Soft Computing by Considering Evolutionary and Dendritic Mechanisms

by

Zhenyu Song

A dissertation

submitted to the Graduate School of Innovative Life Science

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Engineering

University of Toyama

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

2017

(Submitted January 5, 2017)

Acknowledgements

I would like to deeply thank the people who during my study and research, given me useful assistance and beneficial help. Without their care and help, this thesis is not able to complete.

Thank for my mentor Prof. Zheng Tang at University of Toyama, he gave me a detailed description of the development of intelligent evolutionary computation and plenty of interesting research, for his support and continued encouragement. Without his kinds of guidance and encouragement, I would never have done this research paper. In addition, his help and support not only in my study career, but also concerned about my everyday life in Japan. Many heated discussion and support let me continue to since I came to Japan in the past time. Thanks to him, I can successfully complete the dissertation in three years.

I would like to thank my thesis referees, Prof. Gao, Originally he gave me the research direction, from University of Toyama for a review and qualification of my thesis and giving various valuable comments and suggestions. Thanks to my side classmates for three years, with their help and care makes life enrich and interesting, constantly enrich myself in the study, I believe that this must be a best memory in my life.

Thanks to the parents in the home, they have been the indispensable material and spiritual pillar in my study life. They give support make me full of gratitude and respect, and I wish them healthy and safely.

Abstract

The purpose of the research on artificial intelligence (AI) is to comprehend the intelligence entity. Research the AI on the one hand is to better understand ourselves, on the other hand is to construct the intelligent entities, and will be of great use in our life also. With the development of AI continuously, people have already made many products has important effect to the human. Although no one can predict the future development of AI, but the computer is as intelligent as humans will give us a significant impact to our daily life.

Swarm intelligence refers to a way to solve the problem in the process of interacting with the simple information processing unit. The concept of swarm indicate that it has diversity, randomness and chaos, intelligent shows that it is one way to solve the problem successfully, Information processing unit group can be a bunch of insects, a flock of birds or a group of human, may also be a set of elements, a group of robots or independent workstation, it can be true, can also is abstract. Their coupling can be established in a generalized characteristic, but between unit must be interacting.

Swarm intelligence algorithm is in recent years to the rise of the optimization algorithm, since the 80s of the last century, caused the attention of multi disciplinary domain experts, And it has become a hotspot in the field of optimization technology, AI and economic, social, biological and other cross disciplinary research hotspot. Some classical swarm optimization algorithm with genetic algorithm, ant colony algorithm, particle swarm algorithm, quantum behaved particle swarm algorithm.

In last few years, various swarm intelligent algorithm inspired by nature phenomena were proposed. There are show by numerous experiments that these algorithms are good tool to solve complex single and multiple object optimization problems. To op-

timization problems in high-dimensional space, the all traditional class optimization algorithms cannot provide suitable solve solution, continuing to study optimization algorithm still has its essential value. Gravitational Search Algorithm (GSA) is a new optimization algorithm for the same purpose, due to its simple principle and its high efficiency in solving various nonlinear functions in recent years has become a search hot spot, and has been applied in some fields. Search of the algorithm has two main aspects: on the hand, how to improve the search accuracy, one the other hand, how to accelerate its speed of convergence rate. The article is from the two aspects to improve GSA. GSA based on Newton Law of Gravity and mass interactions is proposed and a new optimization algorithm. This paper is mainly about the law of gravitation formula in the corresponding transformation to improve the GSA for improving the search accurate and accelerate speed of convergence.

On the other hand, in Newton Law of Gravity, the gravitational force between two particles is proportional to the product of their masses, and inversely proportional to the square of the distance. To improve the search accurate of GSA, we assign a weight value to every agent in each iteration process. Large inertial mass particle inertness is bigger, smaller inertial mass particle inertness is smaller. The moving distance of large inertial mass is smaller in each iteration process, the moving distance of small inertial mass is larger in each iteration process. Therefore all of the agents will rapidly move to the optimal position. improving the search accurate at the same time, accelerating convergence.

We use of the ergodicity and stochasticity of chaotic search, is combined into GSA, embedded 12 kinds of chaos and 3 kinds of chaotic search algorithm. The experimental results show that incorporated chaotic search can effectively improve the solution quality.

Recently, more neuroscience researches focus on the role of dendritic structure during the neural computation. Inspired by the specified topologies of numerous dendritic trees, we proposed a single neural model with a particular dendritic structure. The dendrites are composed of several branches, and these branches correspond to three distributions in coordinate, which are used to classify the training data as required.

Genetic algorithm is used as the training algorithm. Experimental results based on t-wo benchmark classification problems verify the effectiveness of the proposed method, and the distributions of trained dendritic structures are also presented.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Meta-heuristic Optimization

The two basic goals of computer science is to find out the algorithm that can prove its efficiency and can get the best or second best solution. The heuristic algorithm tries to provide one or all of the goals at a time. For example, it can often find a good solution, but there is no way to prove that it will not get worse solution. It is usually obtained from the answer in a reasonable time, but there is no way to know whether it can solve such speed every time. Sometimes, people will find that in some special cases, the heuristic algorithm will get a very bad answer or the efficiency is very poor, however, resulting in the special circumstances of the data combination, may never appear in the real world. Therefore, heuristic algorithms commonly used to solve the problem in the real world A heuristic algorithm for handling many practical problems usually get a good answer in reasonable time.

Comparison with Iterative Reconstruction Methods and optimization algorithms (OAs), meta-heuristics do not assure that a globally optimal solution can be upon some classify problems [1]. Many algorithms are implemented in the form of stochastic optimization, which depends on the generation of random variables in order to find the solution. In combinatorial optimization problems, by a heuristic search in a set of feasible solutions, the solution can be found with less computational amount than the iterative method and the simple heuristic algorithm. Therefore, they are useful methods for optimization problems. Most of the heuristic literature is experimental

in nature, and it is generally described the experimental results based on the algorithm. Through the convergence of the algorithm and the global optimal can be found, some of the results obtained are very available. Many heuristic algorithms have been recognized for his practicality and novelty. Research on the heuristic algorithm in the optimization problem The optimization is starting from the nature, such as engineering design, economic construction, vacation planning, network routing Because we have limited resources, time and money, so these available resources can get the best use for us is very important [2].

Most of the optimization in the real world is nonlinear and multi dimension, and in many complex conditions, the goals may be contradictory. Even if it is a goal, there may be no optimal solution at all. Usually, to find an optimal solution, or even to find a suboptimal solution is not an easy task. The following describes the history of the heuristic algorithm, algorithm principles and characteristics.

1.1.1 A Brief History

In human history, many problems tend to heuristic, however, as the scientific research method of heuristic optimization is a common phenomenon in modern From 1940s to 1960s, the heuristic method has already been used in a variety of applications, until the emergence of the evolutionary algorithm, which was the first to have milepost sense symbol, In 1963 Ingo Rechenberg and Hans Paul Schwefel was the first development of evolutionary algorithm, in 1966 L. J. Fogel began to develop the evolutionary algorithm. The genetic algorithm (GA) was by J. Holland developed in 1960s to 1970s, he also published a book about the GA in 1975s [3].

In 1980s and 1990s was the most exciting moment of the meta-heuristic algorithms. The simulated annealing (SA) algorithm was developed in 1983 [4], which makes the research of heuristic algorithm was a big step forward, it is a kind of optimization method, pioneered by S. Kirkpatrick et al., inspired by the metal annealing process, is the inspiration, to create the SA algorithm. In 1986, another important development is from the Farmer et al. development of the artificial immune system.

Marco Dorigo completed natural OA doctoral dissertation in 1992, thesis describes his innovative research of ant colony optimization (ACO) [5]. This search technology was inspired by the swarm intelligence activities of ants, which use pheromone as a chemical messenger. Then, John R. Koza published a book on genetic programming in 1992, which was to lay the foundation for a new field of machine learning, this was a revolutionary programming of the computer.

In 1995 Kennedy Russell and C. Eberhart James proposed particle swarm optimization (PSO) algorithm [6], Storn R. and Price K. proposed vector based calculation of the evolutionary algorithm in1997 , it was differential evolution (DE) [7], Which is proved to be more effective than the GA in many applications.

At the beginning of 21 Century, Zong Woo Geem et al. proposed the harmony search algorithm (HSA) [8], which was inspired by music in 2001. Passino et al. proposed a bacterial foraging algorithm, which was inspired by the bacterial colony foraging around 2002.

In 2004 S. Nakrani and C. Tovey et al. proposed the application of honey bee algorithm and network center, in 2005 D. T. Pham and D. Karaboga et al. to optimize the bee colony algorithm, and put forward the artificial bee colony (ABC) algorithm [9]. The firefly algorithm (FA) was proposed by Krishnanand et al. in 2008 [10]. In 2009 Xin-She Yang and Suash Deb et al. presents an efficient cuckoo search (CS) algorithm [11], it has been proved that cuckoo search algorithm than most of the meta-heuristic algorithms more effective including the PSO algorithm [12].

1.1.2 Optimization Algorithms

Search capability and efficient OA is necessary to solving optimization problems, OAs can be classified according to the emphasis and characteristics in many aspects.

If we focus on the gradient or derivative of a function, the optimization problems can be divided into a gradient based algorithm and a gradient derivative free algorithm. Gradient based algorithms using derivative information as mountain climbing, they are generally very effective. Derivative free algorithms do not use any derivative

information, but with the value of the function itself. Some functions are discontinuous and computational expense is high, so the derivative free algorithm is simplex and practical.

From different point of view, the OA can be divided into trajectory based algorithm and population based algorithm. Trajectory based algorithms generally use a single agent or only one solution, based on a path of continuous iteration. SA algorithm is a typical trajectory based heuristic algorithm. On the other hand, the population based algorithms, such as PSO algorithm, ant colony algorithm, artificial bee colony algorithm, which use multiple agents to find the optimal solutions.

OA can also be classified according to certainty and randomness, if in the implementation of a deterministic mechanical algorithm, without any randomness and contingency, it is called a deterministic OA. This kind of algorithms if we set the same initial point, it will find the same solution finally On the other hand, if the algorithm has certain randomness, so even if set the same initial point, the algorithm usually gets a different solution after each execution. For example, GA and particle swarm algorithm is a typical stochastic algorithm.

