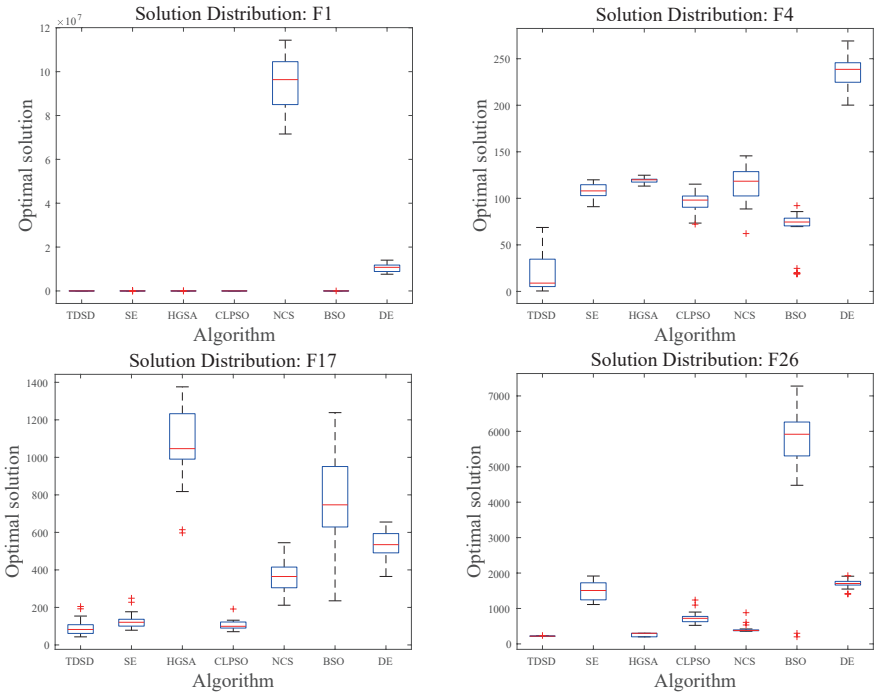

Figure 3.3: The box-and-whisker diagrams of optimal solutions obtained by seven algorithms on F1, F4, F17 and F26.

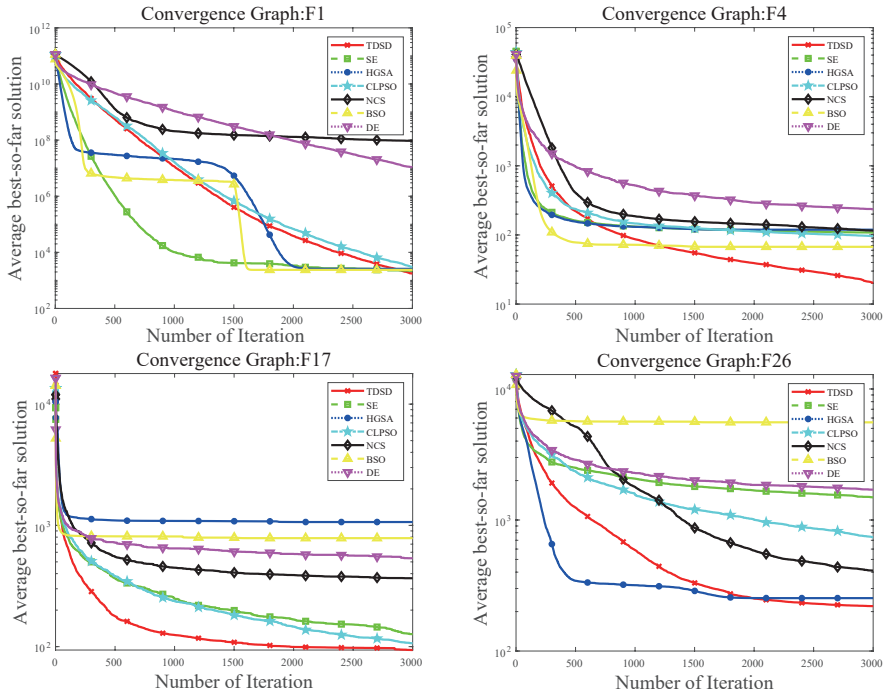

Figure 3.4: The convergence graphs of average best-so-far solutions obtained by seven algorithms on F1, F4, F17 and F26.

strategies according to own search situation. Their search diversity is remarkably improved. Consequently, they can find better solutions than six NMH algorithms on numerous functions.

We plot box-and-whisker diagrams and convergence graphs on F1, F4, F17 and F26 to show characteristics of seven algorithms. In Fig. 3.3, box-and-whisker diagrams of optimal solutions obtained by seven algorithms are shown. From it, we can observe that TDSD has smaller distribution of optimal solutions than others'. On these four functions, optimal solutions of six NMH algorithms change great whereas optimal solutions of TDSD maintain relatively steady. It indicates the effectiveness and stability of TDSD. Fig. 3.4 displays convergence graphs of average best-so-far solutions of seven algorithms. The horizontal axis indicates the number of iteration and the vertical axis denotes log value of average best-so-far solution. According to Fig. 3.4, we can see that TDSD shows gradual convergence on these four functions. Six NMH algorithms either trap into premature convergence or do not find superior solutions. Compared with them, TDSD shows more effective convergence characteristics. It can maintain good convergence speed. In Fig. 3.4, TDSD does not converge quickly in the early search process, hence premature convergence does not occur. In the late search process, TDSD still finds better solutions, which shows its good exploitation ability. In other words, exploration and exploitation abilities of TDSD are balanced such that its convergence is effective and continuous. To show the convergence of individuals, we plot their search graphs on F10 with 2 dimensions in Fig. 3.5. The population size is 100 and the number of iteration is 3000. F10 is a multi-modal function which has many local optima. Fig. 3.5(a) shows one hundred individuals are randomly distributed with the first iteration. Fig. 3.5(b) displays that individuals converge towards some areas with the 200th iteration. Fig. 3.5(c) exhibits that individuals further reduce convergence areas. Fig. 3.5(d) reveals final

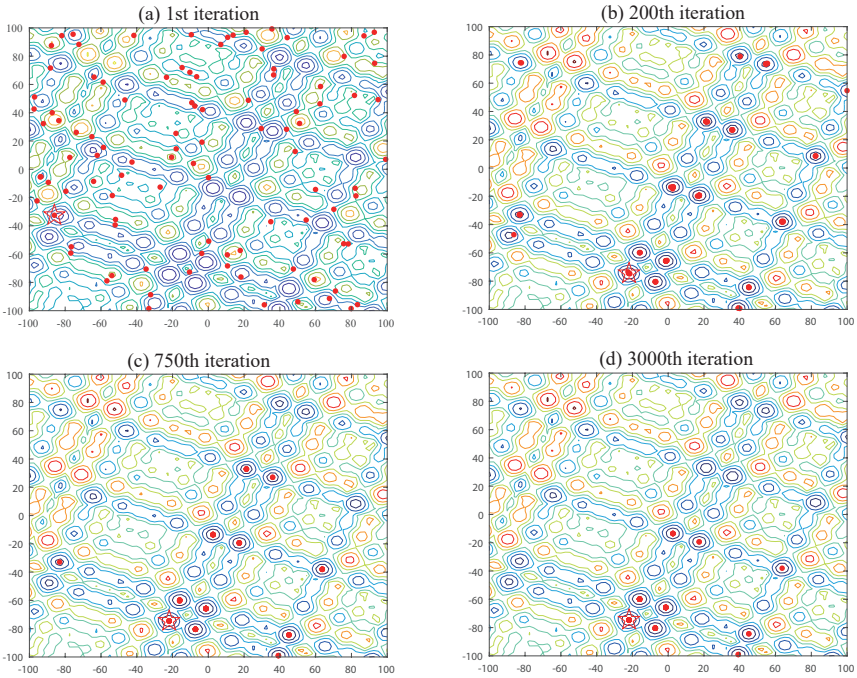

Figure 3.5: The search graphs of TDSD on F10 with 2 dimensions where indicates the best-so-far solution.

convergence of individuals with the 3000th iteration. From Fig. 3.5, we can see that individuals gradually converge into different areas, suggesting high search diversity. This is because each individual only depends on own search situation and strategy. All individuals do not completely converge into one point. Thus, TDSD maintains the population diversity to improve its search performance.

3.4.3 Analysis of Parameters

TDSD uses CLS to further exploit local areas of solutions found by HS. In TDSD, the number of times of CLS ($nCLS$) may influence the exploitation ability. To analyze it, we set $nCLS$ to 25, 50, 75 and 100. Experimental results are listed in Table 3.5.

According to Table 3.5, we can find that $nCLS = 50$ is slightly better than other values. Compared with $nCLS = 25$, CLS executes 50 times to obtain better solutions on four functions. It indicates that 50 times for CLS is more sufficient. However, experimental results of $nCLS = 75$ and $nCLS = 100$ do not show that more number

of times of CLS is more effective, and they are worse than $nCLS = 50$ on one function. Thus, we conclude that 50 times is suitable for CLS.

In addition to CLS, SE and HS have two parameters F_0 and σ_0. Although these two parameters adaptively change during the execution of TDSD, their initial values may also influence the search performance. To determine them, we set F_0 to 1, 2.5 and 5, and σ_0 to 0.1, 0.5 and 0.9. Thus, nine groups of parameter settings are conducted. Their experimental and statistical results are shown in Tables 3.6 and 3.7.

From Tables 3.6 and 3.7, we can observe that the group of $F_0 = 2.5$ and $\sigma_0 = 0.5$ is relatively superior to other groups. To be specific, when $F_0 = 2.5$, $\sigma_0 = 0.5$ is slightly better than $\sigma_0 = 0.1$, but equivalent to $\sigma_0 = 0.9$ according to statistical results. It indicates that a great σ_0 value is slightly good. Compared with the groups of $F_0 = 1$, the group of $F_0 = 2.5$ and $\sigma_0 = 0.5$ is better. It may be because a small F_0 value decreases the exploration ranges of individuals. Also, the groups of $F_0 = 5$ are inferior because they may excessively increase the search areas of individuals such that overstepping the boundary is prone to occur. Thus, based on these results, we regard $F_0 = 2.5$ and $\sigma_0 = 0.5$ as a group of suitable parameter setting.

3.4.4 Three real-world optimization problems

To evaluate the practicality of TDSD, we test it on three real-world optimization problems from CEC2011. They are parameter estimation for frequency-modulated sound waves problem (FMSWP), spread spectrum radar polly phase code design (SSRP-PCD), and optimal control of a non-linear stirred tank reactor problem (NLSTRP). Their specific details can be referred in [65]. Seven comparative algorithms including WOA [66], DE [64], CGSA-P [67], BSO [34], EPSDE [68], SaDE [69] and jDE [70] are used. Their parameter settings are adopted according to corresponding literatures. The maximum number of iteration for three problems is 600, 2000 and 100, respectively. Experimental results are listed in Tables 3.8, 3.9 and 3.10 where Mean,

Std, Best and Worst indicate mean, standard deviation, best and worst solutions, respectively.

From Tables 3.8, 3.9 and 3.10, we can find that TDSD obtains the least means on these three problems. In addition, TDSD has the best solutions on FMSWP and NLSTRP. Thus, experimental results demonstrate that TDSD has the practicality for real-world optimization problems. Compared with various NMH algorithms, TDSD uses triple distinct search dynamics to optimize problems and find better solutions.

3.4.5 Comparison with champion algorithms

Four champion algorithms including EBOwithCMAR [71], LSHADE-cnEpSin [72], LSHADE-SPACMA [73] and SHADE [74] are used to compare with TDSD on thirty CEC2017 benchmark functions with 30 dimensions and three real-world optimization problems. EBOwithCMAR uses covariance matrix to improve the local search ability of effective butterfly optimizer. LSHADE-cnEpSin constitutes ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood. LSHADE-SPACMA combines a semi-parameter adaptation method with modified CMA-ES. SHADE is an adaptive DE with history-based parameter adaptation. Their experimental results are listed in Tables 3.11, 3.12, 3.13 and 3.14.

From Table 3.11, we can see that TDSD is better than four champion algorithms on 3 functions, i.e., F4, F25 and F26. Although TDSD does not totally perform better than four champion algorithms, it is still a promising one. Compared with those complex methods, TDSD only executes a simple control strategy to adaptively adjust the search style. Thus, its performance is able to be improved by diverse strategies in the future. Besides, the fact that TDSD finds better solutions than four champion algorithms on 3 functions indicates its search is the best on a part of functions.

According to Tables 3.12, 3.13 and 3.14, we can observe that TDSD obtains the least mean on NLSTRP whereas it does not perform the best on FMSWP and SSRP-

PCD. Four champion algorithms show different optimization abilities for these three problems. For FMSWP and SSRPPCD, LSHADE-cnEpSin and EBOwithCMAR have the least mean, respectively. However, TDSD can obtain the best solution on FMSWP and its best solution for SSRPPCD is the medium rank, suggesting that it is capable of optimizing these two problems.

3.5 Conclusion

In this paper we propose a novel algorithm called TDSD for functions and real-world optimization problems. Three kinds of distinct search styles are integrated to enhance the search performance. According to their characteristics, we design effective control strategies to switch the search style of TDSD. Experimental results indicate TDSD outperforms other state-of-the-art algorithms in terms of its effectiveness and robustness. In the future, we plan to study more search styles and try to incorporate them for improvement of algorithms.

Table 3.5: Experimental results of different $nCLS$.