According to the search capability, the OA can be classified, and it is generally divided into the global search algorithm and the local search algorithm. Local search algorithms generally converge to a local optimal solution, this solution is often not the global optimal solution, this class of algorithms is usually has not ability to go out of the local optimal. However, for a global optimization problem, the local search algorithm cannot meet the requirements obviously, it should use the global search algorithm. Now the meta-heuristic algorithm can carry out generally be a global optimization, but it is not necessarily effective or successful. In essence, the random selection is an important component of the global search algorithm.

Obviously, between the algorithm and another algorithm can be combined with the association, so we can design high capacity and effective algorithm.

1.1.3 Characteristics of Meta-heuristics

Previously, we put the stochastic components algorithm called heuristic, actually simple says, the heuristic means to find out the unknown field to explore through the trial. Heuristic was firstly put forward in the paper by Fred Glover, where the meta is to represent the implementation or beyond a higher level, and its ability to execute are usually better than the simple heuristic.

In addition, almost all of the meta-heuristic algorithms are used in random selection and local search. In a reasonable period of time, to solve the complex optimization problems, looking for a high quality solution, but cannot guarantee that the best solution can be achieved. Almost all of the meta-heuristic algorithms are suitable for global optimization problems.

All heuristic algorithms are composed of intensification and diversification, they have the ability of exploration and exploitation Diversity represents a lot of different solutions in the global scope of exploration. Intensification is represents concentrated in a local scale to search out the current best solution.

In order to improve the solution quality and convergence speed of the algorithm, it is needed to well balance the intensification and diversification. The best solution is to guarantee the convergence to the global optimal, by random search in local optimal solution space to the diversity of the solution. If we can achieve a good combination of these two components, it is usually able to get the global optimal solution.

1.2 Evolutionary algorithm

Evolutionary algorithms (EA) is a cluster algorithm, although there was a lot of change, with different genetic expression patterns, different crossover and mutation operator, reference to a special operator, different regeneration and selection method, but their inspiration from nature of biological evolution. The traditional compared with a calculus method and enumerative method based on OA , evolutionary computation is a mature with high robustness and applicability of the global optimization

method, with self-organizing, adaptive, self-learning characteristics to limit is not affected by the nature of the problem and effectively deal with the traditional OAs to solve complex problems.

Evolutionary computation is a kind of search algorithm based on natural selection and natural genetic mechanism. As the same with the ordinary search method, evolutionary computation is a kind of iterative algorithm, the different is evolutionary computation in the optimal solution search process, usually from a set of the original problem solution of improvement to another group of good solution, from the improved solution of further improvement.

And in the evolution of the problem, when the optimization model of the original problem is established, the solution of the original problem must be coded. Evolutionary computation in the search process using structured and random information, so that the most satisfied with the goal of the decision to obtain the maximum survival possible, is a probabilistic algorithm.

Evolutionary computation includes 4 typical methods of GA, genetic programming (GP), evolutionary strategy (ES) and evolutionary programming (EP). The first kind of method is more mature, and has been widely used. The application of evolutionary strategy and evolutionary programming in scientific research and practical problems is more and more widely.

The main genetic operation of GA are selection, crossover and mutation, and in the evolution rule and strategy, the evolution mechanism of source on selection and mutation.

In terms of fitness, GAs are used to select good parent generation (excellent offspring generation), and evolutionary rules and evolutionary strategies are used to select offspring.

Evolutionary rules and evolutionary strategies are generally not used to encode, eliminating the process of encoding and decoding procedures are more suitable for continuous optimization problems, but it cannot be non-numerical optimization. Evolutionary strategies can be used to determine the mechanisms that produce the parent generation for reproduction, and the GA and evolutionary rules emphasize the de-

pendence of individual fitness and probability.

In addition, evolutionary rules are abstracted to the similarity between populations, and the evolutionary strategy is the similarity between individuals. Evolutionary strategies and evolutionary rules have been applied to many fields of continuous function optimization, pattern recognition, machine learning, neural network training, system identification and intelligent control.

The EA is based on Darwin's theory of evolution, through the simulation of biological evolution process and mechanism of the problem of self-organizing, adaptive artificial intelligence technology. Biological evolution is realized through reproduction, variation, competition and selection, and the EA is mainly to solve the optimization problem by selecting, recombination and mutation of these three kinds of operations.

Evolutionary algorithms, including GAs, genetic programming, evolutionary programming and evolutionary strategies, etc.. The basic framework of the EA is a simple GA described in the framework, but there is a big difference in the way of evolution, selection, crossover, mutation, population control, there are many changes.As with GA. The convergence of EAs have some results and literature proved in preserving the best individual general evolutionary computation is convergence, but much of EAs results from GA to calculated.

GA is more important to crossover operation, which is considered that the mutation operation is an auxiliary operation of the algorithm. However, Evolutionary programming and evolutionary strategies are considered that the crossover is not superior to variation in the general sense, even may not cross operation.

Evolutionary computation is a kind of robust method, which can be adapted to different environments, and the effective solution can be obtained in most cases. It gives a coding scheme for the whole parameter space of the problem, but there is not directly on the specific parameters of the problem, and it is not from a single initial point to start the search, rather than from a set of initial point search. Search in the use in value of the objective function for information, we cannot use the objective function of the derivative information or specific issues related to the special knowledge. EAs have a widely range for applications, highly nonlinear, easy to modify and

parallelism.

In addition, the algorithm can use dynamic adaptive technology also by itself, in the evolutionary process of automatic adjustment algorithm control parameters and coding precision, such as the use of fuzzy adaptive method.

1.3 Computational Intelligence

Since the introduction of computer artificial intelligence (AI) has been the goal of the pursuit of computer scientists. As an important field of AI, computational intelligence (CI) because of its intelligence, parallel and robust, with good adaptive ability and strong global search ability, get the wide attention of many researchers, has in algorithm theory and performance made breakthrough progress, and has been widely used in various fields, plays an important role in the scientific research and production practice.

Computing intelligence research in the optimization calculation, pattern recognition, image processing, automatic control, economic management, mechanical engineering, electrical engineering, communication networks and biomedical etc. Multiple has been made in the field of successful application, the application relates to all aspects of national defense, science and technology, economy, industry and agriculture.

Computational intelligence is based on the idea of biological evolution. According to this view, intelligence is produced in the biological genetic, variation, growth and the natural selection of the external environment. In the process of using the waste back, the survival of the fittest, the high degree of adaptation (mind) structure is preserved, the intelligent level is also improved. So CI is based on the structure of the evolution of intelligence.

The main methods of CI are ANNs, GAs, GP, EP, local search, SA and so on. These methods have the following common elements: adaptive structure, randomly generated or specified initial state, adapt to evaluation function, modify the structure of the operation, system state memory, termination condition are calculated, indicating the method and control process parameters. These methods have the ad-

vantages of self-learning, self-organization, self adaptation, simple, general, robust, and suitable for parallel processing. It is widely used in the aspects of parallel search, associative memory, pattern recognition, knowledge acquisition and so on.

Typical such as GA, immune algorithm, SA algorithm, ant colony algorithm, PSO is a kind of bionic algorithm based on "from nature get the concept of wisdom", through the people's cognition of nature unique rules extracted for acquiring knowledge of a set of computational tools. In general, through the characteristics of adaptive learning, these algorithms achieve the goal of global optimization.

Chapter 2

Artificial Neural Network

2.1 Introduction

Artificial Neural Network (ANN) is a research hotspot in the field of artificial intelligence since the 1980s. It abstracts the neural network of human brain from the information processing point of view, builds a simple model and composes different networks according to different connection modes. In engineering and academia also often referred to directly as neural network or neural network. Neural network is a computing model, by a large number of nodes (or neurons) constitute a link between each other. Each node represents a specific output function, called the activation function. The connection between every two nodes represents a weighted value of the signal passing through the connection, called the weight, which is equivalent to the memory of the ANN. The output of the network depends on how the network is connected, the weights, and the incentive functions. The network itself is usually a natural algorithm or function approximation, it may be the expression of a logical strategy.The basic structure of neural network shown in Fig 2.1. In the past ten years, the research of ANN has made great progress. It has successfully solved many problems in the field of pattern recognition, intelligent robot, automatic control, prediction and estimation, biology, medicine and economy. Modern computer is difficult to solve the practical problems, showing a good intelligent characteristics. The basic model of artificial neural network as shown in Fig 2.2.

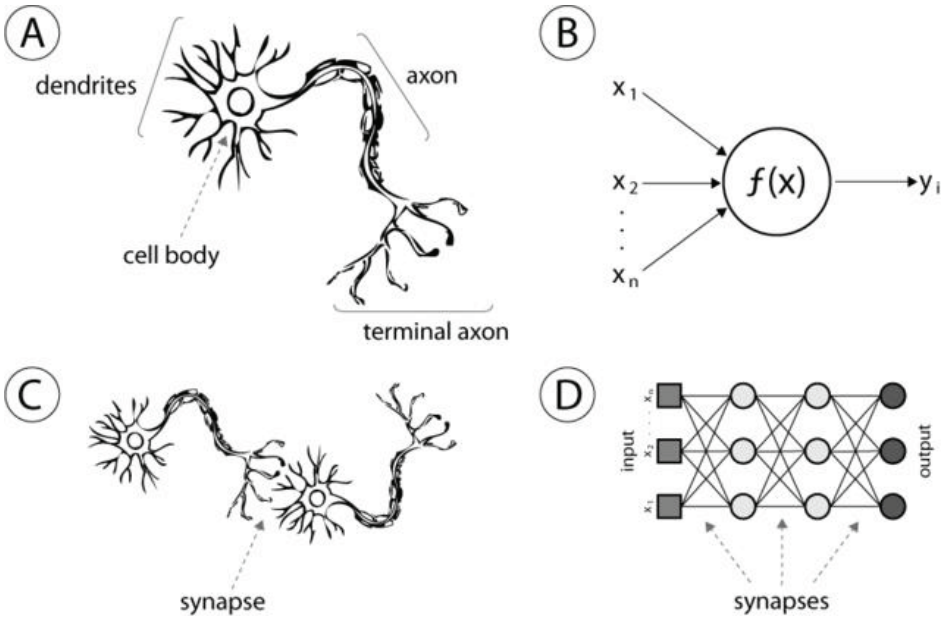


Figure 2.1: Evolution of Artificial Neural Networks

2.1.1 Basic characteristics

In an ANN, a neuron processing unit may represent a different object, such as a feature, a letter, a concept, or some meaningful abstract pattern. Types of processing units in the network are divided into three categories: input unit, output unit and hidden unit. The input unit accepts the signals and data from the outside world; the output unit realizes the output of the system processing result; the hidden unit is the unit between the input and output units, which cannot be observed from the outside of the system. The connection weight between the neurons reflects the connection strength between the cells. The representation and processing of the information are reflected in the connection relation of the network processing unit. ANN is a non-procedural, adaptive, brain-style information processing, its essence is through the network transformation and dynamic behavior of a parallel distributed information processing functions, and at different levels and levels to imitate people The information processing function of the cranial nervous system. It is involved in neuroscience, thinking science, artificial intelligence, computer science and other fields of interdisciplinary. ANN is a parallel distributed system. It adopts a completely