Fun.	$nCLS = 50$ Mean ± Std Dev	$nCLS = 25$ Mean ±Std Dev	$nCLS = 75$ Mean ±Std Dev	$nCLS = 100$ Mean ±Std Dev
F1	1.74E+03 ± 9.07E+02	1.79E+03 ± 1.12E+03 ≈	1.77E+03 ± 1.11E+03 ≈	**1.72E+03 ± 9.81E+02** ≈
F2	**1.31E+12 ± 2.30E+12**	1.81E+12 ± 2.70E+12 ≈	2.48E+12 ± 4.89E+12 ≈	2.61E+12 ± 5.62E+12 ≈
F3	4.04E+04 ± 9.54E+03	4.75E+04 ± 7.25E+03 +	3.92E+04 ± 9.30E+03 ≈	**3.86E+04 ± 7.71E+03** ≈
F4	**2.02E+01 ± 2.01E+01**	2.48E+01 ± 2.20E+01 ≈	2.53E+01 ± 2.16E+01 ≈	2.60E+01 ± 2.56E+01 ≈
F5	**7.99E+01 ± 1.15E+01**	8.37E+01 ± 8.65E+00 ≈	8.20E+01 ± 1.24E+01 ≈	8.39E+01 ± 9.40E+00 ≈
F6	3.28E+00 ± 6.77E-01	**3.05E+00 ± 6.48E-01** ≈	3.07E+00 ± 6.06E-01 ≈	3.31E+00 ± 4.67E-01 ≈
F7	1.32E+02 ± 1.40E+01	1.32E+02 ± 1.47E+01 ≈	1.33E+02 ± 1.43E+01 ≈	**1.30E+02 ± 1.24E+01** ≈
F8	8.33E+01 ± 8.17E+00	8.48E+01 ± 8.86E+00 ≈	**8.22E+01 ± 9.72E+00** ≈	8.66E+01 ± 8.77E+00 ≈
F9	1.71E+03 ± 3.47E+02	**1.70E+03 ± 3.32E+02** ≈	1.75E+03 ± 3.36E+02 ≈	1.73E+03 ± 3.21E+02 ≈
F10	2.19E+03 ± 2.26E+02	2.19E+03 ± 2.20E+02 ≈	2.17E+03 ± 2.26E+02 ≈	**2.12E+03 ± 3.23E+02** ≈
F11	6.86E+01 ± 2.40E+01	6.98E+01 ± 2.11E+01 ≈	6.92E+01 ± 1.95E+01 ≈	**6.77E+01 ± 2.12E+01** ≈
F12	2.92E+05 ± 1.72E+05	**2.58E+05 ± 1.25E+05** ≈	2.60E+05 ± 1.30E+05 ≈	3.11E+05 ± 1.40E+05 ≈
F13	**6.96E+02 ± 3.31E+02**	8.45E+02 ± 3.83E+02 +	7.95E+02 ± 4.01E+02 ≈	8.19E+02 ± 3.94E+02 ≈
F14	8.42E+03 ± 6.08E+03	9.65E+03 ± 6.36E+03 ≈	**7.55E+03 ± 5.38E+03** ≈	8.20E+03 ± 5.47E+03 ≈
F15	3.56E+02 ± 2.27E+02	4.42E+02 ± 2.24E+02 +	**3.07E+02 ± 1.71E+02** ≈	3.71E+02 ± 2.59E+02 ≈
F16	4.86E+02 ± 1.14E+02	5.01E+02 ± 1.18E+02 ≈	4.70E+02 ± 1.13E+02 ≈	**4.68E+02 ± 1.17E+02** ≈
F17	**9.38E+01 ± 4.02E+01**	1.11E+02 ± 4.70E+01 ≈	1.15E+02 ± 4.61E+01 +	1.15E+02 ± 4.87E+01 +
F18	8.15E+04 ± 3.68E+04	9.35E+04 ± 3.14E+04 +	7.71E+04 ± 2.96E+04 ≈	**7.54E+04 ± 2.65E+04** ≈
F19	1.53E+02 ± 1.04E+02	1.38E+02 ± 8.49E+01 ≈	**1.37E+02 ± 7.74E+01** ≈	1.53E+02 ± 7.41E+01 ≈
F20	1.38E+02 ± 5.43E+01	1.54E+02 ± 6.49E+01 ≈	**1.35E+02 ± 5.02E+01** ≈	1.49E+02 ± 6.01E+01 ≈
F21	2.22E+02 ± 8.15E+01	**2.21E+02 ± 7.96E+01** ≈	2.35E+02 ± 7.73E+01 ≈	2.32E+02 ± 7.55E+01 ≈
F22	**1.10E+02 ± 1.88E+00**	1.10E+02 ± 2.23E+00 ≈	1.11E+02 ± 1.78E+00 ≈	1.10E+02 ± 2.16E+00 ≈
F23	4.14E+02 ± 1.60E+01	**4.11E+02 ± 1.60E+01** ≈	4.11E+02 ± 1.60E+01 ≈	4.15E+02 ± 1.61E+01 ≈
F24	4.54E+02 ± 1.73E+02	**3.71E+02 ± 2.00E+02** ≈	4.02E+02 ± 1.81E+02 ≈	4.20E+02 ± 1.80E+02 ≈
F25	**3.83E+02 ± 1.14E+01**	3.85E+02 ± 1.27E+00 ≈	3.85E+02 ± 1.12E+00 ≈	3.85E+02 ± 9.46E-01 ≈
F26	2.20E+02 ± 5.55E+00	2.22E+02 ± 1.80E+01 ≈	2.24E+02 ± 1.60E+01 ≈	**2.20E+02 ± 5.10E+00** ≈
F27	5.20E+02 ± 6.16E+00	**5.18E+02 ± 5.65E+00** ≈	5.18E+02 ± 6.50E+00 ≈	5.19E+02 ± 5.55E+00 ≈
F28	**4.08E+02 ± 1.49E+01**	4.10E+02 ± 5.71E+00 ≈	4.09E+02 ± 6.20E+00 ≈	4.11E+02 ± 4.69E+00 ≈
F29	5.89E+02 ± 4.57E+01	5.85E+02 ± 5.34E+01 ≈	**5.83E+02 ± 6.41E+01** ≈	5.96E+02 ± 7.20E+01 ≈
F30	**5.04E+03 ± 8.32E+02**	5.20E+03 ± 9.73E+02 ≈	5.17E+03 ± 1.14E+03 ≈	5.07E+03 ± 1.19E+03 ≈
+/≈/-		4/ 26 /0	1/ 29 /0	1/ 29 /0

Table 3.6: Experimental results of different settings of F_0 and σ_0.

Parameter	F1	F2	F3	F4	F5	F6
$F_0 = 2.5, \sigma_0 = 0.5$	1.74E+03 ± 9.07E+02	1.31E+12 ± 2.30E+12	**4.04E+04 ± 9.54E+03** +	**2.02E+01 ± 2.01E+01**	7.99E+01 ± 1.15E+01	3.28E+00 ± 6.77E-01
$F_0 = 2.5, \sigma_0 = 0.1$	**1.31E+03 ± 6.92E+02** −	**1.23E+12 ± 3.20E+12** ≈	4.47E+04 ± 7.88E+03 +	2.74E+01 ± 2.09E+01 +	8.10E+01 ± 1.02E+01 ≈	3.21E+00 ± 4.86E-01 ≈
$F_0 = 2.5, \sigma_0 = 0.9$	1.88E+03 ± 1.07E+03 ≈	3.30E+12 ± 9.44E+12 ≈	4.26E+04 ± 1.09E+04 ≈	2.08E+01 ± 2.01E+01 ≈	7.86E+01 ± 7.85E+00 ≈	**2.90E+00 ± 5.61E-01** −
$F_0 = 1, \sigma_0 = 0.1$	4.19E+03 ± 1.82E+03 +	4.58E+12 ± 6.92E+12 +	4.41E+04 ± 7.79E+03 +	3.07E+01 ± 2.34E+01 +	8.38E+01 ± 9.35E+00 ≈	3.69E+00 ± 5.91E-01 +
$F_0 = 1, \sigma_0 = 0.5$	4.37E+03 ± 1.94E+03 +	7.85E+12 ± 2.35E+13 +	4.25E+04 ± 9.87E+03 ≈	2.70E+01 ± 2.18E+01 ≈	8.18E+01 ± 9.92E+00 ≈	3.66E+00 ± 5.51E-01 +
$F_0 = 1, \sigma_0 = 0.9$	5.32E+03 ± 2.02E+03 +	8.38E+12 ± 1.45E+13 +	4.48E+04 ± 9.19E+03 +	3.34E+01 ± 2.32E+01 +	8.37E+01 ± 1.17E+01 ≈	3.32E+00 ± 5.77E-01 ≈
$F_0 = 5, \sigma_0 = 0.1$	2.08E+03 ± 1.48E+03 ≈	1.73E+12 ± 3.62E+12 ≈	4.45E+04 ± 7.74E+03 ≈	3.79E+01 ± 2.25E+01 +	8.13E+01 ± 1.33E+01 ≈	3.28E+00 ± 5.62E-01 ≈
$F_0 = 5, \sigma_0 = 0.5$	2.40E+03 ± 1.05E+03 +	7.87E+12 ± 1.62E+13 +	4.45E+04 ± 9.26E+03 +	2.46E+01 ± 2.13E+01 ≈	8.06E+01 ± 9.39E+00 ≈	3.04E+00 ± 3.90E-01 −
$F_0 = 5, \sigma_0 = 0.9$	2.76E+03 ± 1.51E+03 +	5.59E+12 ± 9.96E+12 +	4.95E+04 ± 1.12E+04 +	2.97E+01 ± 2.70E+01 ≈	**7.61E+01 ± 1.26E+01** ≈	2.96E+00 ± 5.75E-01 −
	F7	F8	F9	F10	F11	F12
$F_0 = 2.5, \sigma_0 = 0.5$	1.32E+02 ± 1.40E+01	8.33E+01 ± 8.17E+00	1.71E+03 ± 3.47E+02	2.19E+03 ± 2.26E+02	6.86E+01 ± 2.40E+01	2.92E+05 ± 1.72E+05
$F_0 = 2.5, \sigma_0 = 0.1$	1.36E+02 ± 1.37E+01 ≈	8.35E+01 ± 1.19E+01 ≈	1.66E+03 ± 3.27E+02 ≈	2.23E+03 ± 3.04E+02 ≈	**6.69E+01 ± 2.18E+01** ≈	2.63E+05 ± 1.31E+05 ≈
$F_0 = 2.5, \sigma_0 = 0.9$	1.26E+02 ± 1.45E+01 ≈	8.50E+01 ± 7.47E+00 ≈	1.67E+03 ± 3.43E+02 ≈	**2.15E+03 ± 2.54E+02** ≈	7.12E+01 ± 1.61E+01 ≈	2.66E+05 ± 1.35E+05 ≈
$F_0 = 1, \sigma_0 = 0.1$	1.36E+02 ± 1.42E+01 ≈	8.63E+01 ± 1.13E+01 ≈	1.90E+03 ± 3.90E+02 ≈	2.19E+03 ± 2.26E+02 ≈	7.47E+01 ± 2.13E+01 ≈	2.96E+05 ± 1.41E+05 ≈
$F_0 = 1, \sigma_0 = 0.5$	1.36E+02 ± 1.28E+01 ≈	8.44E+01 ± 1.37E+01 ≈	1.77E+03 ± 4.32E+02 ≈	2.23E+03 ± 1.80E+02 ≈	6.74E+01 ± 2.34E+01 ≈	2.98E+05 ± 1.71E+05 ≈
$F_0 = 1, \sigma_0 = 0.9$	1.35E+02 ± 1.41E+01 ≈	8.54E+01 ± 1.04E+01 ≈	1.72E+03 ± 3.34E+02 ≈	2.20E+03 ± 2.02E+02 ≈	7.41E+01 ± 1.80E+01 ≈	3.18E+05 ± 1.97E+05 ≈
$F_0 = 5, \sigma_0 = 0.1$	1.35E+02 ± 1.33E+01 ≈	8.51E+01 ± 1.30E+01 ≈	1.73E+03 ± 3.58E+02 ≈	2.18E+03 ± 2.07E+02 ≈	7.09E+01 ± 2.39E+01 ≈	**2.46E+05 ± 9.88E+04** ≈
$F_0 = 5, \sigma_0 = 0.5$	1.34E+02 ± 1.39E+01 ≈	8.16E+01 ± 1.17E+01 ≈	1.57E+03 ± 2.99E+02 ≈	2.16E+03 ± 2.65E+02 ≈	7.31E+01 ± 2.11E+01 ≈	2.90E+05 ± 1.55E+05 ≈
$F_0 = 5, \sigma_0 = 0.9$	**1.31E+02 ± 1.19E+01** ≈	**7.91E+01 ± 9.32E+00** −	**1.51E+03 ± 3.17E+02** −	2.20E+03 ± 2.14E+02 ≈	7.26E+01 ± 1.94E+01 ≈	2.99E+05 ± 1.64E+05 ≈
	F13	F14	F15	F16	F17	F18
$F_0 = 2.5, \sigma_0 = 0.5$	**6.96E+02 ± 3.31E+02**	8.42E+03 ± 6.08E+03	3.56E+02 ± 2.27E+02	4.86E+02 ± 1.14E+02	**9.38E+01 ± 4.02E+01**	8.15E+04 ± 3.68E+04
$F_0 = 2.5, \sigma_0 = 0.1$	7.79E+02 ± 2.70E+02 +	7.25E+03 ± 5.65E+03 ≈	**3.08E+02 ± 1.66E+02** ≈	4.68E+02 ± 1.25E+02 ≈	1.05E+02 ± 4.19E+01 ≈	7.66E+04 ± 3.12E+04 ≈
$F_0 = 2.5, \sigma_0 = 0.9$	8.42E+02 ± 3.73E+02 +	**6.66E+03 ± 5.11E+03** ≈	3.46E+02 ± 1.85E+02 ≈	4.89E+02 ± 1.14E+02 ≈	1.02E+02 ± 4.29E+01 ≈	8.39E+04 ± 3.21E+04 ≈
$F_0 = 1, \sigma_0 = 0.1$	8.05E+02 ± 3.91E+02 ≈	1.11E+04 ± 7.25E+03 ≈	4.07E+02 ± 2.40E+02 ≈	**4.30E+02 ± 1.15E+02** −	1.13E+02 ± 4.69E+01 +	8.25E+04 ± 2.26E+04 ≈
$F_0 = 1, \sigma_0 = 0.5$	9.16E+02 ± 4.37E+02 +	9.25E+03 ± 6.15E+03 ≈	3.19E+02 ± 1.68E+02 ≈	4.86E+02 ± 1.53E+02 ≈	1.09E+02 ± 4.51E+01 ≈	8.15E+04 ± 3.10E+04 ≈
$F_0 = 1, \sigma_0 = 0.9$	9.91E+02 ± 3.75E+02 +	1.03E+04 ± 6.23E+03 ≈	4.76E+02 ± 2.55E+02 +	5.31E+02 ± 1.14E+02 ≈	1.01E+02 ± 3.61E+01 ≈	**6.47E+04 ± 2.72E+04** −
$F_0 = 5, \sigma_0 = 0.1$	7.46E+02 ± 3.58E+02 ≈	9.05E+03 ± 6.66E+03 ≈	3.15E+02 ± 1.53E+02 ≈	5.20E+02 ± 1.14E+02 ≈	1.12E+02 ± 4.61E+01 +	8.42E+04 ± 3.25E+04 ≈
$F_0 = 5, \sigma_0 = 0.5$	7.68E+02 ± 5.44E+02 ≈	9.64E+03 ± 6.13E+03 ≈	3.90E+02 ± 1.88E+02 ≈	4.90E+02 ± 1.05E+02 ≈	1.06E+02 ± 3.85E+01 ≈	7.86E+04 ± 2.35E+04 ≈
$F_0 = 5, \sigma_0 = 0.9$	9.34E+02 ± 5.57E+02 +	9.10E+03 ± 6.83E+03 ≈	3.99E+02 ± 2.30E+02 ≈	5.13E+02 ± 1.06E+02 ≈	1.05E+02 ± 4.67E+01 ≈	8.07E+04 ± 2.79E+04 ≈
	F19	F20	F21	F22	F23	F24
$F_0 = 2.5, \sigma_0 = 0.5$	1.53E+02 ± 1.04E+02	**1.38E+02 ± 5.43E+01**	2.22E+02 ± 8.15E+01	**1.10E+02 ± 1.88E+00**	**4.14E+02 ± 1.60E+01**	4.54E+02 ± 1.73E+02
$F_0 = 2.5, \sigma_0 = 0.1$	1.27E+02 ± 6.88E+01 ≈	1.60E+02 ± 6.27E+01 ≈	2.19E+02 ± 8.03E+01 ≈	1.11E+02 ± 2.06E+00 ≈	4.14E+02 ± 1.66E+01 ≈	4.03E+02 ± 1.85E+02 ≈
$F_0 = 2.5, \sigma_0 = 0.9$	1.44E+02 ± 8.84E+01 ≈	1.54E+02 ± 6.18E+01 ≈	2.28E+02 ± 7.86E+01 ≈	1.10E+02 ± 1.98E+00 ≈	4.16E+02 ± 1.48E+01 ≈	4.62E+02 ± 1.73E+02 ≈
$F_0 = 1, \sigma_0 = 0.1$	1.45E+02 ± 1.32E+02 ≈	1.82E+02 ± 6.21E+01 +	2.22E+02 ± 8.26E+01 ≈	1.11E+02 ± 1.87E+00 ≈	4.21E+02 ± 1.82E+01 ≈	**3.49E+02 ± 1.87E+02** ≈
$F_0 = 1, \sigma_0 = 0.5$	1.59E+02 ± 1.02E+02 ≈	1.65E+02 ± 5.52E+01 +	2.19E+02 ± 8.06E+01 ≈	1.11E+02 ± 2.17E+00 ≈	4.20E+02 ± 1.67E+01 ≈	3.95E+02 ± 1.85E+02 ≈
$F_0 = 1, \sigma_0 = 0.9$	1.56E+02 ± 1.09E+02 ≈	1.73E+02 ± 4.85E+01 +	1.98E+02 ± 7.99E+01 ≈	1.12E+02 ± 1.55E+00 +	4.14E+02 ± 1.65E+01 ≈	4.27E+02 ± 1.86E+02 ≈
$F_0 = 5, \sigma_0 = 0.1$	**1.25E+02 ± 7.53E+01** ≈	1.58E+02 ± 6.72E+01 ≈	2.12E+02 ± 8.16E+01 ≈	1.11E+02 ± 1.37E+00 +	4.16E+02 ± 1.46E+01 ≈	4.55E+02 ± 1.67E+02 ≈
$F_0 = 5, \sigma_0 = 0.5$	1.61E+02 ± 1.05E+02 ≈	1.45E+02 ± 6.70E+01 ≈	**2.08E+02 ± 8.11E+01** ≈	1.12E+02 ± 1.65E+00 +	4.14E+02 ± 1.67E+01 ≈	4.70E+02 ± 1.61E+02 ≈
$F_0 = 5, \sigma_0 = 0.9$	2.48E+02 ± 2.08E+02 +	1.60E+02 ± 6.43E+01 ≈	2.21E+02 ± 7.92E+01 ≈	1.11E+02 ± 1.97E+00 ≈	4.15E+02 ± 1.61E+01 ≈	4.79E+02 ± 1.44E+02 ≈
	F25	F26	F27	F28	F29	F30
$F_0 = 2.5, \sigma_0 = 0.5$	**3.83E+02 ± 1.14E+01**	2.20E+02 ± 5.55E+00	5.20E+02 ± 6.16E+00	**4.08E+02 ± 1.49E+01**	5.89E+02 ± 4.57E+01	5.04E+03 ± 8.32E+02
$F_0 = 2.5, \sigma_0 = 0.1$	3.85E+02 ± 1.45E+00 ≈	**2.18E+02 ± 2.94E+00** ≈	5.20E+02 ± 6.95E+00 ≈	4.11E+02 ± 4.11E+00 ≈	5.77E+02 ± 5.35E+01 ≈	**4.80E+03 ± 1.06E+03** ≈
$F_0 = 2.5, \sigma_0 = 0.9$	3.85E+02 ± 1.56E+00 ≈	2.20E+02 ± 5.13E+00 ≈	5.18E+02 ± 7.08E+00 ≈	4.11E+02 ± 5.39E+00 ≈	5.93E+02 ± 4.28E+01 ≈	5.32E+03 ± 1.03E+03 ≈
$F_0 = 1, \sigma_0 = 0.1$	3.85E+02 ± 1.19E+00 ≈	2.31E+02 ± 2.97E+01 ≈	5.20E+02 ± 9.26E+00 ≈	4.15E+02 ± 4.61E+00 +	5.89E+02 ± 6.62E+01 ≈	5.46E+03 ± 1.49E+03 ≈
$F_0 = 1, \sigma_0 = 0.5$	3.86E+02 ± 1.57E+00 +	2.22E+02 ± 6.19E+00 ≈	5.18E+02 ± 6.29E+00 ≈	4.14E+02 ± 8.09E+00 +	5.80E+02 ± 4.95E+01 ≈	5.85E+03 ± 1.19E+03 +
$F_0 = 1, \sigma_0 = 0.9$	3.85E+02 ± 1.22E+00 +	2.23E+02 ± 2.04E+01 ≈	5.20E+02 ± 5.73E+00 ≈	4.13E+02 ± 8.17E+00 ≈	6.01E+02 ± 5.73E+01 ≈	5.47E+03 ± 1.24E+03 ≈
$F_0 = 5, \sigma_0 = 0.1$	3.85E+02 ± 1.34E+00 ≈	2.24E+02 ± 1.63E+01 ≈	**5.16E+02 ± 5.86E+00** −	4.13E+02 ± 4.04E+00 +	5.89E+02 ± 5.81E+01 ≈	5.08E+03 ± 9.73E+02 ≈
$F_0 = 5, \sigma_0 = 0.5$	3.85E+02 ± 1.30E+00 ≈	2.21E+02 ± 4.29E+00 ≈	5.18E+02 ± 4.93E+00 ≈	4.12E+02 ± 3.88E+00 ≈	5.65E+02 ± 5.41E+01 −	5.76E+03 ± 1.08E+03 +
$F_0 = 5, \sigma_0 = 0.9$	3.85E+02 ± 1.26E+00 +	2.26E+02 ± 1.88E+01 +	5.18E+02 ± 4.26E+00 ≈	4.11E+02 ± 4.50E+00 ≈	**5.62E+02 ± 4.68E+01** −	5.34E+03 ± 1.35E+03 ≈