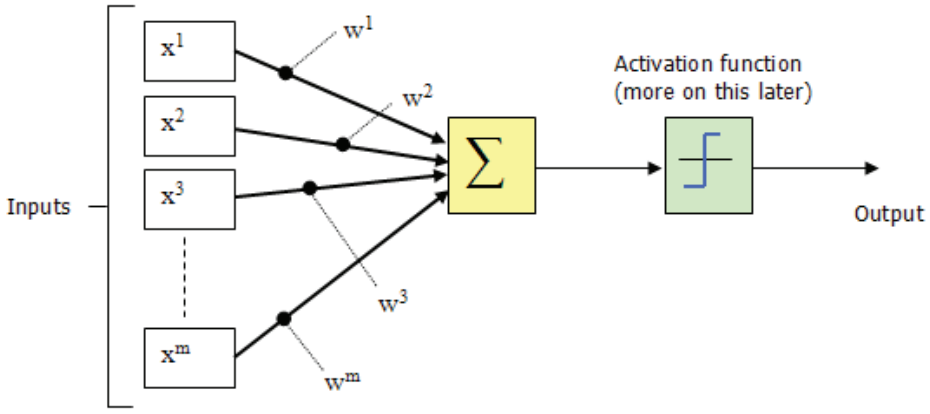


Figure 2.2: Artificial Neural Network Model

different mechanism from traditional artificial intelligence and information processing technology. It overcomes the shortcomings of traditional artificial intelligence based on logical symbols in dealing with intuitionistic and unstructured information, Self-organization and real-time learning. ANN is a nonlinear and adaptive information processing system composed of a large number of processing units interconnected. It is based on the results of modern neuroscience research, and attempts to process information by simulating neural network processing and memory information. ANN has four basic characteristics:

1. Nonlinearity: Non-linear relationship is the universal nature of nature. The wisdom of the brain is a non-linear phenomenon. Artificial neurons in the activation or inhibition of two different states, this behavior in the mathematical performance of a nonlinear relationship. Networks with threshold neurons have better performance and can improve fault tolerance and storage capacity.

2. Non-limiting: A neural network is usually connected by multiple neurons. The overall behavior of a system depends not only on the characteristics of a single neuron, but also on the interplay between the units. Simulating the non-limitation of the brain through a large number of connections between cells. Associative memory is a typical example of non-limitation.

3. Non-qualitative: ANN with adaptive, self-organizing, self-learning ability. Neural networks not only deal with the information can have a variety of changes, but also in dealing with information at the same time, nonlinear dynamic system itself is constantly changing. The iterative process is often used to describe the evolution of dynamical systems.

4. Non-convexity: The direction of evolution of a system, under certain conditions, will depend on a particular state function. Such as the energy function, its extreme value corresponds to the system is relatively stable state. Nonconvex is that this function has multiple extremes, so the system has a number of more stable equilibrium state, which will lead to the evolution of the diversity of the system.

2.1.2 Types of learning

Learning is an important part of neural network research, and its adaptability is achieved through learning. According to changes in the environment to adjust the weights to improve the system behavior. The Hebb learning rules proposed by Hebb laid the foundation for the learning algorithm of neural networks [13]. Hebb rule that the learning process occurs in the neurons between the synaptic sites, synaptic contact strength with the activities of the neurons before and after the synaptic changes. On this basis, a variety of learning rules and algorithms are proposed to meet the needs of different network models. The learning algorithm can make the neural network construct the internal representation of the objective world by adjusting the weight of the connection, and form the characteristic information processing method. The information storage and processing are embodied in the network connection. According to the different learning environment, neural network learning can be divided into supervised learning and unsupervised learning. In the supervised learning, the training sample data is added to the network input terminal, and the corresponding expected output is compared with the network output to obtain the error signal, thereby controlling the adjustment of the weight connection strength, and after repeated training converges to one determined weights. When the sample

situation changes, by learning to modify the weights to adapt to the new environment. The use of supervised learning neural network model has anti-transmission network, sensor and so on. Unsupervised learning, not given a standard sample in advance, the network directly into the environment, the learning stage and working stage into one. At this time, the change of learning rule follows the evolution equation of connection weight. The simplest example of unsupervised learning is the Hebb learning rule. Competitive learning rules are a more complex example of unsupervised learning, which is based on established clustering for weight adjustment. Self-organizing map, adaptive resonance theory network and so on are the typical models related to competitive learning.

2.1.3 Superiority of ANN

The characteristics and superiority of ANN are mainly manifested in three aspects:

1. Self-learning ability: For example, when image recognition is implemented, only a number of different image templates and the corresponding recognition results are input into the ANN, and the network learns to recognize similar images by self-learning function. The self-learning function is of particular importance for prediction. Expected future ANN computer will provide human economic forecasting, market forecasting, prediction efficiency, its application prospects are very ambitious.

2. associative storage capacity: This association can be achieved with a feedback network of ANNs.

3. High-speed look for the ability to have solutions: In order to find the optimal solution of a complex problem, we often need a large amount of computation. Using a feedback ANN designed to solve the problem, we can find the optimal solution quickly by using the high-speed computing capability of the computer.

2.1.4 Application analysis

Neural network has been applied in many fields, but it needs to be studied in many aspects. The combination of neural networks with other techniques, such as distribut-

ed memory, parallel processing, self-learning, self-organization, non-linear mapping, and the hybrid method and system have become a hotspot. Because other methods have their own merits, neural network and other methods are combined to learn from each other, and then can get better application effect. At present, this work has the integration of neural network and fuzzy logic, expert system, genetic algorithm, wavelet analysis, chaos, rough set theory, fractal theory, evidence theory and gray system. After decades of development, neural network theory in pattern recognition, automatic control, signal processing, decision support, artificial intelligence and many other research fields have achieved broad success. The following describes the application status of neural networks in some areas.

2.1.4.1 Application of ANN in Information Field

1. Information processing: The problem of modern information processing is very complex. ANN has the function of imitating or replacing with the thinking of human being. It can realize automatic diagnosis, solve the problem and solve the problem that traditional method can not or difficult to solve. ANNs have high fault tolerance, robustness and self-organization. Even if the connection lines are damaged to a great extent, it is still in the state of optimization, which is widely used in the military system electronic equipment application. The existing intelligent information systems are intelligent instruments, automatic tracking and monitoring instrument system, automatic control guidance system, automatic fault diagnosis and alarm systems.

2. Pattern recognition: Pattern recognition is the process of describing and identifying, classifying, and interpreting things or phenomena by processing and analyzing the various forms of information that characterize things or phenomena. The technology is based on the theory of Bayesian probability theory and Shennong's information theory. The process of information processing is closer to the logical thinking process of the human brain. There are two basic pattern recognition methods, namely statistical pattern recognition method and structure pattern recognition method. ANN is a common method in pattern recognition. The recognition method of

ANN, which is developed in recent years, gradually replaces the traditional pattern recognition method. After years of research and development, pattern recognition has become a more advanced technology, is widely used in text recognition, speech recognition, fingerprint recognition, remote sensing image recognition, face recognition, handwritten character recognition, industrial fault detection, precision guidance aspect.

2.1.4.2 Application of ANN in Economic Field

1. Market price forecast: Analysis of changes in commodity prices can be attributed to the impact of market supply and demand of the many factors of comprehensive analysis. The traditional statistical economics method is difficult to make the scientific forecast to the price fluctuation because of its inherent limitation, and the ANN is easy to deal with the incomplete, fuzzy uncertain or the regularity not obvious data, therefore uses the artificial nerve network to carry on price forecasting is a traditional method can not be compared to the advantage. Based on the market price determination mechanism, a more accurate and reliable model is established based on the factors such as the number of households, the disposable income per capita, the loan interest rate and the urbanization level, which influence the commodity price. The model can predict the trend of commodity price and get accurate and objective evaluation results.

2. Risk assessment: Risks refer to the possibility of economic or financial loss, natural destruction or damage arising from the uncertainties in the conduct of a particular activity. The best way to prevent the risk is to make a scientific risk forecast and assessment. The forecasting idea of ANN is to construct the structure and algorithm of credit risk model according to the realistic risk source, get the risk evaluation coefficient, and then determine the solution to the practical problem. The empirical analysis of the model can make up for the lack of subjective assessment, and can achieve satisfactory results.

2.1.4.3 Application of ANN in Medicine

Because of the complexity and unpredictability of the human body and the disease, the detection and signal expression of biological signal and information manifestation, change rule (self-change and medical intervention), data and information analysis, Decision-making and many other aspects of the existence of a very complex non-linear relationship, suitable for the application of ANNs [14]. The current research involves almost everything from basic medicine to clinical medicine, mainly used in biological signal detection and automatic analysis, medical expert system and so on. Although the ANN has made some progress, but there are still many shortcomings, such as: the application of the surface is not wide enough, the result is not accurate enough, the existing model algorithm training speed is not high enough, and the algorithm integration is not high enough. At the same time, we hope to find a new breakthrough in theory, the establishment of new generic models and algorithms. We need to further study the biological neuron system, and constantly enrich the understanding of the human brain.