Table 3.7: Statistical results of different settings of F_0 and σ_0.

$F_0 = 2.5, \sigma_0 = 0.5$ vs.	$F_0 = 2.5, \sigma_0 = 0.1$	$F_0 = 2.5, \sigma_0 = 0.9$	$F_0 = 1, \sigma_0 = 0.1$	$F_0 = 1, \sigma_0 = 0.5$	$F_0 = 1, \sigma_0 = 0.9$	$F_0 = 5, \sigma_0 = 0.1$	$F_0 = 5, \sigma_0 = 0.5$	$F_0 = 5, \sigma_0 = 0.9$
+/≈/-	3/26/1	1/28/1	8/21/1	8/22/0	9/20/1	4/25/1	5/23/2	7/19/4

Table 3.8: Experimental results of eight algorithms on FMSWP.

Algorithm	Mean	Std	Best	Worst
TDSD	3.93	4.97	0.00	12.03
WOA	30.36	1.67	26.87	35.43
DE	31.47	20.05	21.94	114.43
CGSA-P	24.43	1.51	18.51	26
BSO	13.98	5.88	0.99	21.11
EPSDE	9.86	5.03	0.25	16.92
SaDE	7.41	4.82	0.12	15.69
jDE	5.81	4.65	0.06	14.43

Table 3.9: Experimental results of eight algorithms on SSRPPCD.

Algorithm	Mean	Std	Best	Worst
TDSD	1.023629	0.077497	0.870792	1.156039
WOA	1.920357	0.230182	1.440039	2.293614
DE	3.310202	0.405792	2.659947	3.925028
CGSA-P	1.347604	0.145627	1.090817	1.609239
BSO	1.483097	0.107354	1.236537	1.680372
EPSDE	1.150966	0.107714	0.837685	1.282661
SaDE	1.223077	0.106451	0.871849	1.398925
jDE	1.376812	0.093742	1.177771	1.538283

Table 3.10: Experimental results of eight algorithms on NLSTRP.

Algorithm	Mean	Std	Best	Worst
TDSD	13.93	0.17	13.77	14.33
WOA	15.23	2.22	13.87	21.00
DE	49.92	24.61	22.01	107.11
CGSA-P	15.95	2.02	13.86	21.01
BSO	16.50	2.32	14.34	21.00
EPSDE	21.31	1.21	15.42	22.96
SaDE	15.53	2.62	13.79	20.96
jDE	16.00	2.70	13.95	20.98

Table 3.11: Experimental results of TDSD versus four champion algorithms on CEC2017 benchmark functions.

Fun.	TDSD Mean ± Std Dev	EBOwithCMAR Mean ±Std Dev	LSHADE-cnEpSin Mean ±Std Dev	LSHADE-SPACMA Mean ±Std Dev	SHADE Mean ±Std Dev
F1	1.7446E+03 ± 9.0697E+02	**0.0000E+00 ± 0.0000E+00** −	**0.0000E+00 ± 0.0000E+00** +	**0.0000E+00 ± 0.0000E+00** −	**0.0000E+00 ± 0.0000E+00** −
F2	1.3076E+12 ± 2.3022E+12	**0.0000E+00 ± 0.0000E+00** −	**0.0000E+00 ± 0.0000E+00** −	**0.0000E+00 ± 0.0000E+00** −	2.7407E+02 ± 1.0392E+03 −
F3	4.0435E+04 ± 9.5356E+03	**0.0000E+00 ± 0.0000E+00** −	**0.0000E+00 ± 0.0000E+00** −	**0.0000E+00 ± 0.0000E+00** −	**0.0000E+00 ± 0.0000E+00** −
F4	**2.0235E+01 ± 2.0113E+01**	5.5531E+01 ± 1.4687E+01 +	4.2950E+01 ± 2.5689E+00 +	5.8562E+01 ± 1.0556E-14 +	5.6610E+01 ± 1.0692E+01 +
F5	7.9872E+01 ± 1.1533E+01	8.7675E+00 ± 4.2044E+00 −	1.1364E+01 ± 2.9291E+00 −	**3.7881E+00 ± 2.4059E+00** −	1.4578E+01 ± 2.3593E+00 −
F6	3.2773E+00 ± 6.7748E-01	**0.0000E+00 ± 0.0000E+00** −	1.4828E-08 ± 4.1845E-08 −	**0.0000E+00 ± 0.0000E+00** −	3.7220E-05 ± 4.7525E-05 −
F7	1.3181E+02 ± 1.3996E+01	3.7875E+01 ± 4.4565E+00 −	4.2302E+01 ± 2.3960E+00 −	**3.3751E+01 ± 9.7846E-01** −	4.4090E+01 ± 2.3697E+00 −
F8	8.3313E+01 ± 8.1652E+00	9.8595E+00 ± 3.6964E+00 −	1.3063E+01 ± 2.1663E+00 −	**3.4509E+00 ± 1.7838E+00** −	1.5361E+01 ± 2.9416E+00 −
F9	1.7129E+03 ± 3.4730E+02	**0.0000E+00 ± 0.0000E+00** −	**0.0000E+00 ± 0.0000E+00** −	**0.0000E+00 ± 0.0000E+00** −	5.4162E-02 ± 1.2276E-01 −
F10	2.1866E+03 ± 2.2587E+02	1.4668E+03 ± 1.7212E+02 −	1.4678E+03 ± 2.1985E+02 −	**1.3634E+03 ± 2.2745E+02** −	1.7004E+03 ± 1.9089E+02 −
F11	6.8555E+01 ± 2.4028E+01	1.2784E+01 ± 2.2626E+01 −	1.6263E+01 ± 2.1597E+01 −	**1.2344E+01 ± 1.9931E+01** −	4.3102E+01 ± 3.2603E+01 −
F12	2.9161E+05 ± 1.7235E+05	3.9285E+02 ± 2.3306E+02 −	**3.3086E+02 ± 2.0912E+02** −	5.1785E+02 ± 3.1118E+02 −	1.3650E+03 ± 3.8762E+02 −
F13	6.9559E+02 ± 3.3086E+02	1.4124E+01 ± 5.8471E+00 −	1.3886E+01 ± 6.8736E+00 −	**1.3545E+01 ± 5.9654E+00** −	3.8598E+01 ± 1.1942E+01 −
F14	8.4226E+03 ± 6.0815E+03	2.2845E+01 ± 1.3652E+00 −	2.2266E+01 ± 1.0951E+00 −	**2.2158E+01 ± 4.0819E+00** −	2.9245E+01 ± 4.9525E+00 −
F15	3.5640E+02 ± 2.2676E+02	3.4983E+00 ± 1.4404E+00 −	**2.4667E+00 ± 1.3895E+00** −	5.2391E+00 ± 2.5540E+00 −	2.3941E+01 ± 2.3627E+01 −
F16	4.8592E+02 ± 1.1375E+02	6.8201E+01 ± 7.9343E+01 −	**2.8105E+01 ± 3.6815E+01** −	4.6016E+01 ± 5.4802E+01 −	2.8724E+02 ± 1.4021E+02 −
F17	9.3752E+01 ± 4.0179E+01	2.8784E+01 ± 6.1390E+00 −	**2.8581E+01 ± 6.3855E+00** −	3.0778E+01 ± 1.0007E+01 −	6.1379E+01 ± 3.7815E+01 −
F18	8.1465E+04 ± 3.6777E+04	2.2392E+01 ± 1.4821E+00 −	**2.0217E+01 ± 3.6098E+00** −	2.2985E+01 ± 1.9390E+00 −	6.0993E+01 ± 2.9753E+01 −
F19	1.5314E+02 ± 1.0351E+02	8.6930E+00 ± 2.2489E+00 −	**5.0324E+00 ± 1.5898E+00** −	9.8727E+00 ± 2.3712E+00 −	1.3824E+01 ± 7.5571E+00 −
F20	1.3811E+02 ± 5.4266E+01	3.7966E+01 ± 6.4979E+00 −	**3.3111E+01 ± 1.2658E+01** −	1.0058E+02 ± 6.1952E+01 −	8.6271E+01 ± 5.8196E+01 −
F21	2.2212E+02 ± 8.1525E+01	2.0919E+02 ± 4.5825E+00 ≈	2.1221E+02 ± 2.5305E+00 ≈	**2.0712E+02 ± 3.5690E+00** ≈	2.1685E+02 ± 2.5003E+00 ≈
F22	1.1038E+02 ± 1.8762E+00	**1.0000E+02 ± 0.0000E+00** −	**1.0000E+02 ± 0.0000E+00** −	1.0000E+02 ± 2.0465E-13 −	1.0000E+02 ± 1.9574E-13 −
F23	4.1439E+02 ± 1.6030E+01	3.5622E+02 ± 3.6904E+00 −	**3.5569E+02 ± 3.9340E+00** −	3.5571E+02 ± 3.5495E+00 −	3.6540E+02 ± 5.2491E+00 −
F24	4.5389E+02 ± 1.7290E+02	4.2853E+02 ± 2.7450E+00 −	**4.2772E+02 ± 2.3446E+00** −	4.2917E+02 ± 2.6616E+00 −	4.3673E+02 ± 4.1509E+00 −
F25	**3.8263E+02 ± 1.1444E+01**	3.8674E+02 ± 1.3890E-02 +	3.8668E+02 ± 7.3605E-03 +	3.8670E+02 ± 7.5392E-03 +	3.8684E+02 ± 7.7924E-02 +
F26	**2.2014E+02 ± 5.5468E+00**	7.4035E+02 ± 3.3557E+02 +	9.5167E+02 ± 4.0428E+01 +	9.6700E+02 ± 4.5753E+01 +	1.0947E+03 ± 5.4342E+01 +
F27	5.2013E+02 ± 6.1588E+00	5.0532E+02 ± 4.0639E+00 −	**5.0503E+02 ± 5.7341E+00** −	5.0666E+02 ± 4.6646E+00 −	5.0556E+02 ± 6.3681E+00 −
F28	4.0840E+02 ± 1.4862E+01	**3.0000E+02 ± 1.1499E-09** −	3.2932E+02 ± 4.9535E+01 −	3.2932E+02 ± 4.9535E+01 −	3.3683E+02 ± 5.3710E+01 −
F29	5.8901E+02 ± 4.5694E+01	**4.3370E+02 ± 1.2496E+01** −	4.3558E+02 ± 9.1763E+00 −	4.4520E+02 ± 1.5303E+01 −	4.6247E+02 ± 4.2134E+01 −
F30	5.0400E+03 ± 8.3171E+02	**1.9959E+03 ± 6.2734E+01** −	2.0042E+03 ± 6.7403E+01 −	2.0078E+03 ± 7.3730E+01 −	2.0952E+03 ± 1.5166E+02 −
+/≈/-		3/ 1 /26	3/ 1 /26	3/ 1 /26	3/ 1 /26

Table 3.12: Experimental results of five algorithms on FMSWP.

Algorithm	Mean	Std	Best	Worst
TDSD	3.93	4.97	0.00	12.03
LSHADE-cnEpSin	1.09	3.29	0.00	11.76
LSHADE-SPACMA	12.66	5.46	0.00	20.17
EBOwithCMAR	2.19	4.49	0.00	11.89
SHADE	1.82	2.60	0.00	8.79

Table 3.13: Experimental results of five algorithms on SSRPPCD.

Algorithm	Mean	Std	Best	Worst
TDSD	1.023629	0.077497	0.870792	1.156039
LSHADE-cnEpSin	1.022424	0.057046	0.901506	1.204511
LSHADE-SPACMA	1.115314	0.100054	0.821232	1.278417
EBOwithCMAR	0.632786	0.120243	0.500000	0.872078
SHADE	1.225618	0.097473	1.034590	1.413917

Table 3.14: Experimental results of five algorithms on NLSTRP.

Algorithm	Mean	Std	Best	Worst
TDSD	13.93	0.17	13.77	14.33
LSHADE-cnEpSin	18.45	3.25	13.77	21.08
LSHADE-SPACMA	19.23	3.05	13.77	21.54
EBOwithCMAR	20.01	2.24	13.77	21.08
SHADE	14.28	0.20	13.85	14.59

Chapter 4

An improved gravitational search algorithm based on negative correlation(NC-GSA)

4.1 Introduction

The study on intelligence computational algorithms has been thriving since 1980s [75, 76]. As one of the greatest findings for solving optimization problems, it has experienced a few transitions inevitably. Especially, the research boom of intelligence computation transiting from individual-based algorithms to population-based algorithms has the profound influence among the scientific community [77] on the solid foundation provided by experimental results [4,22,23,78–80], which made population-based algorithms the most extensively studied intelligence computational algorithms ever since. Population-based algorithms are typified by genetic algorithms [81], evolution strategies [82], evolutionary programming [83], differential evolution [12, 84], particle swarm optimization [85], ant colony optimization [86], brain storm optimization [20,87,88], etc. There is no coincidence in the superior performance of population-based algorithms. With the participation of the population, the time complexity of the searching algorithm is diminished from exponential to polynomial level [89]. On the other hand, it shows more expedient capacity for parallelization to enhance the efficiency of calculation effectively [90].

Gravitational search algorithm (GSA), developed by Rashedi et al. in 2009 [31] with the inspiration of the law of gravity and mass interactions, is one of the most effective heuristic optimization algorithms based on population [91,92]. GSA is based on the concept that every particle attracting every other particle is ubiquitous in the universe with Newton's gravitational force as one of the four fundamental interactions in nature [31,93]. With moving towards one another according to the dynamic law, the particles would form to be in a balance state eventually. As shown in Fig. 4.1, GSA calculates inertial mass of each particle by the fitness, which accounts for the quality of each particle's position. With the update of functional fitness, the inertia mass of the particle is modified correspondingly throughout searching iterations. Particles with heavier inertia mass hold larger effective attraction radius for other particles, hence a greater intensity of gravitational force leading towards the optimal position.

Over the last few decades, GSA has been applied as an effective technique for solving complex and difficulty optimization problems in practice owing to its searching characteristic [2, 18, 67, 94–97]. Particles commune information through the gravitational force, establishing an accessible space with transparency that allows particles observe the others. Moreover, without the demand of recording the optimal solution, GSA operates with low memory cost and low time consumption, while some argued that the lack of memory property is not necessarily a strong suit for optimization problems [98]. In spite of those upsides GSA possesses, the rapid searching speed could cause premature convergence [99–102]. The risk of stagnation that the algorithm is immersed in a local optimum discourages the optimization accuracy.

Likewise, the practical solution to perform a greater compromise between exploration and exploitation is the research emphasis in terms of optimization algorithms for the sake of achieving a better performance. Empirically, an algorithm enhances exploration in the initial searching stage so as to avoid the stagnation, and the ex-

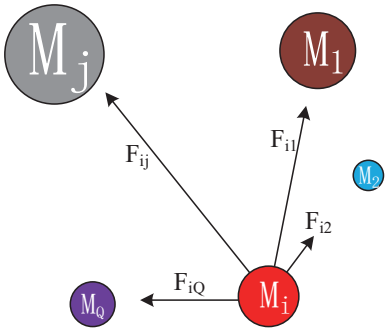

Figure 4.1: The gravitational phenomena used in GSA.

ploitation is expected to be intensified with the lapse of time. The study in [103] reveals that the performances between optimization algorithms is equivalent as all functions realize mutual compensation in the No Free Lunch Theorem applied for the comparison between variant optimization algorithms specifically, which can be formulated as follows:

$$\sum_{\forall \phi} P(y|\phi, m, A) = \sum_{\forall \phi} P(y|\phi, m, B), \tag{4.1}$$

where A, B stand for two separate optimization algorithms, and $m = 1, 2, ..., N$ is the number of iteration for searching process. The hybridization of different search mechanisms from two optimization algorithms absorbs advanced solutions respectively to reach a more effective solution [104].

On the other hand, the negatively correlated search (NCS) [15] is an improved stochastic optimization inspired by human cooperation. NCS applies a parallel search pattern of multi-agent for modeling based on probability distribution to promote the diversity of search behaviors in an information sharing and cooperation way, which establishes a novel approach for cooperation among individuals. NCS is constituted of multiple search processes that develops into a new individual with the influence of the search results from itself and others. Generally, the relationship among search processes is negatively correlated, which means that the framework of the algorithm

tends to form a situation where each search process is distributed in different regions and avoid duplication to prevention of local optimum.

NCS is highlighted by the consideration of both the fitness of individuals as quality measurement and individual differences. In a steep solution space, the global optimal solution where an individual with poor fitness comprising a piece of excellent gene, in all probability, might be missed or eliminated in the early stage. As a solution for preserving misinterpreted solution, NCS selects candidate solutions according to search behavior (probability distribution of new individuals) rather than distance between individuals. Based on the information of population distribution and transformation of population distribution, NCS leads to sub population division to simplify the solution space of the original problem through information transmission among individuals. Compared with the traditional evolutionary algorithm, NCS has the advantages of high efficiency and the characteristics of robustness.

In this paper, an improved gravitational search algorithm based on negative correlation is proposed with the hypothesis that hybridization of algorithms improves optimization performances and efficiency. For the ease of consistent description, the novel GSA based on negative correlation learning is abbreviated to NC-GSA. As population-based search methods, GSA and NCS exchange information among individuals through fitness and search area, respectively. While GSA conducts exploitation in the search space, NCS fulfills exploration by encouraging discrepant search behaviors to increase the optimization accuracy. By doing so, NC-GSA is expected to well balance the exploitation and exploration, thus possessing a significant better or competitive performance in comparison with its component algorithms. Experimental results based on thirty benchmark optimization functions demonstrate the robustness of NC-GSA, and also indicate that NC-GSA performs better than GSA and NCS in terms of solution quality and convergence speed. Additionally, a contrast

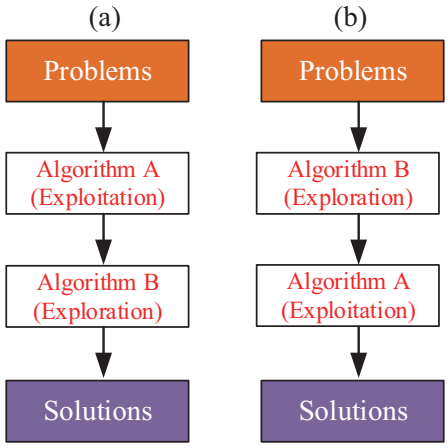

Figure 4.2: Generic architectures of hybrid algorithms.

experiment is also conducted to verify the influence of the implementation sequence of component algorithms on the performance, and the results suggest that NCS should be implemented prior to GSA, and the reason seems to be that NCS diversifies the distribution of solutions, upon which GSA utilizes the gravitational search to obtain a fast convergence.

The remainder of this paper is organized as follows. In Section II, a novel GSA based on negative correlation learning is introduced in detail. Section III summarises the experimental results on benchmark function suit. Finally, we give some general remarks and present follow-up work with our insights and perspectives in Section IV.

4.2 Negative Correlation Gravitational Search Algorithm

The general architectures of hybrid algorithms are illustrated in Fig. 4.2. Two kinds of implementation sequences can be realized. Fig. 4.2(a) depicts that the algorithm A which mainly takes role of exploiting is carried out prior to the algorithm B which is used to refine the solutions generated by the algorithm A and thus performing the exploration in the search landscape to avoid local minima trapping problem. Fig. 4.2(b) shows the opposite circumstances. Based on this general implementation

framework, in this paper we for the first time propose a hybrid algorithm by combining GSA and NCS sequentially. It is worth pointing out that the method of combining different algorithms in a parallel (or distributed) manner [90] is not discussed in this paper, but would be studied in our future work. The main procedures of the proposed algorithms are illustrated in Fig. 4.3, where Fig. 4.3(a) depicts NC-GSA and Fig. 4.3(b) shows the main processes of GSA-NC.

In this section, we render the foremost strategies of NC-GSA comprehensively. It should be noted that GSA-NC is implemented in a reversed sequence of NC-GSA. As a population-based algorithm, NC-GSA is carried out in a solution space with D-dimension where N individuals are included. The position of the ith individual is defined as:

$$X_i = (x_i^1, x_i^2, \cdots, x_i^k, \cdots, x_i^n) \ for \ i = 1, 2, \cdots, N, \tag{4.2}$$

where x_i^k presents the position of the i-th individual in the kth dimension. The measurement of each individual's quality is pursuant to the value of the objective optimization function, denoted as $f(x)$.

As shown in Fig. 4.1, the gravitational force $F_{ij}^d(t)$ acting on the ith individual from the jth individual at the tth iteration is represented as Eq. (4.3):

$$F_{ij}^d(t) = G(t)\frac{M_j(t)M_i(t)}{R_{ij}(t) + \varepsilon}(x_j^d(t) - x_i^d(t)), \tag{4.3}$$

where $G(t)$ denotes the gravitational constant, M_i and M_j are masses of individuals, ε is a constant to avoid NULL values, and $R_{ij}(t)$ represents the Euclidean distance between two individuals:

$$R_{ij}(t) = \| \ x_i(t), x_j(t) \ \|_2 \ . \tag{4.4}$$

It has been systematically studied the effects of the gravitational constant $G(t)$

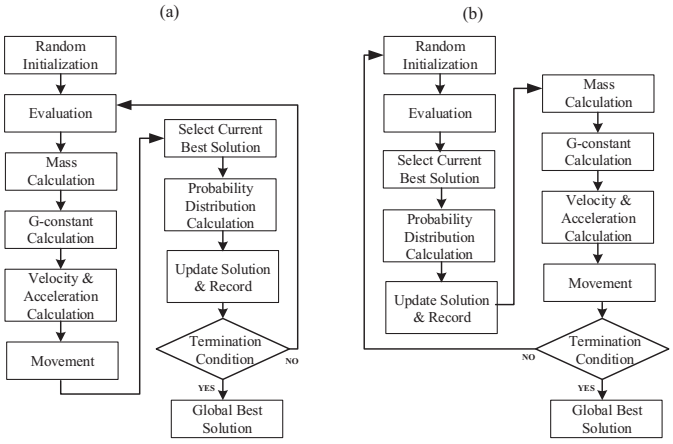

Figure 4.3: Flowchart of (a) NC-GSA and (b) GSA-NC.

[105] and various developed ones are proposed. To independently observe the effects of the combination of such two algorithms, this work only considers the original realization of both algorithms. At the beginning of iterations $G(t)$ is initialized to $G(0)$ and then reduced with time to control the search accuracy:

$$G(t) = G_0 e^{-\alpha t/T_{max}}, \tag{4.5}$$

where α is a shrinking parameter. Under this condition, the total gravitational force on the ith individual is expressed by:

$$F_i^d(t) = \sum_{j=1, j\neq i}^{Kbest} rand_j F_{ij}^d(t), \tag{4.6}$$

where $Kbest$ is the set of first K individuals with bigger masses, and $rand_j$ is a random number in the interval [0,1]. The acceleration $a_i^d(t)$ of the individual i is shown as:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)}, \tag{4.7}$$

where $M_i(t)$ is the map of fitness calculated as in the following:

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)}, \tag{4.8}$$

and

$$m_i = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)}, \tag{4.9}$$

where $best(t)$ is the best fitness of all individuals, $worst(t)$ is the worst fitness of all individuals, and $fit_i(t)$ denotes the fitness of individual M_i by calculating the objective functions.

The new velocity of an individual is considered as a fraction of its current velocity added to its acceleration. Thus the position and the velocity of the ith individual is expressed as follows:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t), \tag{4.10}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1). \tag{4.11}$$

After the movement of all individuals, N best current solutions are selected to undergo the negative correlation search. With the contribution of negative correlation mechanism, the individuals with the tendency towards negative correlational cooperation are distributed in different searching regions which are reflected on the probability distribution for calculation. Eq. (4.12) and Eq. (5.5) pose the Bhattacharyya distance [106] for continuous probability distribution and discrete probability distribution, respectively.

$$D_B(p_i,\ p_j) = -\ln\left(\int \sqrt{p_i(x)p_j(x)}dx\right), \tag{4.12}$$

$$D_B(p_i,\ p_j) = -\ln\left(\sum_{x\in X}\sqrt{p_i(x)p_j(x)}\right), \tag{4.13}$$

where p_i and p_j denote the probability density functions.