2.2 Conclusion

ANN has a preliminary adaptive and self-organizing ability. The synaptic weights are changed during the learning or training process to suit the requirements of the surrounding environment. The same network can have different functions because of different learning methods and contents. ANN is a learning ability of the system, can develop knowledge, so that more than the designer's original level of knowledge. In general, its learning and training methods can be divided into two kinds, one is to monitor the learning, then use the given sample criteria for classification or imitation. Another one is unsupervised learning, only provides the learning mode or some rules, the specific learning content with the system environment (the input signal situation), the system can automatically find the environmental characteristics and regularity, with more similar to the function of the human brain.

ANN is like a love of learning children, you teach her knowledge she will not forget and will apply their knowledge. We add each input in the learning set to the ANN and tell what the neural network output should be. After all the learning sets have been run, the ANN sums up her own ideas based on these examples. Then we can put the test set in the test case using a ANN for testing respectively, if testing (such as 80% or 90% of the correct rate), then the neural network is constructed success. After this we can use ANNs to determine the classification of the transaction.

The ANN is the human brain through the base unit (neurons) and coupled modeling, simulation model to explore the human brain system functions, and to develop a kind of learning, associative memory and pattern recognition intelligent information processing systems. An important characteristic of neural networks is that it can learn from the environment, and store the distribution of the learning results in the network's synaptic connections. ANN learning is a process, in which the environment under the excitation, the network need to some input samples, and in accordance with certain rules (learning algorithm) adjust the weight matrix of the network layer until the network layer weights are convergent to a certain value, the learning process is over. Then we can use the generated ANN to classify the real data.

Chapter 3

Chaos

3.1 Introduction

Chaos is to determine the initial state of the long-term behavior of power system, or system parameters are very sensitive, but not divergence and cannot be precisely repeated phenomenon, it is nonlinear systems generally have a complex dynamic behavior [15, 16]. Chaotic variables seem to be chaotic change process, in fact, it contains the inherent regularity. By using the random, ergodic and regularity of chaotic variables can optimize the search. The basic idea is to put the chaotic variables mapping to the value of the optimization variable, then using chaotic variables to search. However, the algorithm has a long computation time and cannot search the optimal solution in the large space and multi variable optimization search. Therefore, we can use a class of chaotic maps with infinite number of folds to generate chaotic variables, and select the search space of the optimization variables, and constantly improve the search accuracy and other methods to solve such problems.

Chaos is one of the important branches of nonlinear science. It is a kind of singular steady-state behavior of nonlinear dynamical systems. It represents the essential feature of a complex phenomenon in nature and human society. Therefore, chaotic science advocates Shlesinger and famous physicist Ford and a large number of chaotic scholars that chaos is the third biggest physics revolution in the 20th century, the first two is the theories of quantum mechanics and relativity, chaos optimization is chaotic discipline in the face of engineering application in the field of an important

research direction. Its application specific is the use of chaotic motion characteristics, to overcome the defect of traditional optimization methods, so that the optimization results to achieve better.

Chaos also is nonlinear dynamical systems with special form of a motion, it widely exists in the nature, such as physics, chemistry, biology, geology, technology science, social science and other fields of science. Due to the chaotic characteristic: deterministic, boundedness, extreme sensitivity to initial conditions, long-term unpredictability and so on, it is becoming more and more widely applied, at present mainly used include: image data encryption, secure communications, computer graphics processing, signal processing and communications, control system and optimization and direction in the field of more and more. Among them, the chaos application in information security is the last ten years has aroused great concern of scholars, and has been widely used in the field of information security, and has achieved some good results.

3.2 Lyapunov exponent

Lyapunov exponent(LE), the basic characteristics of chaos system is the system extreme sensitivity to the initial value, the two produced trajectory by the same initial value, with the passage of time according to exponential mode separation, LE is the quantitative description of this phenomenon [17]. LE is an important quantitative indicator of a measure of the system dynamic characteristics, which represents the average index of convergence or divergence between adjacent orbits in phase space. For whether there is a system dynamic chaos, whether from the maximum LE greater than 0 is intuitive judgment: a positive LE, which means in system phase space, no matter how small the spacing of the two initial trajectory, the difference will in an index increase with time evolution so that to achieve the impossible to predict, which is the chaotic phenomenon. The sum of LE represents the spheroid volume growth or decreases rate, on the Hamiltonian system, which sum is zero; on the dissipative system, which sum as negative. If the dissipative systems attractor is a fixed point, then whole LE is generally negative. If it is a simple m dimensional manifold (m=1

or m=2, for a curve or surface respectively), then the first m LE is 0, and the rest of the LE is negative. Regardless of the system is dissipative, if LE greater than 0 there will be a chaos. When the LE is less than 0, which means that the adjacent points to be moved merging into one point finally. If the LE is greater than 0, which means that the adjacent points will be separated finally. The LE increases, that chaos is more obvious, the higher the degree of chaos.

3.3 Chaotic characteristics

From the view of phenomena, the chaotic motion seemingly random process, but in fact, there are essential differences between the chaotic motion and random process. Chaos is a deterministic algorithm similar stochastic process, it is different from general randomness, which refers to the nonlinear algorithm without the influence of external random factors, An intrinsic random generated by the sensitive algorithm of initial value.

Chaos phenomenon is a deterministic pseudo random process in the algorithm of nonlinear dynamics. This process is not periodic, the overall stability and expansionary in local. Chaos is the inherent randomness of the determined nonlinear algorithm. Namely, chaos is the inherent algorithm, which is shown by the complexity of the algorithm itself for inherent factors, and not generated in the external disturbance. Chaotic motion is caused by the inherent characteristics of the physical law of certainty, which is derived from the external performance of the intrinsic characteristic, therefore, it is also called the deterministic chaos, and the random process is caused by the noise of the external characteristic.

3.3.1 Intrinsic randomness

Under the concept of the steady state of chaos is not usually determine the movement of the three states: static, periodic motion and quasi periodic motion, which is complex situation motion and always limited to finite area and tracks will never

repeat. First, chaos is inherent, the system is shown by the complexity of the system itself in the internal factors, and not in the external disturbance, which is the system's internal random performance. Second, the randomness of chaos is deterministic. Chaotic deterministic is divided into two aspects, firstly, the chaotic system is determined system; secondly, chaotic performance is seemingly random, and not the real randomized, system state of each moment are affected by the previous state is to determine, not as casual as stochastic system, the state of chaotic system can be completely recreate and stochastic systems are different. Third, the performance of chaotic system is complex, which is not a periodic motion, nor is it a quasi periodic motion, but has a good self correlation and low frequency broadband characteristics.

3.3.2 Long-term unpredictability

Due to the initial conditions are limited to a finite precision, and the tiny difference of the initial conditions can have a great influence on the later time evolution, therefore cannot be long-term forecast dynamic characteristics at some point in the future. Long-term evolution behavior of chaotic system is unpredictable.

3.3.3 Sensitive dependence on initial value

As time goes on, all initial conditions will show their independent time evolution, which is sensitive to the initial conditions. Even if the initial data is very small deviation, after several iterations, the gap will be great.

3.3.4 Universal applicability

When the system is tend to more chaotic and characteristics exhibited by the universal applicability, the system does not change because of the differences in the concrete system different and the motion equation of the system, even different chaotic map, the chaotic state in appearance is similar.

3.3.5 Fractal character

Fractal is a new word which is used by B.B.Mandelbrot in 70s to create fractal geometry. So-called Fractal refers to the n dimensional space a geometric properties of a set of points, they have unlimited fine structure, under any scale there are self similar and integral similar properties, which are non integer dimension less than the space dimension n. this point set is called fractal objects. Fractal dimension is a non-integer dimension, which to quantitatively describe the basic characteristics of the fractal.

3.3.6 Ergodic property

Ergodic property is also known as mixture. Because chaos is a kind of sexual complex movement which is always confined to a limited area and the orbit is never repeated. Therefore, with the passage of time, the trajectory of the chaotic motion will never stay in a certain state but traverse every point in the regional space, in other words, as long as the time is sufficiently long, chaos will not be repeated to walk through every point.

3.3.7 Boundedness

Its movement trajectory has always been confined to a certain region, the region known as the chaotic domain of attraction. So in general, the chaotic system is stable.

3.3.8 Fractal dimension

The running state of chaotic system has a multi leaf and multi layer structure, and the leaf layer is divided into more and more detailed, showing an infinite level of self similar structure.

3.3.9 Statistical property

For the chaotic system, positive Lyapunov exponents show that trajectories in each locality are unstable and adjacent to the track according to the exponential separation. However, due to the boundedness of the attractor, the orbit can be folded repeatedly in a confined region, but never intersect, forming a special structure of the chaotic attractor.

3.4 The application of chaos theory

Chaos theory has been widely used in both natural science and social science, and its potential application can be summarized as follows:

1. **Optimization**: Using the randomness, ergodicity and regularity of chaotic motion to find the most advantages, can be used in many aspects such as system identification, optimal parameter design and so on [18, 19].

2. **Neural network**: Put the chaos and neural network integration, so that the neural network by the initial chaotic state gradually degenerated into a general neural network, using the dynamics of chaotic state intermediate process neural network escape from local minimum, so as to guarantee the global optimum, it can be used for associative memory, robot path planning, etc. [20].

3. **Image data compression**: The complex image data with a group can produce a simple dynamic equations of the chaotic attractor, so only memory to store a set of parameters of dynamics equations, the amount of data than the original image data is greatly reduced, in order to achieve the image data compression.

4. **High-speed search**: Using the ergodicity of chaos can be retrieved, namely change initial value at the same time, The data to be retrieved is compared with the values that have just entered the chaotic state, Retrieves the state of the data which is close to the stay retrieved data, this method has higher retrieval speed than random search or genetic algorithm.

5. **Nonlinear time series prediction**: Any time series can be as a determined

by a nonlinear mechanism of input and output system, if the irregular movement phenomenon is a kind of chaotic phenomena, nonlinear technology can forecasting accurately for short-term.

6. **Pattern recognition**: Using chaotic trajectory sensitivity to initial conditions, it is possible to identify the different modes of the system with only a tiny difference.

7. **Qualitative prediction of economic chaos and quantitative prediction of economic system**: Using chaos theory to study the economic and management issues, especially on stock market price index, exchange rate change.