Assuming the objective function $f(x)$ exceeding the minimum in a D-dimensional continuous optimization problem, NC-GSA adopts the Gaussian mutation operator [107] as the search operator for individual variation. Derived from the current existing solution x_i, a new solution x_i' is engendered with the Gaussian mutation operator, as given in Eq. (4.14):

$$x'_{id} = x_{id} + N(0,\ \sigma_i), \tag{4.14}$$

where x_{id} is the dth element of x_i, $N(0,\ \sigma_i)$ is a Gaussian random variable with the mean value of 0.0 and the standard deviation of σ_i. The standard deviation σ_i of all individuals are initialized to an identical value, yet with further search, they diverge by applying the 1/5 successful rule [107], which can be formulized as:

$$\sigma_i = \begin{cases} \frac{\sigma_i}{r}, & if\ \frac{t}{epoch} > 0.2 \\ \sigma_i * r, & if\ \frac{t}{epoch} < 0.2 \\ \sigma_i, & if\ \frac{t}{epoch} = 0.2, \end{cases} \tag{4.15}$$

where r is a parameter of the 1/5 successful rule that is set beneath 1.0 by suggestion, c is the number of replacement by the superposition of *epoches*. Larger c means that the better solution can be detected in swiftness. On the contrary, the search is better off slackened.

Furthermore, the rule of correspondence between each individual and the pertinent probability distribution establishes procedure of substitution for individuals to determine the next generation during each iteration. The negative correlation amongst probability distributions each corresponding to a particular individual is presented

as:

$$Corr(p_i) = \min_j \{ D_B(p_i,\ p_j) \mid j \neq i \ \}, \tag{4.16}$$

where p_j denotes other probability distribution from other individuals.The individuals with larger $Corr(p_i)$ are out of the realm of possibility of generating similar solution with others. Thus, a solution with small $f(x)$ and large $Corr(p_i)$ is preferred for solving minimization problems. However, as the balance between exploration and exploitation during search, the proportional relation between $f(x)$ and $Corr(p_i)$ is delicate as well. In order to discard individual in fairness, a heuristic rule with a certain randomicity is adopted, as shown in Eq. (4.17) and Eq. (5.7).

$$\begin{cases} discard\ x_i, & \textbf{if } \frac{f(x_i')}{Corr(p_i')} < \lambda \\ discard\ x_i', & \textbf{otherwise}, \end{cases} \tag{4.17}$$

where x_i stands for the current solution of the ith individual, and x_i' is the new solution originated from the current one.

$$\lambda_t = N(1.0,\ 0.1 - 0.1 * \frac{t}{T_{max}}), \tag{4.18}$$

where $\lambda_t \in (0, -\infty)$ is a random parameter varied by time dominating the trade-off between solution quality and probability distributions, which affects the performance of algorithm in a large extent, T_{max} is the total number of iterations for execution defined by user. In initial stages, NC-GSA explores the solution space with large step size, whereas small step size while entering the final phase of search as a result of the tendency towards convergence. Given the consideration above, the value of λ_t is employed under Gaussian distribution with the mean value of 1.0 on the random sampling method during each iteration. The standard deviation of the distribution is initialized to 0.1, gradually decay to 0.0 in the final stage with the searching process.

After the negative correlation search, the obtained best optimal solutions are

Table 4.1: The objective evaluations of different methods

	NC-GSA		GSA-NC		GSA		NCS	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	6.27E+10	3.12E+09	1.64E+11	1.28E+10	6.48E+10	5.64E+09	1.07E+11	1.48E+10
F3	8.93E+04	2.90E+03	7.11E+13	7.22E+13	8.93E+04	3.36E+03	2.03E+05	3.64E+04
F4	2.04E+04	1.29E+03	3.12E+04	4.26E+03	1.98E+04	2.64E+03	3.85E+04	1.10E+04
F5	9.39E+02	1.60E+01	1.21E+03	4.87E+01	9.56E+02	1.49E+01	1.09E+03	4.84E+01
F6	7.04E+02	2.45E+00	7.14E+02	8.73E+00	7.02E+02	4.22E+00	7.16E+02	1.03E+01
F7	1.48E+03	1.71E+01	3.49E+03	2.25E+02	1.49E+03	2.28E+01	3.08E+03	3.16E+02
F8	1.17E+03	1.68E+01	1.42E+03	3.39E+01	1.18E+03	1.65E+01	1.32E+03	4.03E+01
F9	1.29E+04	1.09E+03	2.94E+04	3.06E+03	1.21E+04	9.81E+02	2.53E+04	3.04E+03
F10	9.29E+03	3.39E+02	9.61E+03	4.82E+02	9.38E+03	3.36E+02	9.59E+03	4.32E+02
F11	1.30E+04	2.08E+03	5.12E+08	6.89E+08	1.01E+04	1.86E+03	2.49E+04	6.55E+03
F12	1.58E+10	7.92E+08	2.50E+10	3.15E+09	1.68E+10	3.61E+09	2.31E+10	4.87E+09
F13	1.80E+10	3.78E+08	1.26E+10	3.44E+09	1.91E+10	4.06E+09	2.08E+10	7.74E+09
F14	4.98E+06	3.96E+05	1.37E+08	1.17E+08	1.31E+07	9.11E+06	1.34E+07	7.01E+06
F15	1.11E+09	3.11E+08	5.83E+09	2.07E+09	1.13E+09	4.51E+08	4.66E+09	2.43E+09
F16	8.13E+03	1.28E+02	4.55E+04	9.61E+03	7.48E+03	1.02E+03	6.68E+03	1.07E+03
F17	5.74E+03	4.35E+02	3.87E+05	1.95E+05	6.89E+03	3.28E+03	9.15E+03	9.66E+03
F18	1.56E+08	8.38E+07	9.41E+08	7.04E+08	1.77E+08	1.01E+08	2.52E+08	2.34E+08
F19	1.27E+09	3.32E+08	5.34E+09	2.15E+09	1.33E+09	4.88E+08	4.23E+09	2.01E+09
F20	3.41E+03	1.14E+02	3.38E+03	1.75E+02	3.37E+03	1.46E+02	3.38E+03	1.82E+02
F21	2.82E+03	1.77E+01	2.88E+03	3.64E+01	2.83E+03	4.52E+01	2.84E+03	6.21E+01
F22	9.99E+03	3.91E+02	1.14E+04	4.68E+02	1.01E+04	6.69E+02	1.10E+04	5.21E+02
F23	3.63E+03	4.63E-13	4.54E+03	1.20E+02	3.81E+03	1.42E+02	3.72E+03	1.47E+02
F24	3.73E+03	1.01E+00	4.71E+03	1.47E+02	4.16E+03	2.46E+02	4.08E+03	2.02E+02
F25	5.65E+03	3.08E+02	1.22E+04	1.67E+03	5.84E+03	3.61E+02	1.57E+04	2.71E+03
F26	1.21E+04	3.09E+02	2.52E+04	2.37E+03	1.23E+04	7.27E+02	1.48E+04	1.57E+03
F27	4.70E+03	1.31E+01	5.35E+03	3.89E+02	5.24E+03	5.21E+02	5.04E+03	3.57E+02
F28	7.97E+03	3.27E+02	9.91E+03	1.11E+03	8.07E+03	4.34E+02	1.16E+04	1.64E+03
F29	1.02E+04	4.78E+02	4.17E+05	2.94E+05	9.94E+03	2.35E+03	1.72E+04	1.21E+04
F30	4.29E+08	6.49E+05	7.82E+09	1.33E+09	2.39E+09	1.03E+09	2.22E+09	1.06E+09

recorded and enter the next iteration of gravitational search until the termination condition is fulfilled.

4.3 Experimental evaluation

In this section, the experimental results of the proposed algorithms examined on IEEE Congress on Evolutionary Computation 2017 (CEC2017) benchmark functions as test functions for optimization are presented. With brief introduction of these benchmark functions for comprehensive understanding in mathematics, they are classified into four categories: functions $F1$-$F3$ are unimodal functions; functions $F4$-$F10$ are sim-

ple multimodal functions; functions $F11$-$F20$ are hybrid functions; functions $F21$-$F30$ are composition functions. To be noted, the experiment rules out F2 function on account of its unstable behavior on high dimensions. On the ground of consistency and impartiality of study, the population size of individuals is set as $popsize = 100$ with limited generations $T_{max} = 3000$ on a search space of 30-dimensions. The independent running number is set as 30 to make a statistical analysis. The experiment is conducted with Matlab on a Personal PC with Intel i5 CPU and 8GB memory.

As the presentation of results, the strategy of the participation of negative correlation is superior than the original algorithms of as listed in Table 4.1. In the 29 test benchmark functions, apart from function $F4$, $F6$, $F11$, $F20$, $F29$ which are leaded by the traditional GSA and F11 leaded by NCS, NC-GSA prominent its superiority on the performance of the remaining optimization functions. By comparing the results of NC-GSA and GSA-NC, we confirm the theory of gravitational search algorithm conducting exploitation in the search space at the earlier stage while negative correlation fulfills exploration by encouraging discrepant search behaviors to increase the optimization accuracy.

To give more insights into the results, the box-and-whisker plot of F30 is plotted to show the distribution of the final 30 solutions of each algorithm, as illustrated in Fig. 4.4. From it, we find that NC-GSA can find better and more stabler solutions in comparison with its competitors. In addition, Fig. 4.5 depicts the convergence graph of all compared algorithms for F30. Similar results can also be observed for the other benchmark functions. It is clear that all algorithms converge very fast, and NC-GSA possess the fastest convergence speed. The reason is that the negative search first diverges the search regions, thus making the population diversity be a high level. Along with the gravitational search, the algorithm quickly converges to the promising areas, and thereafter finding better solutions.

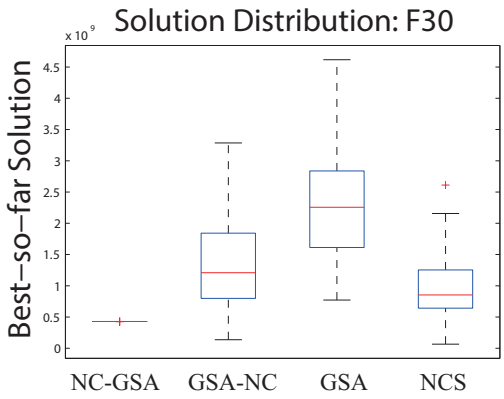

Figure 4.4: Box-and-whisker plot of the obtained 30 final solutions by all compared algorithms for the benchmark function F30.

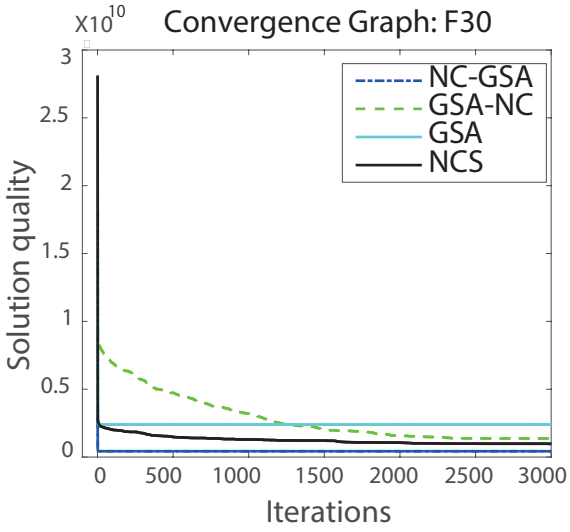

Figure 4.5: Convergence graph of all compared algorithms for the benchmark function F30.

Finally, two non-parametric statistical test methods including the Friedman test and Wilcoxon test are employed to detect the existence of significant difference among algorithms respectively. The procedures of Wilcoxon rank-sum test for detecting significant differences between average values of 30 independent runs for two compared algorithms are implemented as in the following. First, we calculate the differences D_i between the two compared algorithms on each problem. Here, we use NC-GSA as the control algorithm. Then These differences D_i are ranked by absolute values, where R^+ is the sum rank for the problems in which the NC-GSA performs better than its

Table 4.2: Results obtained by the Friedman and Wilcoxon tests for the control algorithm NC-GSA.

	Ranking	R^+	R^-	Exact P-value	Asymptotic P-value
NC-GSA	1.3966	–	–	–	–
GSA-NC	3.7069	408.0	27.0	4.698E-6	0.000036
GSA	1.9483	315.0	91.0	0.009536	0.009562
NCS	2.9483	421.0	14.0	4.098E-7	0.00001

competitor and R^- is for the opposite. If $D_i = 0$, the rank of it will be divided evenly among the sums, this is,

$$R^+ = \sum_{D_i>0} rank(D_i) + 0.5 \sum_{D_i=0} rank(D_i), \tag{4.19}$$

$$R^- = \sum_{D_i<0} rank(D_i) + 0.5 \sum_{D_i=0} rank(D_i), \tag{4.20}$$

where $T = min(R^+, R^-)$. According the value of T, p-values are calculated and then we judge to reject the null hypothesis of quality of average values of p-values is less than 0.05.

From Table 4.2, we can find that NC-GSA has the smallest ranking value 1.3966 based on the Friedman test which suggests that NC-GSA is statistically better than its competitor algorithms. Additionally, all p-values obtained by Wilcoxon rank-sum test are smaller than 0.05, indicating that NC-GSA is significant better than GSA-NC, GSA, and NCS in terms of average quality of solutions.

4.4 Conclusions

In this paper, an improved gravitational search algorithm is proposed by incorporating with negative correlation. As an experimental study, traditional gravitational search algorithm and concrete implementation of negatively correlated search are taken into consideration. The algorithm developed the global exploration property of negatively correlated search and the local exploitation property of gravitational search. Sev-

eral experiments conducted on CEC2017 benchmark function suit indicated that the performance of the novel algorithm is more effective than both gravitational search algorithm and negatively correlated search algorithm. In the follow-up work, a solution of determination of the balance between individual fitness and individual distance will be brought forward. Furthermore, the study and analysis will be more focused on the meaning of hybridism in the perspective of population diversity and robustness for other practical problems, such as complex networks [108–110] and Internet of vehicles [111–113].

Chapter 5

Negative correlation learning enhanced search behavior in BSA(BSANCS)

5.1 Introduction

In many areas, optimization is a very important research subject. The purpose of optimization is to find the best value in complex environments. Optimization usually aims to find the best point in the search space, that is, finding the global minimum of objective function. However, the objective function may be non-linear, complex or non-differentiable. When the objective function is non-linear, complex or non-differentiable, evolutionary algorithm (EA) [1, 2] is usually used to solve the global optimal problem. EA is a mature global optimization method with high robustness and wide applicability. EAs can solve different types of problems. Therefore, EAs have be used by various industries to solve the global optimum problem [3, 4]. In the family of EAs, many algorithms exhibit great search abilities. For example, the particle swarm optimization algorithm (PSO) [5] simulates the social behavior of creatures, such as bird flock or fish schooling. The ant colony optimization algorithm (ACO) [6] is stochastic search algorithm that simulates the process of natural ants seeking source of food. The artificial bee colony algorithm (ABC) [7–9] is an optimization algorithm that simulates the foraging behavior of honey bees. The genetic

algorithm (GA) [10] is a metaheuristic algorithm inspired by the process of natural selection strategy. The differential evolution algorithm (DE) [11–14] is a method to optimize problems by iteratively trying to improve a candidate solution with regard to a given measure of quality.

Backtracking search optimization algorithm (BSA) [16] is a new EA that to solve global optimization problems. BSA is based on basic genetic rules. Therefore,it includes five processes: initialization, selection-I, mutation, crossover and selection-II. In initialization process, the population (pop) is randomly generated in the search space. In selection-I stage, BSA possesses a memory to store historical population ($pop_{historical}$) that to be used for calculating the search direction. In mutation process, BSA generates a mutant population ($pop_{mutation}$) using Eq. (5.1):

$$pop_{mutation} = pop + F \cdot (pop_{historical} - pop), \tag{5.1}$$

where F is a parameter that controls the amplitude of the search direction matrix ($pop_{historical}$). The F is a standard normal distribution, which is formulated as $F = 3 \cdot randn$.

Then, in crossover stage, trial population ($offsprings$) is generated by crossover process using Eq. (5.2):

$$offsprings = pop + (map \cdot F) \cdot (pop_{historical} - pop), \tag{5.2}$$

where map is binary integer value matrix of size $Popsize \cdot Dim$ ($Popsize$ is size of population. Dim is population dimension) that guides crossover operation. Finally, in selection-II process, population (pop) is updated by trial population ($offsprings$) with the better fitness value. If the fitness value of the best population (P_{best}) is better than the currently global minimum value, the global minimum population is updated to be P_{best}, the global minimum value is updated to the fitness value of P_{best}.

Due to BSA only evaluates the individuals based on the fitness, individuals with promising exploration prospects may be discarded when updating population. Therefore, we should take these individuals into account when executing selection operation. Now, many ways of optimizing algorithms are made by hybridizing two or more algorithms [17–20]. Inspired by this, we combine BSA with negatively correlated search (NCS) [15] to enhance search behavior backtracking search mechanism. We call this hybridized algorithm BSANCS. The NCS models the search behavior of individual search process as a probability distribution, and calculates the correlation among different individual via a variable called Bhattacharyya distance. The NCS maintains search behaviors by encouraging differences among the probability distributions. The main property of NCS is that it selects individual with better fitness value and larger probability distributions into the next generation. NCS can population and promote negatively correlated search by sharing information to improve exploration abilities of BSA.

In this paper, the hybridization of BSA and NCS is introduced in Section II. Section III exhibits the experiment result of BSANCS on benchmark function suit CEC'17 [114]. Finally, we conclude this paper in Section IV.

5.2 Hybrid BSANCS

The structure of BSA is similar to DE, including selection, mutation, crossover process. Due to this structure BSA can share the population information and utilize negatively correlated search to further improve its exploration abilities. Therefore, the hybridization of BSA and NCS is a good choice. The main mechanism of NCS is that utilizes Bhattacharyya distance to select solution with high quality and larger probability distributions. In NCS, the former mean that a better fitness $f(x)$, and

the latter is defined as shown in Eq. (5.3):

$$Corr(p_i) = \min_j \{ D_B(p_i, p_j) \mid i \neq j \}, \tag{5.3}$$

where p_i represents a probability distributions that the i-th individual in the population. $D_B(p_i, p_j)$ indicates that Bhattacharyya distance [106] between the i-th individual and j-th individual and the larger $Corr(p)$ is required. D_B is defined as:

$$D_B(p_i, p_j) = -\ln\left(\int \sqrt{p_i(x)p_j(x)}dx\right), \tag{5.4}$$

$$D_B(p_i, p_j) = -\ln\left(\sum_{x\in X} \sqrt{p_i(x)p_j(x)}\right). \tag{5.5}$$

Eq. (5.4) gives the Bhattacharyya distance for continue probability distribution, and Eq. (5.5) is the Bhattacharyya distance for discrete probability distribution. When the probability density function is not clear, the Bhattacharyya coefficient is usually used to estimate the Bhattacharyya distance.

Suppose x_i denotes the current solution, x_i' denotes the new solution. In each iteration, the better solution is selected to generate new solution in the next iteration and others will be discarded. Because $f(x_i)$ and $Corr(p_i)$ are maybe in different scales. $f(x_i)$ is negative and $Corr(p_i)$ is non-negative, it could be difficult to determine an appropriate trade-off. Thus, we let the normalizations of $f(x_i)+f(x_i')$ and $Corr(p_i)+Corr(p_i')$ equal to 1. In the normalization stage, $f(x_i)$ and $Corr(p_i)$ equal to $1-f(x_i')$ and $1 - Corr(p_i')$, respectively. The samller the $f(x_i')$, the better the quality of x_i' , and the larger $Corr(p_i)$, the less likely that the new generated x_i' is similar to other solution of individuals. Thus, smaller $f(x_i')$ and larger $Corr(p_i')$ will be selected, and a heuristic rule as defiened by Eq. (5.6) is adopted.

$$\begin{cases} discard\ x_i, & \textbf{if } \frac{f(x_i')}{Corr(p_i')} < \lambda \\ discard\ x_i', & \textbf{otherwise}, \end{cases} \tag{5.6}$$

where $\lambda \in (0, -\infty)$ is a parameter.

Given x_i and x_i', Eq. (5.6) indicates that λ determines whitch solution will be kept or discarded. Thus, the value of λ will affect search process and influent search ability. Setting λ to 1 indicates that $f(x_i')$ and $Corr(x_i')$ are equally improtant, usually as default value. However, the value of λ is variant in accordance with different search stages. Hence, a time-variant λ_t is adopted, as defined as Eq. (5.7):

$$\lambda_t = N(1, 0.1 - 0.1 * \frac{t}{T_{max}}), \tag{5.7}$$

where T_{max} is the user-defined total number of iterations. The Bhattacharyya distance can be written as Eq. (5.8):

$$D_B(p_i, p_j) = \frac{1}{8}(x_i - x_j)^T \Sigma^{-1}(x_i - x_j) + \frac{1}{2}\mathbf{ln}\left(\frac{\mathbf{det\Sigma}}{\sqrt{\mathbf{det\Sigma_i det\Sigma_j}}}\right), \tag{5.8}$$

where $\Sigma = \frac{\Sigma_i + \Sigma_j}{2}$.

Algorithm 2 shows the pseudo-code of BSANCS. Initialization process, and a population of solutions will be randomly generated and evaluated at lines 1-5. pop_{old} is randomly created and shuffled at lines 8-10. BSANCS generates mutation population by Eq. (5.1) and generates trial population by Eq. (5.2) at lines 12-29. Then, individuals search space is limited by utilizing boundary control at lines31-37. $f(x_i) + f(x_i')$ and $Corr(p_i) + Corr(p_i')$ euqal to 1 at line 41. According to Eq. (5.6), the best solution is found and P_{best} is updated at lines 42-43. Finally, for every *epoch* iteration,

Algorithm 2: Pseudo-Code of BSANCS

```
//Initialization;
Set t= 0;
globalminmum = inf;
pop_i = GeneratePopulation(popsize, dim, low, up);
fitnessP_i = function(pop_i);
for iteration from 1 to maxcycle do
    //Selection-I;
    if (a < b|a, b ~ U(0,1)) then
        pop_old = pop;
    pop_old = permuting(pop_old);
    //Mutation;
    pop_mutation = pop + F · (pop_historical − pop);
    pop_mutation = pop + F · (pop_historical − pop);
    //Crossover;
    if (c < d|c, d ~ U(0,1)) then
        for i from 1 to N do
            map_{i,u(1:rand)} = 0;
    else
        for i from 1 to N do
            map_{i,rand} = 0;
    T = mutant;
    for i from 1 to N do
        for j from 1 to D do
            if map_{i,j} = 1 then
                T_{i,j} = P_{i,j};
    //Boundary Control;
    for i from 1 to N do
        for j from 1 to D do
            if (T_{i,j} < low_j)or(T_{i.j} > up_j) then
                T_{i,j} = rand · (up_j − low_j) + low_j;
    //Selcetion-II;
    Set λ_t = N(1, 0.1 − 0.1 ∗ t/T_max);
    Compute f(x'_i), Corr(p_i) and Corr(p'_i);
    Normalization of f(x_i) + f(x'_i) and Corr(p_i) + Corr(p'_i) equal to 1;
    if f(x'_i)/Corr(p'_i) < λ then
        Update x_i with x'_i;
    t = t + 1;
    if mod(t, epoch) = 0 then
        Update F for each individuals according to 1/5 successful rule;
```

the search step-size will be update by Eq. (5.9) at lines 46-47.

$$\sigma_i = \begin{cases} \frac{\sigma_i}{r}, & if\ \frac{t}{epoch} > 0.2 \\ \sigma_i * r, & if\ \frac{t}{epoch} < 0.2 \\ \sigma_i, & if\ \frac{t}{epoch} = 0.2, \end{cases} \tag{5.9}$$

where r is a parameter, t is the number of replacements during the past *epoch* iterations.

5.3 Experimental Results

In this section, the proposed BSANCS algorithm is tested on the CEC2017 benchmark function suit. Due to F2 shows unstable behavior on higher dimensions, it is excluded in the CEC'17. Thus, the BSANCS algorithm is tested on the 29 function. The population size is set up to 100 and, dimensions is 30 and the number of max iteration is set up 3000. In order to reduce the random error, all the comparisons are run 30 times. In addition to BSA, three meta-heuristic optimization algorithms are selected to compare with BSANCS, including grey wolf optimization (GWO) [115,116], artificial bee colony algorithm (ABC), and sine cosine algorithm (SCA) [117].

Table 5.1 lists the results of all compared algorithms for all tested optimization functions and highlights the best using bold values. The mean value of 30 independent runs indicates the average performance of the algorithm, while the standard deviation (Std) denotes the robustness of the algorithm when addressing an optimization function. From Table 5.1, it can be found that BSANCS performs the best for 23 out of 30 optimization functions. Thus, it is obvious that the proposed BSANCS algorithm obtains the best value on most functions compared with BSA and has competitive performance on other functions. Then, Wilcoxon matched-pairs signed-rank test is used to precisely analyze the performance among BSANCS and other tested algo-

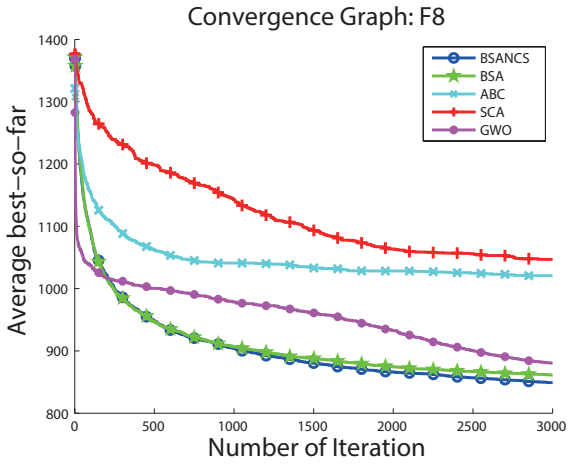

Figure 5.1: Convergence graph of F8.

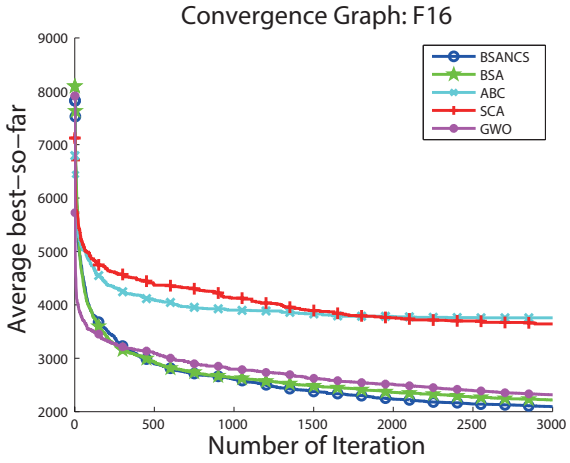

Figure 5.2: Convergence graph of F16.

rithms. Wilcoxon's test is a nonparametric procedure, which is used for hypothesis testing involving two samples. Table 5.2 exhibits the result of the Wilcoxon's test, it shows that the BSANCS algorithm is superior to other algorithms.

Convergence and Box-and-whisker graphs are used to further illustrate the superiority of the proposed algorithm. From Figs. 5.1, 5.2, 5.3 and 5.4, we can find that the convergence ability and speed of BSANCS are better than most of the comparison algorithms. From Figs. 5.5, 5.6, 5.7 and 5.8, we can find that algorithm BSANCS has better search ability and get a better solutions.

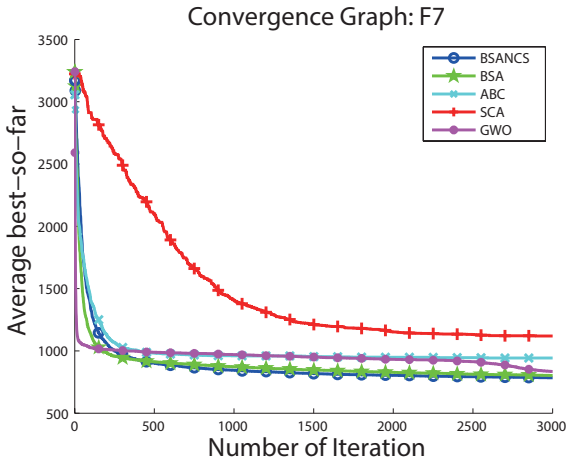

Figure 5.3: Convergence graph of F7.

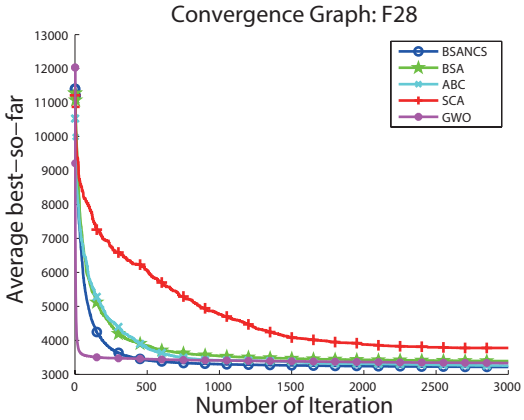

Figure 5.4: Convergence graph of F28.

5.4 Conclusion

In this paper, we proposed a hybrid method that combine BSA with NCS to improve the search ability. Experiment results suggest that the proposed BSANCS is effective in improving exploration ability and solution efficiency. In the future, we plan to apply the proposed BSANCS to multiple-objective optimization [82, 118], combinatorial optimization problems [79, 109], and neural network leaning tasks [119, 120].