3.5 Conclusion

The theory of relativity eliminates the illusion of absolute space and time, which is the Newtonian form of illusion. Quantum mechanics is eliminated the dream about controllable Newton type of measurement process [21]. The chaos is the elimination of Laplasse's theory of determinism in the predictability of fantasy. Chaos theory helps us to break the inherent thinking, and once again deep understanding of the world all the paradox between both opposite and unified dialectical relationship. It will guide us in the field of natural science and social science for further studies. At the same time, we should also take the initiative to combine the chaos theory with their own professional field, in order to have a new discovery and a new breakthrough.

Chapter 4

Multiple Chaos Embedded Gravitational Search Algorithm

4.1 Introduction

Meta-heuristics have been successfully and widely used for solving various optimization problems in the past decades [22]. A considerable number of meta-heuristics have been proposed based on metaphors of natural evolution, swarm mechanisms or man-made processes [23]. These meta-heuristics include evolutionary computation [24], particle swarm optimizations [25], ant colony algorithms [26], artificial immune systems [27], etc. Among them, gravitational search algorithm (GSA) which is inspired from the Newton's law of gravity and motion [28] has demonstrated to be a powerful optimization tool when applied to function optimization problems and many real-world problems [28–31].

Like other nature-inspired meta-heuristic algorithms, GSA is a population-based adaptive search technique. In GSA, a population of candidate solutions are modeled as a swarm of objects. At each iteration, the objects update their position by moving stochastically towards regions previously visited by the other objects. The object with heavier mass has a larger effective attraction radius and hence a greater intensity of attraction. By lapse of time, the objects tend to move towards the heaviest object. In comparison with other well-known optimization algorithms, such as the particle swarm optimization, GSA has been confirmed higher performance in searching ability

[28]. However, GSA still has some inherent disadvantages, such as it usually sticks on local optimal solutions, which indicates that it is unable to improve the solutions' quality in the latter search phases [32, 33].

Many attempts have been made to alleviate the inherent local minima trapping problem and further improve the search performance of GSA [33]. Li and Zhou [34] modified the velocity updating rule by utilizing the memory and social information of agents, aiming to accelerate GSA's convergence speed. An opposition-based learning rule was proposed for population initialization and generation jumping in GSA [35]. A niching GSA was proposed by dividing the main swarm of masses into several small sub-swarms to maintain the diversity of population [36], thus improving the performance of GSA for multimodel optimization problems.

Considerable effort has been devoted to incorporating chaos into meta-heuristics in recent years. Chaos is a universal phenomenon of nonlinear dynamic systems and it is apparently an irregular motion, seemingly unpredictable random behavior exhibited by a deterministic nonlinear system under deterministic conditions. Due to the ergodicity and dynamic properties of chaos, chaotic maps can help meta-heuristic optimization algorithms to enhance the diversity among individuals and avoid premature convergence. In the literature, chaotic maps have been incorporated into evolutionary algorithms [37], particle swarm optimization [38], biogeography-based optimisation [39], water cycle algorithm [40], fruit fly optimization [41], krill herd algorithm [42], bat algorithm [43], differential evolution algorithm [44], harmony search algorithm [45], firefly algorithm [46], ant swarm optimization [47], imperialist competitive algorithm [48], and others.

In our previous work [49], we proposed two kinds of chaos-based GSAs. One used chaotic sequences to substitute random numbers for different parameters, and the other used chaotic variables to perform a chaotic local search. Both embedding strategies of chaos were found to be benefit for improving GSA's search ability, and the latter seemed to be more efficient. The work [49] has been extended by considering five different chaotic maps. Preliminary experimental results in [50] empirically showed that all introduced five chaotic maps generally exhibited effectiveness of im-

proving the performance of GSA. Nevertheless, there is no specific chaotic maps can enable GSA to achieve the best solution for all optimization problems, suggesting that the performance of chaotic GSAs are related not only to the search capacity of the algorithm, but also to the landscape of the solved problems.

In this paper, we investigate the capability of chaotic local search using different chaotic maps and different incorporation strategies for improving the search performance of GSA. The motivation of this study comes from the following aspects. First, the effectiveness of the incorporation of chaos into GSA needs to be verified via extensive experiments. Second, as a number of chaotic maps is available, it is required to find out which one is the most appropriate for GSA. Third, a well-established embedding strategy which can fully utilize the search dynamics of chaos needs to be designed. Based on these considerations, we propose a multiple chaos embedded gravitational search algorithm (MCGSA) in this paper.

In all prior chaotic meta-heuristics [37–50], only a single certain chaotic map is embedded into the meta-heuristic algorithm to perform the chaotic search. Few research studies the integration of multiple chaotic systems which simultaneously perform the search. The prior chaotic GSA in [49] only utilized the well-known Logistic maps to realize the chaotic search, and the extended one [50] compared the performance difference during five chaotic maps. Obviously, the search capacity of such single chaos embedded GSA is limited. Multiple chaos can be expected to provide more dynamic properties for alleviating the local problem trapping problem of GSA. To realize these, we first construct twelve variants of single chaos embedded GSA using twelve different chaotic maps. Then, three novel multiple chaos embedded GSAs (MCGSA) are proposed, i.e., the twelve chaotic maps are (1) randomly, (2) parallelly, and (3) memory-selectively incorporated into GSA. The resultant chaotic GSAs are called CGSA-R, CGSA-P, and CGSA-M, respectively. Extensive experiments are conducted based on 48 widely used benchmark numerical optimization functions. Experimental results and statistical analysis verified that MCGSA can perform better than the traditional GSA and those single chaos embedded GSAs.

The remainder of this paper is organized as follows. Section 2 gives a brief de-

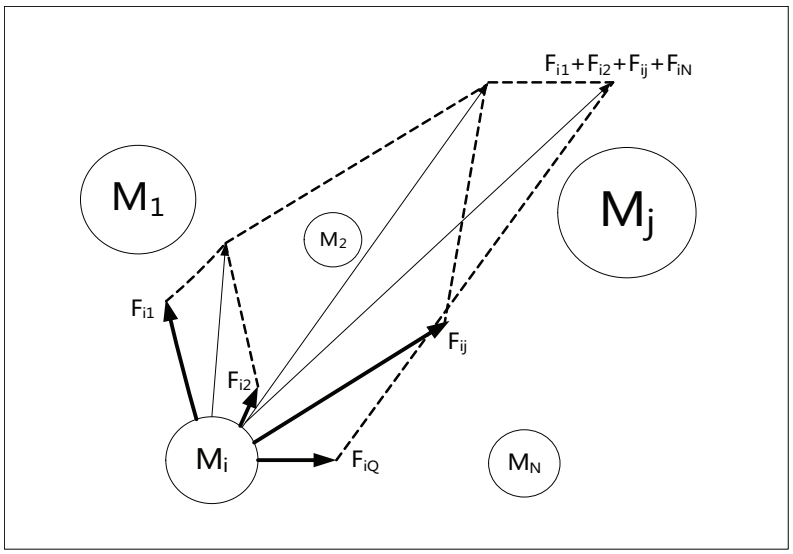


Figure 4.1: Gravitational force:each interaction force between every mass(agent)

scription of the traditional GSA. Section 3 summarizes the twelve chaotic maps used in this study. In Section 4, we first introduce the single chaos embedded GSA. Then three kinds of MCGSA are presented in details. Section 5 provides experimental results. Section 6 concludes the paper.

4.2 Brief Description of Traditional GSA

GSA is a population based meta-heuristic algorithm inspired by the law of gravity among objects. Each agent in the population of GSA is considered as objects and its performance is measured by its mass. The position of agent corresponds to a solution of the optimization problem needed to be solved. Moving the position of agent can result in an improvement of the solution's quality. The gravitational force between the two particles is proportional to the mass of the inertia of the two particles, and is inversely proportional to the square of the distance between the two particles.In Fig. 4.1 figure out the gravitational force between two particles.

Formally, every agent $X_i = (x_i^1, ..., x_i^d, ..., x_i^D)$, $(i = 1, 2, ..., N)$ attracts each other by gravitational forces in a D-dimensional search space, where x_i^d represents the position of i-th agent in the d-th dimension. The corresponding velocity of agent X_i

is expressed by $V_i = (v_i^1, ..., v_i^d, ..., v_i^D)$. The mass of each agent in iteration t, denoted by $M_i(t)$, is calculated via the map of its fitness as follows:

$$M_i(t) = \frac{\text{fit}(X_i(t)) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)} \tag{4.1}$$

where $\text{fit}(X_i(t))$ represents the fitness of agent X_i by calculating the objective function. For a minimization problem, $\text{best}(t)$ and $\text{worst}(t)$ are defined as

$$\text{best}(t) = \min_{j=1,2,...,N} \text{fit}(X_j(t)) \tag{4.2}$$

$$\text{worst}(t) = \max_{j=1,2,...,N} \text{fit}(X_j(t)) \tag{4.3}$$

The force acting on the i-th agent from the j-th agent is defined as:

$$F_{ij}^d(t) = G(t)\frac{M_i(t) \times M_j(t)}{R_{ij}(t) + \varepsilon}(x_j^d(t) - x_i^d(t)) \tag{4.4}$$

where $R_{ij}(t) = ||x_i(t), x_j(t)||_2$ is the Euclidean distance between two agents, and ε is a small constant, preventing the denominator in Eq. (4.4) from being zero. In addition, $G(t)$ is the gravitational constant at time t, defined by

$$G(t) = G_0 \exp(-\alpha \frac{t}{t_{max}}) \tag{4.5}$$

where G_0 is the initial value, α is a shrinking constant, t_{max} is the maximum number of iterations. For the i-th agent, the overall force that acts on it is a randomly weighted sum of the forces exerted from the surrounding agents.

$$F_i^d(t) = \sum_{j \in Kbest, j \neq i} \text{rand}_j F_{ij}^d(t) \tag{4.6}$$

where $Kbest$ is the set of first K agents with the best fitness and biggest mass, rand_j is a random number uniformly generated in the interval $[0, 1]$. Furthermore,

$$K = \lfloor (\beta + (1 - \frac{t}{t_{max}})(1 - \beta))N \rfloor \tag{4.7}$$

It is clear that K is initially set to N and is decreased linearly, which is controlled by a constant β. The operation $\lfloor \cdot \rfloor$ is the floor function. Based on the law of motion, the acceleration of the i-th agent is calculated by:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{4.8}$$

Then, the next velocity of an agent is considered as a fraction of its current velocity added to its acceleration. Therefore, its position and its velocity could be updated as follows:

$$v_i^d(t+1) = \text{rand}_i v_i^d(t) + a_i^d(t) \tag{4.9}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{4.10}$$

where rand_i is a random variable in $[0, 1]$. It should be noted that both rand_i and rand_j are uniformly generated random numbers, and they generally differ from each other. In fact they are an attempt of giving randomized characteristics to the search. The general principle of GSA flow chart is shown in Fig.4.2

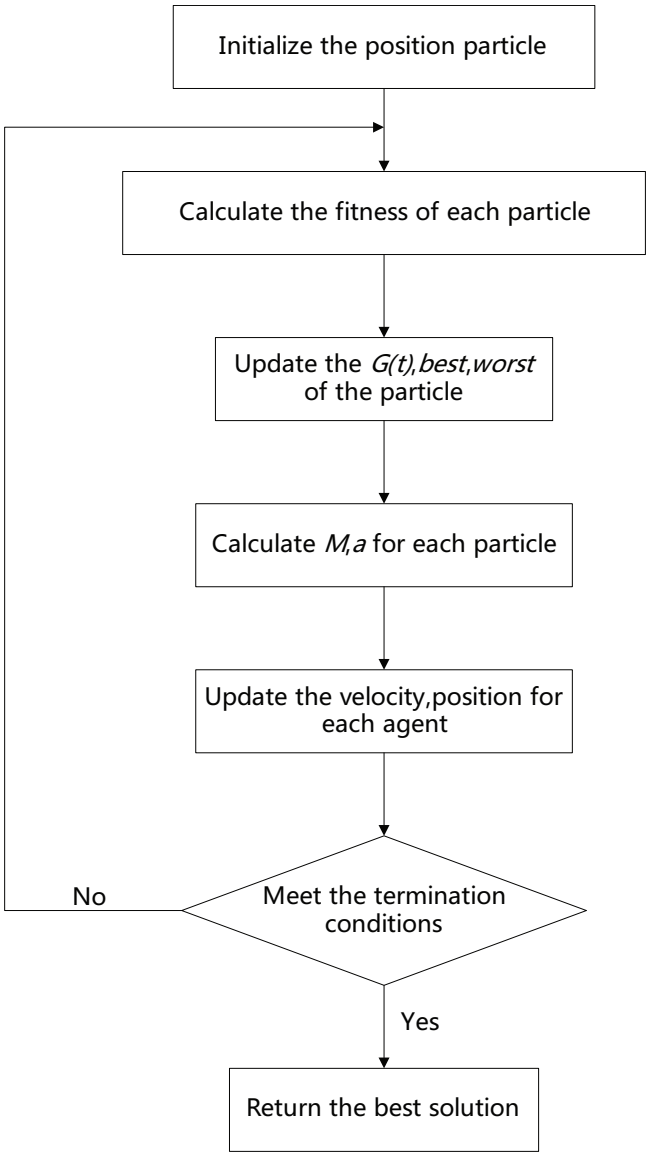


Figure 4.2: GSA general principle flow chart

4.3 Chaotic Maps

One dimensional non-invertible maps are the simplest systems with the capability of generating chaotic motion. In this study, twelve well-known one-dimensional chaotic maps widely used in related researches [37–48] are considered.

(1) Logistic map: this classic logistic map appears in nonlinear dynamics of biological population evidencing chaotic behavior, and can be written as in the following.

$$z_{k+1} = \mu z_k (1 - z_k) \tag{4.11}$$

where z_k is the kth chaotic number. Obviously, $z_k \in (0, 1)$ under the conditions that the initial $z_0 \in (0, 1)$ and that $z_0 \notin \{0.0, 0.025, 0.5, 0.75, 1.0\}$. In our experiment, we set $\mu = 4$ and the initial number $z_0 = 0.152$.

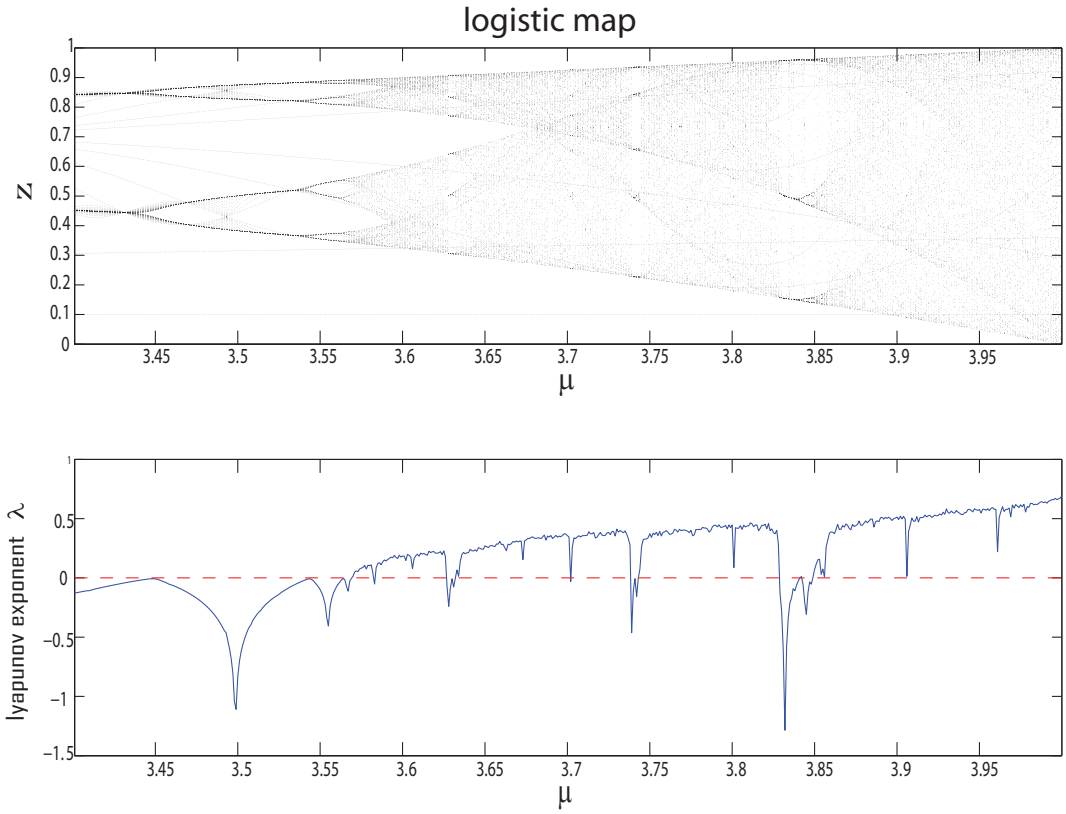


Figure 4.3: Lyapunov exponent of logistic map

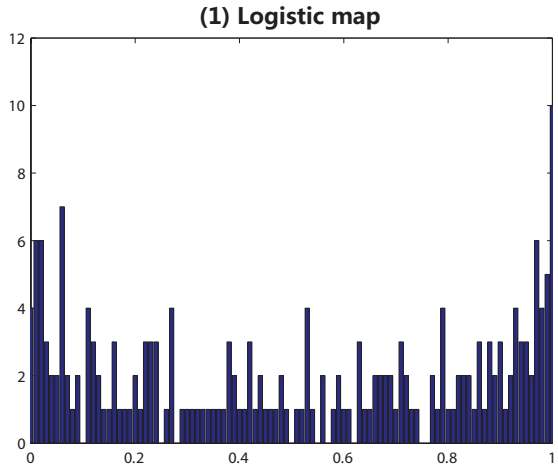


Figure 4.4: Chaotic PDF graph of logistic map

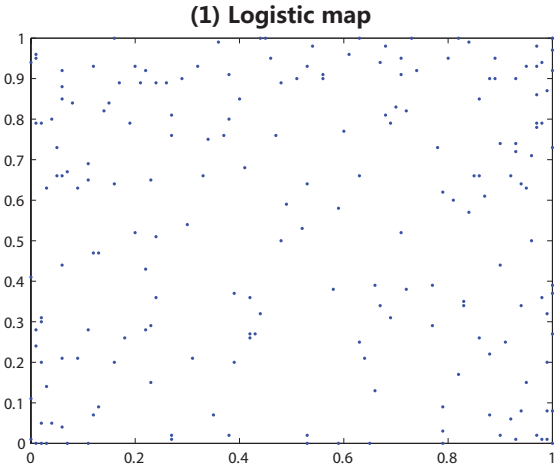


Figure 4.5: Chaotic 2-Dpoint distribution of logistic map

(2) Piecewise linear chaotic map (PWLCM): it has been known as ergodic and has uniform invariant density function on their definition intervals. The simplest PWLCM is governed by the following equation.

$$z_{k+1} = \begin{cases} z_k/p, & z_k \in (0, p) \\ (1 - z_k)(1 - p), & z_k \in [p, 1) \end{cases} \tag{4.12}$$

In the experiment, p is set to be 0.7 and $z_0 = 0.002$.