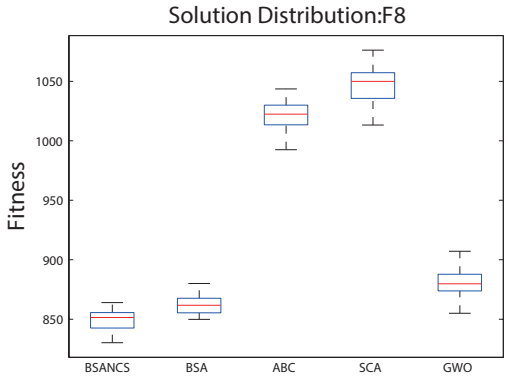

Figure 5.5: Box-and-whisker graph of F8.

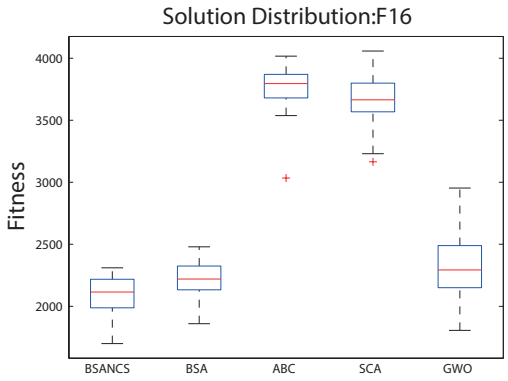

Figure 5.6: Box-and-whisker graph of F16.

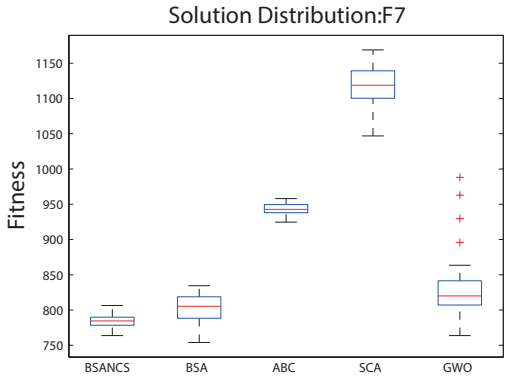

Figure 5.7: Box-and-whisker graph of F7.

Table 5.1: Experiment results of BSANCS and BSA on CEC2017.

	BSANCS		BSA		ABC		SCA		GWO	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	**1.57E+02**	7.79E+01	4.92E+02	4.58E+02	4.72E+04	7.74E+04	1.18E+10	1.77E+09	1.02E+09	8.91E+08
F3	**2.73E+04**	5.23E+03	2.86E+04	5.92E+03	1.03E+05	1.09E+04	3.50E+04	6.31E+03	2.85E+04	9.70E+03
F4	**4.77E+02**	2.36E+01	4.95E+02	2.11E+01	5.19E+02	2.79E+00	1.40E+03	2.74E+02	5.70E+02	4.98E+01
F5	5.69E+02	1.02E+01	**5.56E+02**	7.50E+00	7.18E+02	9.44E+00	7.71E+02	2.17E+01	5.92E+02	2.63E+01
F6	**6.00E+02**	2.41E-01	6.00E+02	5.85E-05	6.00E+02	6.69E-03	6.49E+02	5.34E+00	6.04E+02	2.33E+00
F7	**7.84E+02**	9.77E+00	8.02E+02	2.01E+01	9.43E+02	9.71E+00	1.12E+03	2.88E+01	8.35E+02	4.95E+01
F8	8.68E+02	9.25E+00	**8.61E+02**	7.40E+00	1.02E+03	1.16E+01	1.05E+03	1.63E+01	8.81E+02	1.25E+01
F9	1.06E+03	1.46E+02	**9.01E+02**	1.30E+00	1.90E+03	4.45E+02	5.52E+03	1.10E+03	1.18E+03	1.39E+02
F10	**3.99E+03**	2.32E+02	5.49E+03	2.33E+02	8.10E+03	3.19E+02	8.12E+03	3.34E+02	3.73E+03	5.49E+02
F11	**1.14E+03**	1.43E+01	1.15E+03	1.97E+01	4.37E+03	7.31E+02	2.19E+03	3.99E+02	1.51E+03	4.42E+02
F12	1.91E+05	1.53E+05	1.57E+05	1.22E+05	**1.17E+08**	2.66E+07	1.21E+09	2.30E+08	3.31E+07	3.81E+07
F13	**8.57E+03**	4.69E+03	9.22E+03	5.02E+03	8.02E+07	3.32E+07	4.07E+08	1.98E+08	6.63E+06	2.33E+07
F14	1.47E+03	1.32E+01	1.47E+03	1.90E+01	3.04E+05	1.25E+05	**1.19E+05**	7.05E+04	8.10E+04	1.76E+05
F15	**1.47E+03**	1.68E+01	1.63E+03	1.02E+02	2.08E+07	8.65E+06	1.56E+07	1.29E+07	2.44E+05	5.82E+05
F16	**2.09E+03**	1.49E+02	2.22E+03	1.56E+02	3.76E+03	1.87E+02	3.64E+03	2.13E+02	2.32E+03	2.39E+02
F17	**1.79E+03**	2.41E+01	1.82E+03	4.20E+01	2.49E+03	1.19E+02	2.42E+03	1.64E+02	1.93E+03	1.16E+02
F18	2.02E+04	7.27E+03	**7.84E+03**	3.68E+03	6.34E+06	3.20E+06	2.80E+06	1.21E+06	7.75E+05	1.40E+06
F19	**1.98E+03**	4.97E+01	2.03E+03	1.12E+02	2.39E+07	1.03E+07	2.48E+07	1.13E+07	2.06E+05	3.88E+05
F20	**2.14E+03**	5.44E+01	2.15E+03	5.62E+01	2.74E+03	8.15E+01	2.61E+03	1.29E+02	2.33E+03	1.66E+02
F21	**2.35E+03**	7.38E+00	2.36E+03	9.85E+00	2.52E+03	1.18E+01	2.56E+03	1.92E+01	2.37E+03	1.85E+01
F22	**2.34E+03**	2.14E+01	2.50E+03	7.70E+02	2.64E+03	2.08E+02	8.25E+03	2.37E+03	4.47E+03	1.45E+03
F23	**2.69E+03**	9.63E+00	2.71E+03	8.32E+00	2.89E+03	1.60E+01	2.99E+03	2.34E+01	2.73E+03	3.04E+01
F24	**2.86E+03**	9.37E+00	2.89E+03	1.36E+01	3.04E+03	1.17E+01	3.16E+03	2.96E+01	2.90E+03	4.70E+01
F25	**2.89E+03**	2.08E-01	2.89E+03	1.53E+00	2.89E+03	1.73E-01	3.20E+03	4.91E+01	2.96E+03	2.69E+01
F26	**3.49E+03**	4.71E+02	4.03E+03	2.40E+02	5.74E+03	1.28E+02	6.87E+03	2.56E+02	4.43E+03	2.45E+02
F27	**3.21E+03**	3.97E+00	3.21E+03	5.05E+00	3.46E+03	3.87E+01	3.39E+03	4.83E+01	3.23E+03	1.78E+01
F28	**3.22E+03**	8.64E+00	3.39E+03	3.89E+02	3.26E+03	2.59E+01	3.78E+03	1.36E+02	3.33E+03	4.83E+01
F29	**3.42E+03**	3.99E+01	3.49E+03	4.13E+01	4.93E+03	1.31E+02	4.62E+03	2.62E+02	3.71E+03	1.26E+02
F30	**7.05E+03**	6.56E+02	7.20E+03	7.06E+02	2.67E+07	1.04E+07	7.44E+07	2.59E+07	3.90E+06	3.10E+06

Table 5.2: Results obtained by the Wilcoxon test.

	R^+	R^-	Asymptotic P-value
BSA	338.0	97.0	0.007825
ABC	433.5	1.5	0.000003
SCA	435.0	0.0	0.000002
GWO	419.0	16.0	0.000011

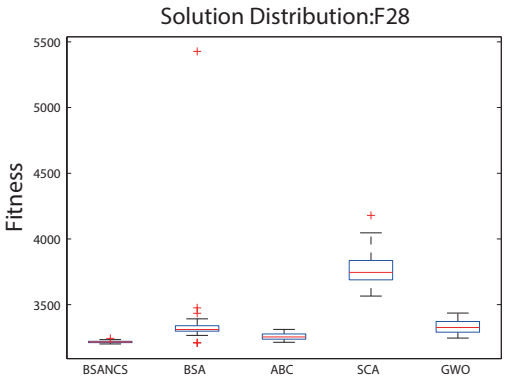

Figure 5.8: Box-and-whisker graph of F28.

Chapter 6

Conclusions and Remarks

In this thesis, we study search dynamics and evolutionary algorithms, and the application to real-world issue fixing. The study have convince us,with the help of hybridization,evolution algorithm has shown diversity, flexibility and effectiveness of meta-heuristic hybridization in their applications. We also take insight into the mechanism of hybridization, there is no quintessential difference among them. The only difference is search operators. Two main factors can be considered: search pattern and search style. We also study the advantages and disadvantages of certain algorithm ,by improve their search ability to reinforce the exploration and exploitation abilities in order to get an excellent hybrid algorithm.

In the future research, we will focus on study the search ability in different search pattern and search style.Also control strategies among these kinds of search styles are conducted to provide a good balance between exploration and exploitation phases,we plan to study more search styles and try to incorporate them for improvement of algorithms.The following studies is planned to carried out in the near future:

- Firstly, this research show the inherent connection between search dynamic and evolutionary algorithms.To achieve a good balance between exploration and exploitation,search styles incorporation and control strategies is the key so I will keep in studying more search styles and design more control strategies.

- Secondly,by studying the recently raised advanced evolutionary algorithms, such the Spherical Evolution, have huge search space,however,leak of convergence speed, can be optimize by the way similar to genetic information transmission to remove its invalid search range,then the algorithm can be greatly enhanced,when the search ability of the basic search style is enhanced,after hybridization,the search quality and robustness will be enhanced correspondingly.

- Last but not least, when coming back to real-world optimization problems,when an algorithm only need to execute a simple control strategy to adjust adaptability of the search style,thus, its performance can be greatly improved by using diverse control strategies in the future.Therefore,it meaningful to study this kind of optimization methods and having more potential for implementation,as a tentative, my next step is adopt improved TDSD algorithm to AGV product at route plan stage.

Bibliography

[1] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[2] D. Shen, T. Jiang, W. Chen, Q. Shi, and S. Gao, "Improved chaotic gravitational search algorithms for global optimization," in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 1220–1226.

[3] A. A. Salman, I. Ahmad, M. G. Omran, and M. G. Mohammad, "Frequency assignment problem in satellite communications using differential evolution," *Computers & Operations Research*, vol. 37, no. 12, pp. 2152–2163, 2010.

[4] S. Gao, S. Song, J. Cheng, Y. Todo, and M. Zhou, "Incorporation of solvent effect into multi-objective evolutionary algorithm for improved protein structure prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 4, pp. 1365–1378, 2018.

[5] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*. Springer, 2011, pp. 760–766.

[6] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of Machine Learning*. Springer, 2011, pp. 36–39.

[7] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[8] P. Civicioglu and E. Besdok, "A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 315–346, 2013.

[9] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[11] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[12] S. Gao, Y. Wang, J. Wang, and J. Cheng, "Understanding differential evolution: a Poisson law derived from population interaction network," *Journal of Computational Science*, vol. 21, pp. 140–149, 2017.

[13] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[14] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 2. IEEE, 2005, pp. 1785–1791.

[15] K. Tang, P. Yang, and X. Yao, "Negatively correlated search," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 542–550, 2016.

[16] P. Civicioglu, "Backtracking search optimization algorithm for numerical optimization problems," *Applied Mathematics and Computation*, vol. 219, no. 15, pp. 8121–8144, 2013.

[17] W. Wang, S. Gao, and Z. Tang, "A complex artificial immune system," in *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, vol. 6. IEEE, 2008, pp. 597–601.

[18] S. Gao, H. Chai, B. Chen, and G. Yang, "Hybrid gravitational search and clonal selection algorithm for global optimization," in *International Conference in Swarm Intelligence*. Springer, 2013, pp. 1–10.

[19] S. Gao, W. Wang, H. Dai, F. Li, and Z. Tang, "Improved clonal selection algorithm combined with ant colony optimization," *IEICE Transactions on Information and Systems*, vol. 91, no. 6, pp. 1813–1823, 2008.

[20] Y. Yu, S. Gao, S. Cheng, Y. Wang, S. Song, and F. Yuan, "CBSO: a memetic brain storm optimization with chaotic local search," *Memetic Computing*, vol. 10, no. 4, pp. 353–367, 2017.

[21] Y. Wang, Y. Yu, S. Cao, X. Zhang, and S. Gao, "A review of applications of artificial intelligent algorithms in wind farms," *Artificial Intelligence Review*, 2019, DOI: 10.1007/s10462-019-09768-7.

[22] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neural model with effective learning algorithms for classification, approximation, and prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 601–614, 2019.

[23] S. Gao, Y. Wang, J. Cheng, Y. Inazumi, and Z. Tang, "Ant colony optimization with clustering for solving the dynamic location routing problem," *Applied Mathematics and Computation*, vol. 285, pp. 149–173, 2016.

[24] Y. Wang, S. Gao, and Y. Todo, "Ant colony systems for optimization problems in dynamic environments," *Swarm Intelligence: Principles, Current Algorithms and Methods*, vol. 119, pp. 85–120, 2018.

[25] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary computation 1: Basic algorithms and operators*. CRC press, 2018.

[26] Y. Wang, S. Gao, Y. Yu, and Z. Xu, "The discovery of population interaction with a power law distribution in brain storm optimization," *Memetic Computing*, vol. 11, no. 1, pp. 65–87, 2019.

[27] Y. Yu, S. Gao, Y. Wang, and Y. Todo, "Global optimum-based search differential evolution," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 379–394, 2019.

[28] B. Dorronsoro and P. Bouvry, "Improving classical and decentralized differential evolution with new mutation operator and population topologies," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 67–98, 2011.

[29] R. Stom, "Differential evolution ¨c a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, p. 341¨C359, 1997.

[30] J. H. Wang, J. J. Liao, Y. Zhou, and Y. Q. Cai, "Differential evolution enhanced with multiobjective sorting-based mutation operators," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2792–2805, 2014.

[31] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[32] M. Yin, Y. Hu, F. Yang, X. Li, and W. Gu, "A novel hybrid k-harmonic means and gravitational search algorithm approach for clustering," *Expert Systems with Applications*, vol. 38, no. 8, pp. 9319–9324, 2011.

[33] G. Tian, Y. Ren, and M. Zhou, "Dual-objective scheduling of rescue vehicles to distinguish forest fires via differential evolution and particle swarm optimization combined algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3009–3021, 2016.

[34] Y. Shi, "Brain storm optimization algorithm," in *International Conference in Swarm Intelligence*. Springer, 2011, pp. 303–309.

[35] Y. Yu, S. Gao, S. Cheng, Y. Wang, S. Song, and F. Yuan, "CBSO: a memetic brain storm optimization with chaotic local search," *Memetic Computing*, vol. 10, no. 4, pp. 353–367, 2018.

[36] Y. Yu, S. Gao, Y. Wang, J. Cheng, and Y. Todo, "ASBSO: an improved brain storm optimization with flexible search length and memory-based selection," *IEEE Access*, vol. 6, pp. 36 977–36 994, 2018.

[37] H. Chen and N. S. Flann, "Parallel simulated annealing and genetic algorithms: a space of hybrid methods," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1994, pp. 428–438.

[38] R.-L. Wang, X.-F. Zhou, and K. Okazaki, "Ant colony optimization with genetic operation and its application to traveling salesman problem," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 92, no. 5, pp. 1368–1372, 2009.

[39] H. S. Lope and L. S. Coelho, "Particle swarn optimization with fast local search for the blind traveling salesman problem," in *Fifth International Conference on Hybrid Intelligent Systems*. IEEE, 2005, pp. 245–250.

[40] M. Malek, M. Guruswamy, M. Pandya, and H. Owens, "Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem," *Annals of Operations Research*, vol. 21, no. 1, pp. 59–84, 1989.

[41] Y. Marinakis and M. Marinaki, "A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem," *Computers & Operations Research*, vol. 37, no. 3, pp. 432–442, 2010.

[42] S. Gao, W. Wang, H. Dai, F. Li, and Z. Tang, "Improved clonal selection algorithm combined with ant colony optimization," *IEICE Transactions on Information and Systems*, vol. 91, no. 6, pp. 1813–1823, 2008.

[43] M. Gündüz, M. S. Kiran, and E. Özceylan, "A hierarchic approach based on swarm intelligence to solve the traveling salesman problem," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 23, no. 1, pp. 103–117, 2015.

[44] B. Shuang, J. Chen, and Z. Li, "Study on hybrid PS-ACO algorithm," *Applied Intelligence*, vol. 34, no. 1, pp. 64–73, 2011.

[45] W. Deng, R. Chen, B. He, Y. Liu, L. Yin, and J. Guo, "A novel two-stage hybrid swarm intelligence optimization algorithm and application," *Soft Computing*, vol. 16, no. 10, pp. 1707–1722, 2012.

[46] M. Mahi, O. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem," *Applied Soft Computing*, vol. 30, pp. 484–490, 2015.

[47] S.-M. Chen and C.-Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," *Expert Systems with Applications*, vol. 38, no. 12, pp. 14 439–14 450, 2011.

[48] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[49] S. A. Uymaz, G. Tezel, and E. Yel, "Artificial algae algorithm (AAA) for nonlinear global optimization," *Applied Soft Computing*, vol. 31, pp. 153–171, 2015.

[50] A. Nasir and M. O. Tokhi, "An improved spiral dynamic optimization algorithm with engineering application," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 6, pp. 943–954, 2015.

[51] D. Tang, "Spherical evolution for solving continuous optimization problems," *Applied Soft Computing*, vol. 81, p. 105499, 2019.

[52] D. Jia, G. Zheng, and M. K. Khan, "An effective memetic differential evolution algorithm based on chaotic local search," *Information Sciences*, vol. 181, no. 15, pp. 3175–3187, 2011.

[53] S. Gao, Y. Yu, Y. Wang, J. Wang, J. Cheng, and M. Zhou, "Chaotic local search-based differential evolution algorithms for optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019, DOI: 10.1109/TSMC.2019.2956121.

[54] S. Gao, C. Vairappan, Y. Wang, Q. Cao, and Z. Tang, "Gravitational search algorithm combined with chaos for unconstrained numerical optimization," *Applied Mathematics and Computation*, vol. 231, pp. 48–62, 2014.

[55] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.

[56] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons & Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[57] Y. Lu, J. Zhou, H. Qin, Y. Wang, and Y. Zhang, "Chaotic differential evolution methods for dynamic economic dispatch with valve-point effects," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 2, pp. 378–387, 2011.

[58] G.-G. Wang, L. Guo, A. H. Gandomi, G.-S. Hao, and H. Wang, "Chaotic krill herd algorithm," *Information Sciences*, vol. 274, pp. 17–34, 2014.

[59] B. L. W. JIANG, "Optimizing complex functions by chaos search," *Cybernetics & Systems*, vol. 29, no. 4, pp. 409–419, 1998.

[60] G.-G. Wang, S. Deb, A. H. Gandomi, Z. Zhang, and A. H. Alavi, "Chaotic cuckoo search," *Soft Computing*, vol. 20, no. 9, pp. 3349–3362, 2016.

[61] S. Mirjalili and A. H. Gandomi, "Chaotic gravitational constants for the gravitational search algorithm," *Applied Soft Computing*, vol. 53, pp. 407–419, 2017.

[62] Y. Wang, Y. Yu, S. Gao, H. Pan, and G. Yang, "A hierarchical gravitational search algorithm with an effective gravitational constant," *Swarm and Evolutionary Computation*, vol. 46, pp. 118–139, 2019.

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

[64] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[65] ——, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," *Jadavpur University, Kolkata*, 2010.

[66] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[67] Z. Song, S. Gao, Y. Yu, J. Sun, and Y. Todo, "Multiple chaos embedded gravitational search algorithm," *IEICE Transactions on Information and Systems*, vol. 100, no. 4, pp. 888–900, 2017.

[68] R. Mallipeddi, P. N. Suganthan, Q.-K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, 2011.

[69] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2008.

[70] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[71] A. Kumar, R. K. Misra, and D. Singh, "Improving the local search capability of effective butterfly optimizer using covariance matrix adapted retreat phase," in *2017 IEEE Congress on Evolutionary Computation*. IEEE, 2017, pp. 1835–1842.

[72] N. H. Awad, M. Z. Ali, and P. N. Suganthan, "Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems," in *2017 IEEE Congress on Evolutionary Computation*. IEEE, 2017, pp. 372–379.

[73] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi, "LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems," in *2017 IEEE Congress on evolutionary computation*. IEEE, 2017, pp. 145–152.

[74] R. Tanabe and A. Fukunaga, "Evaluating the performance of SHADE on CEC 2013 benchmark problems," in *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 1952–1959.

[75] I. H. Osman and J. P. Kelly, "Meta-heuristics: an overview," in *Meta-heuristics*. Springer, 1996, pp. 1–21.

[76] S. Gao, *Bio-Inspired Computational Algorithms and Their Applications*. IntechOpen, London, UK, 2012.

[77] Z. Beheshti and S. M. H. Shamsuddin, "A review of population-based meta-heuristic algorithms," *International Journal of Advances in Soft Computing and Its Applications*, vol. 5, no. 1, pp. 1–35, 2013.

[78] J. Wang, B. Cen, S. Gao, Z. Zhang, and Y. Zhou, "Cooperative evolutionary framework with focused search for many-objective optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018, doi: 10.1109/TETCI.2018.2849380.

[79] S. Gao, J. Zhang, X. Wang, and Z. Tang, "Multi-layer neural network learning algorithm based on random pattern search method," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 2, pp. 489–502, 2009.

[80] S. Wang, S. Gao, Y. Todo, Z. Tang *et al.*, "A multi-learning immune algorithm for numerical optimization," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 98, no. 1, pp. 362–377, 2015.

[81] Y.-J. Gong, J.-J. Li, Y. Zhou, Y. Li, H. S.-H. Chung, Y.-H. Shi, and J. Zhang, "Genetic learning particle swarm optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2277–2290, 2016.

[82] S. Song, S. Gao, X. Chen, D. Jia, X. Qian, and Y. Todo, "AIMOES: Archive information assisted multi-objective evolutionary strategy for ab initio protein structure prediction," *Knowledge-Based Systems*, vol. 146, pp. 58–72, 2018.

[83] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[84] J. Sun, S. Gao, H. Dai, J. Cheng, M. Zhou, and J. Wang, "Bi-objective elite differential evolution for multivalued logic networks," *IEEE Transactions on Cybernetics*, 2018, doi: 10.1109/TCYB.2018.2868493.

[85] H. Yu, Z. Xu, S. Gao, Y. Wang, and Y. Todo, "PMPSO: a near-optimal graph planarization algorithm using probability model based particle swarm optimization," in *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, 2015, pp. 15–19.

[86] H. Yu, X. Qian, Y. Yu, J. Cheng, Y. Yu, and S. Gao, "A novel mutual information based ant colony classifier," in *2017 IEEE International Conference on Progress in Informatics and Computing (PIC)*. IEEE, 2017, pp. 61–65.

[87] Y. Wang, S. Gao, Y. Yu, and Z. Xu, "The discovery of population interaction with a power law distribution in brain storm optimization," *Memetic Computing*, 2017, doi: 10.1007/s12293-017-0248-z.

[88] Y. Yu, S. Gao, Y. Wang, J. Cheng, and Y. Todo, "ASBSO: An improved brain storm optimization with flexible search length and memory-based selection," *IEEE Access*, vol. 6, no. 1, pp. 36 977–36 994, 2018.

[89] J. He and X. Yao, "From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 495–511, 2002.

[90] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, and J.-J. Li, "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," *Applied Soft Computing*, vol. 34, pp. 286–300, 2015.

[91] S. Gao, C. Vairappan, Y. Wang, Q. Cao, and Z. Tang, "Gravitational search algorithm combined with chaos for unconstrained numerical optimization," *Applied Mathematics and Computation*, vol. 231, pp. 48–62, 2014.

[92] J. Ji, S. Gao, S. Wang, Y. Tang, H. Yu, and Y. Todo, "Self-adaptive gravitational search algorithm with a modified chaotic local search," *IEEE Access*, vol. 5, pp. 17 881–17 895, 2017.

[93] B. Schutz, *Gravity from the ground up: An introductory guide to gravity and general relativity*. Cambridge University Press, 2003.

[94] H. Yu, H. Zhu, H. Chen, D. Jia, Y. Yu, and S. Gao, "Gravitational search algorithm combined with modified differential evolution learning for planarization in graph drawing," in *2017 IEEE International Conference on Progress in Informatics and Computing (PIC)*. IEEE, 2017, pp. 1–6.

[95] S. Li, T. Jiang, H. Chen, D. Shen, Y. Todo, and S. Gao, "Discrete chaotic gravitational search algorithm for unit commitment problem," in *International Conference on Intelligent Computing*. Springer, 2016, pp. 757–769.

[96] S. Gao, Y. Todo, T. Gong, G. Yang, and Z. Tang, "Graph planarization problem optimization based on triple-valued gravitational search algorithm," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 9, no. 1, pp. 39–48, 2014.

[97] S. Li, F. Yuan, Y. Yu, J. Ji, Y. Todo, and S. Gao, "Galactic gravitational search algorithm for numerical optimization," in *International Conference on Swarm Intelligence*. Springer, 2018, pp. 397–409.

[98] G. Chen, L. Liu, Z. Zhang, and S. Huang, "Optimal reactive power dispatch by improved GSA-based algorithm with the novel strategies to handle constraints," *Applied Soft Computing*, vol. 50, pp. 58–70, 2017.

[99] G. Sun, A. Zhang, Y. Yao, and Z. Wang, "A novel hybrid algorithm of gravitational search algorithm with genetic algorithm for multi-level thresholding," *Applied Soft Computing*, vol. 46, pp. 703–730, 2016.

[100] G. Sun, A. Zhang, X. Jia, X. Li, S. Ji, and Z. Wang, "DMMOGSA: Diversity-enhanced and memory-based multi-objective gravitational search algorithm," *Information Sciences*, vol. 363, pp. 52–71, 2016.

[101] A. Zhang, G. Sun, J. Ren, X. Li, Z. Wang, and X. Jia, "A dynamic neighborhood learning-based gravitational search algorithm," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 436–447, 2018.

[102] N. Zhang, C. Li, R. Li, X. Lai, and Y. Zhang, "A mixed-strategy based gravitational search algorithm for parameter identification of hydraulic turbine governing system," *Knowledge-Based Systems*, vol. 109, pp. 218–237, 2016.

[103] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[104] S. Mirjalili, S. Z. M. Hashim, and H. M. Sardroudi, "Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11 125–11 137, 2012.

[105] E. Rashedi, E. Rashedi, and H. Nezamabadi-pour, "A comprehensive survey on gravitational search algorithm," *Swarm and Evolutionary Computation*, vol. 41, pp. 141–158, 2018.

[106] T. Kailath, "The divergence and Bhattacharyya distance measures in signal selection," *IEEE Transactions on Communication Technology*, vol. 15, no. 1, pp. 52–60, 1967.

[107] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[108] H. Dai, S. Gao, Y. Yang, and Z. Tang, "Effects of ?rich-gets-richer? rule on small-world networks," *Neurocomputing*, vol. 73, no. 10-12, pp. 2286–2289, 2010.

[109] J. Cheng, X. Wu, M. Zhou, S. Gao, Z. Huang, and C. Liu, "A novel method for detecting new overlapping community in complex evolving networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018, doi: 10.1109/TSMC.2017.2779138.

[110] J. Cheng, M. Chen, M. Zhou, S. Gao, C. Liu, and C. Liu, "Overlapping community change point detection in an evolving network," *IEEE Transactions on Big Data*, 2018, doi: 10.1109/TBDATA.2018.2880780.

[111] J. Cheng, J. Cheng, M. Zhou, F. Liu, S. Gao, and C. Liu, "Routing in internet of vehicles: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2339–2352, 2015.

[112] J. Cheng, H. Mi, Z. Huang, S. Gao, D. Zang, and C. Liu, "Connectivity modeling and analysis for internet of vehicles in urban road scene," *IEEE Access*, vol. 6, pp. 2692–2702, 2018.

[113] Y. Liu, D. Cheng, Y. Wang, J. Cheng, and S. Gao, "A novel method for predicting vehicle state in internet of vehicles," *Mobile Information Systems*, vol. 2018, Article ID 9728328, 2018.

[114] N. Awad, M. Ali, J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2017 special session and competition on single ob-

jective bound constrained real-parameter numerical optimization," in *Technical Report*. NTU, Singapore, 2016.

[115] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[116] H. Yu, Y. Yu, Y. Liu, Y. Wang, and S. Gao, "Chaotic grey wolf optimization," in *Progress in Informatics and Computing (PIC), 2016 International Conference on*. IEEE, 2016, pp. 103–113.

[117] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[118] Y. Zhou, J. Wang, J. Chen, S. Gao, and L. Teng, "Ensemble of many-objective evolutionary algorithms for many-objective problems," *Soft Computing*, vol. 21, no. 9, pp. 2407–2419, 2017.

[119] J. Ji, S. Gao, J. Cheng, Z. Tang, and Y. Todo, "An approximate logic neuron model with a dendritic structure," *Neurocomputing*, vol. 173, pp. 1775–1783, 2016.

[120] T. Zhou, S. Gao, J. Wang, C. Chu, Y. Todo, and Z. Tang, "Financial time series prediction using a dendritic neuron model," *Knowledge-Based Systems*, vol. 105, pp. 214–224, 2016.