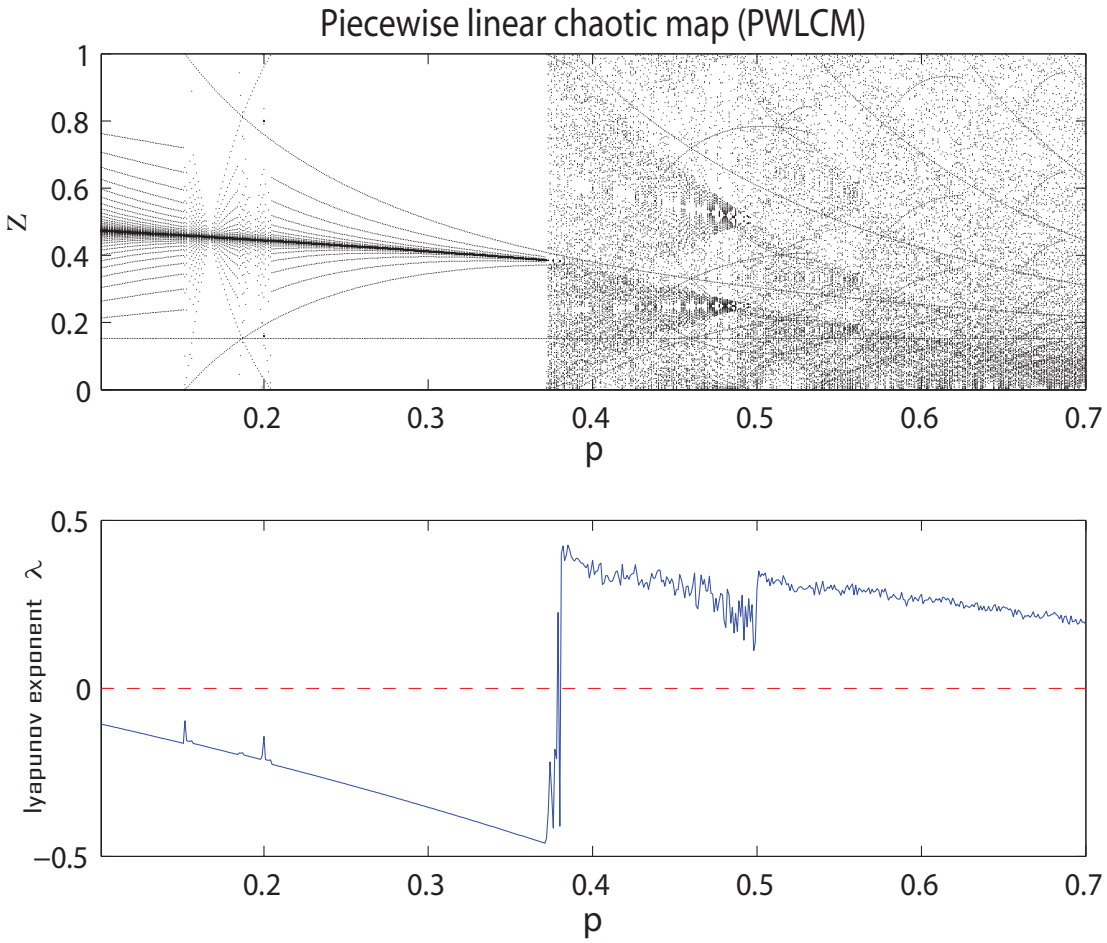


Figure 4.6: Lyapunov exponent of Piecewise linear chaotic map (PWLCM)

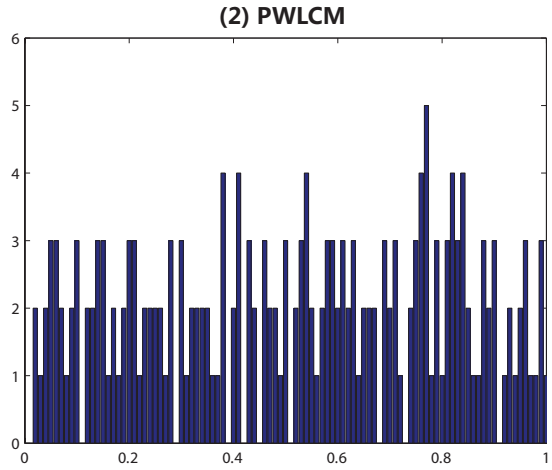


Figure 4.7: Chaotic PDF graph of Piecewise linear map

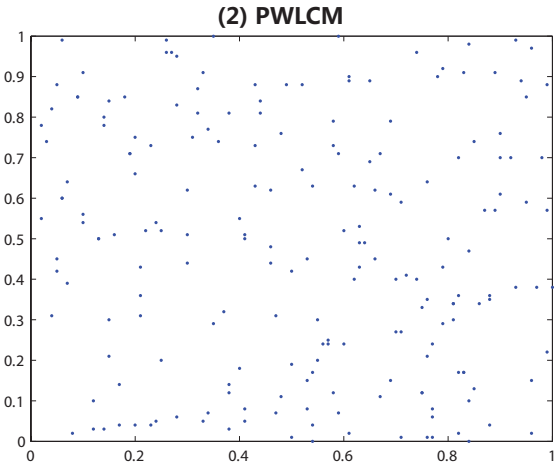


Figure 4.8: Chaotic 2-Dpoint distribution of Piecewise linear map

(3) Singer map: it is a one-dimensional system as given in the following.

$$z_{k+1} = \mu(7.86z_k - 23.31z_k^2 + 28.75z_k^3 - 13.302875z_k^4) \tag{4.13}$$

Singer map exhibits chaotic behaviors when the parameter μ is set as a value between 0.9 and 1.08. In this study, we set $\mu = 1.073$ and $z_0 = 0.152$.

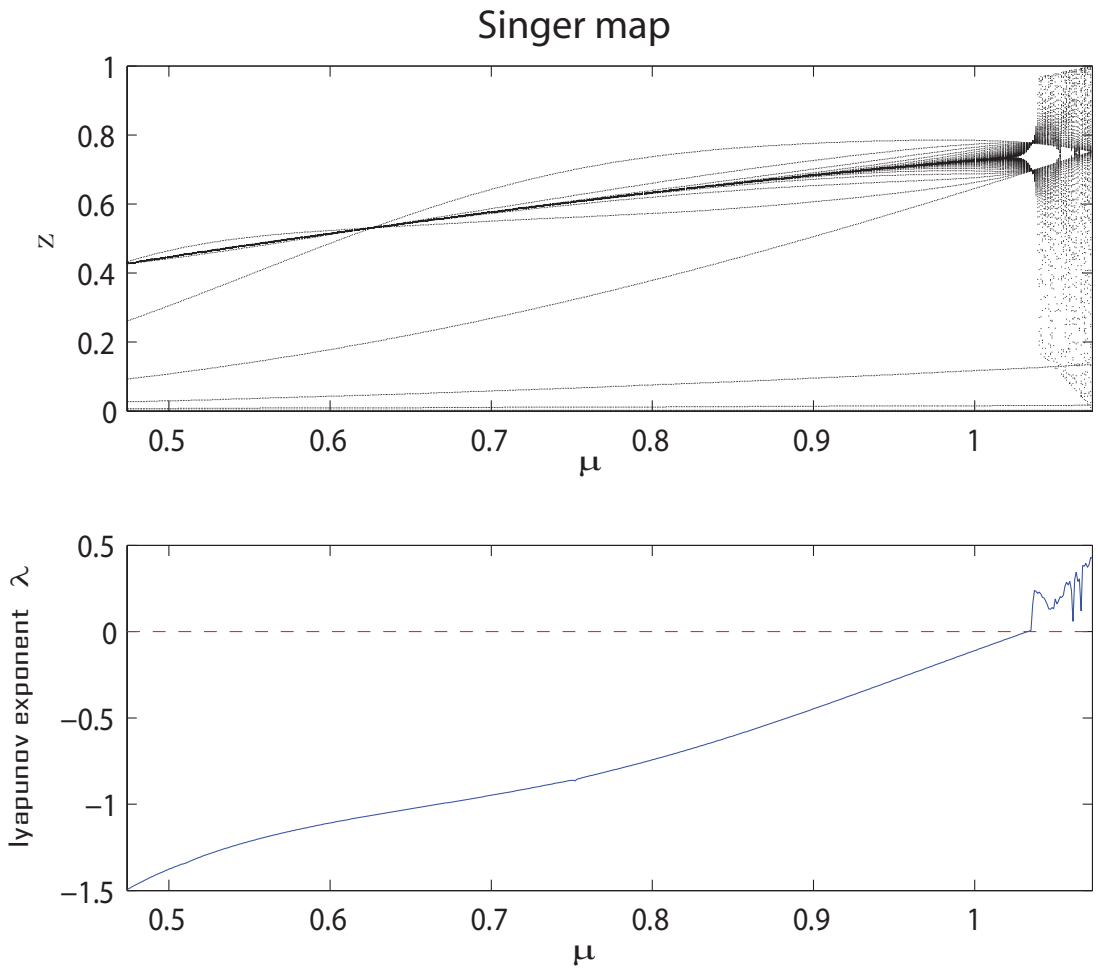


Figure 4.9: Lyapunov exponent of Singer map

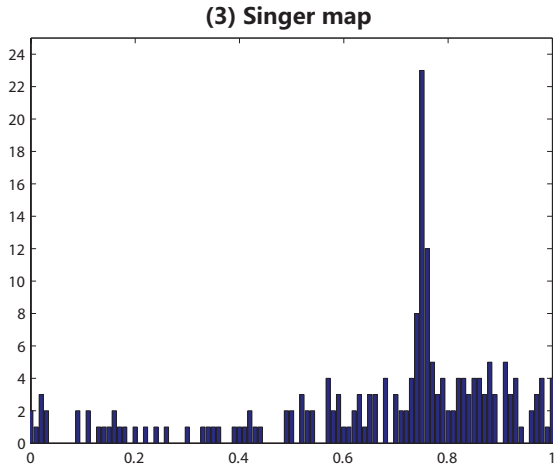


Figure 4.10: Chaotic PDF graph of Singer map

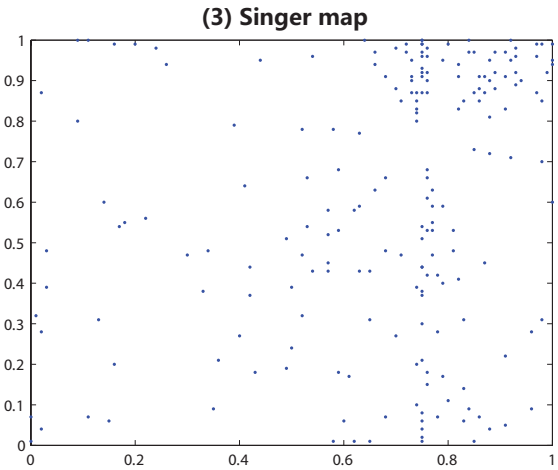


Figure 4.11: Chaotic 2-Dpoint distribution of Singer map

(4) Sine map: it belongs to a unimodal map which is similar to the Logistic map, can it is written as the following equation.

$$z_{k+1} = \frac{a}{4} sin(\pi z_k) \tag{4.14}$$

where the parameter $a \in (0, 4]$, and thus $z \in (0, 1)$. We set $a = 4$ and $z_0 = 0.152$ in the experiment.

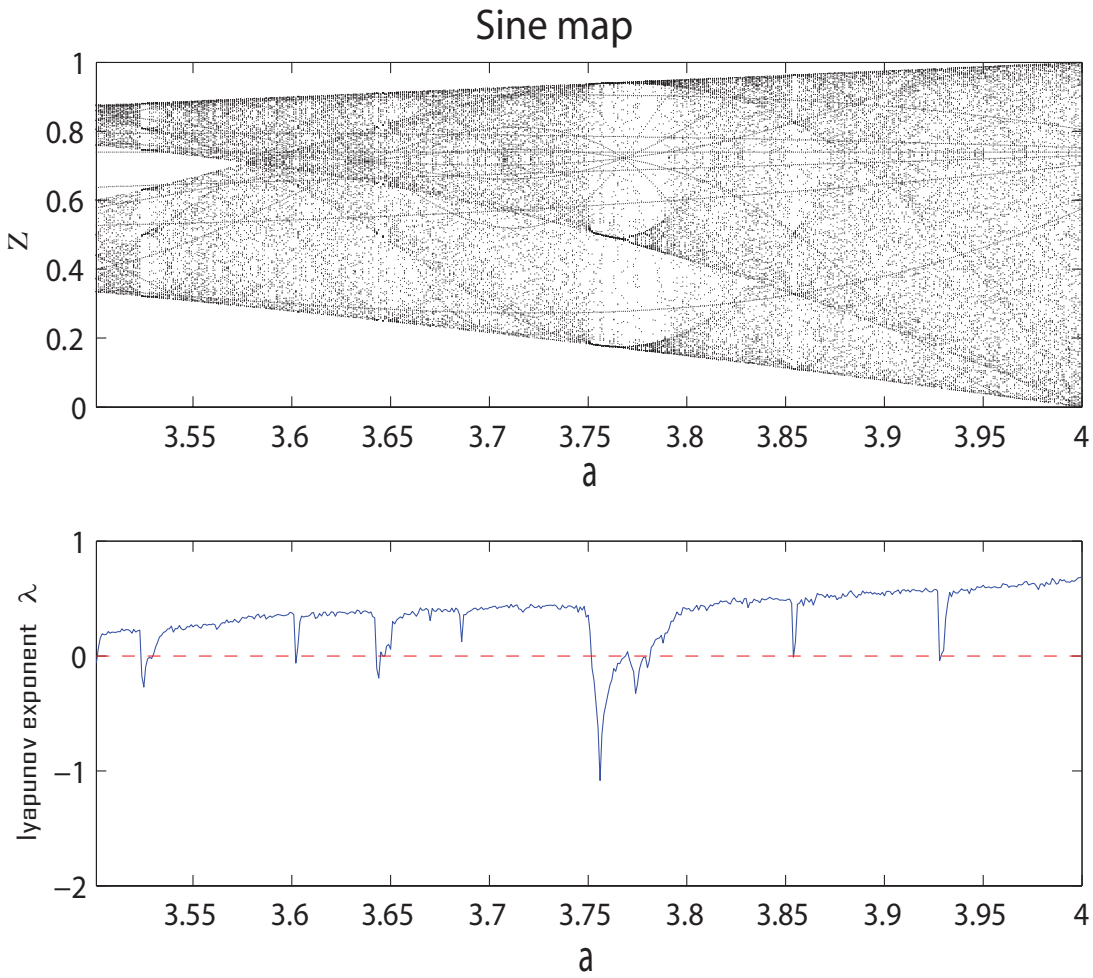


Figure 4.12: Lyapunov exponent of Sine map

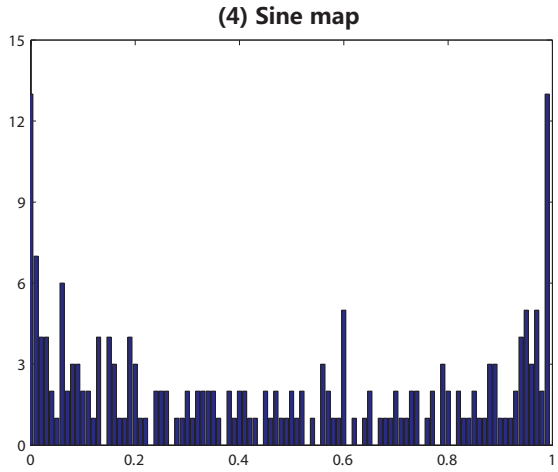


Figure 4.13: Chaotic PDF graph of Sine map

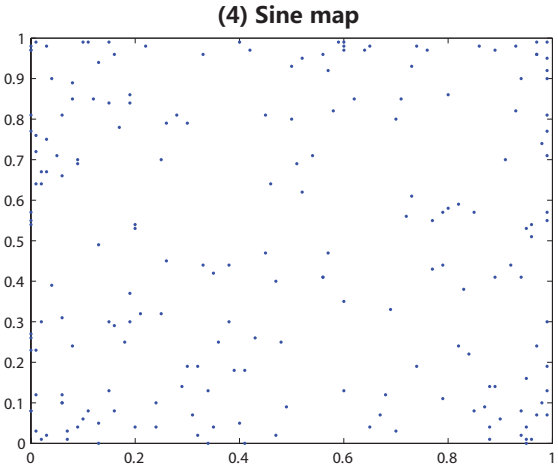


Figure 4.14: Chaotic 2-Dpoint distribution of Sine map

(5) Sinusoidal map: this iterator can be defined as

$$z_{k+1} = az_k^2 sin(\pi z_k) \tag{4.15}$$

where $a = 2.3$ and we set the initial number of this chaotic system as $z_0 = 0.74$.

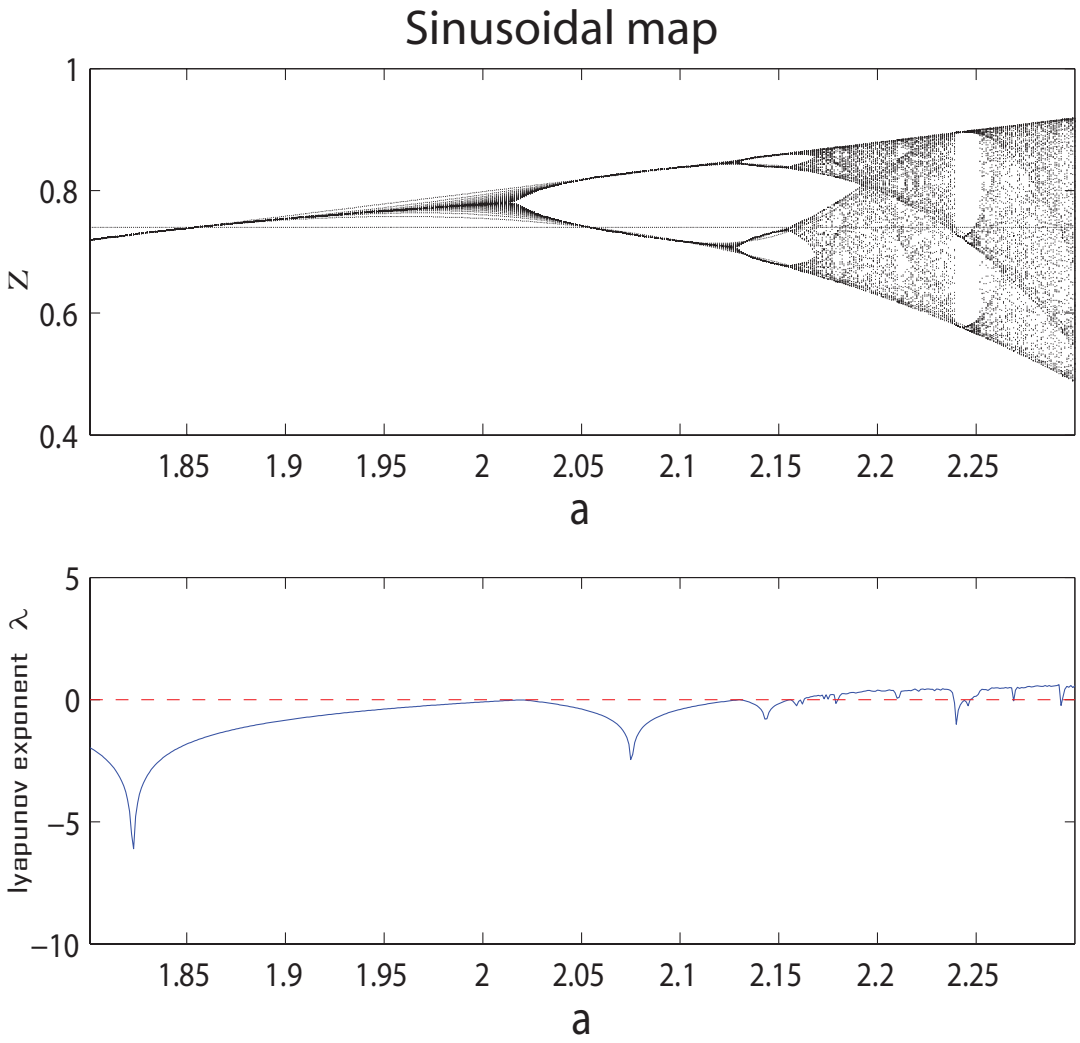


Figure 4.15: Lyapunov exponent of Sinusoidal map

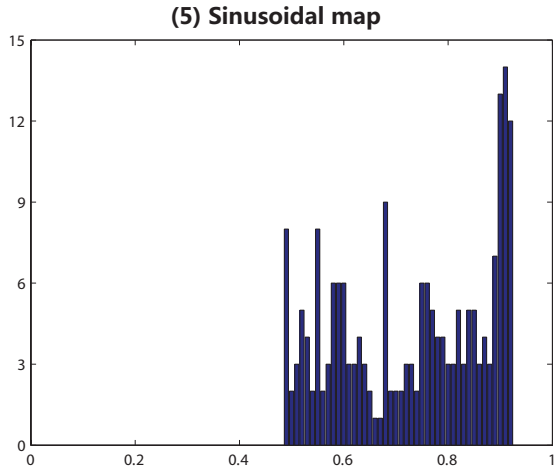


Figure 4.16: Chaotic PDF graph of Sinusoidal map

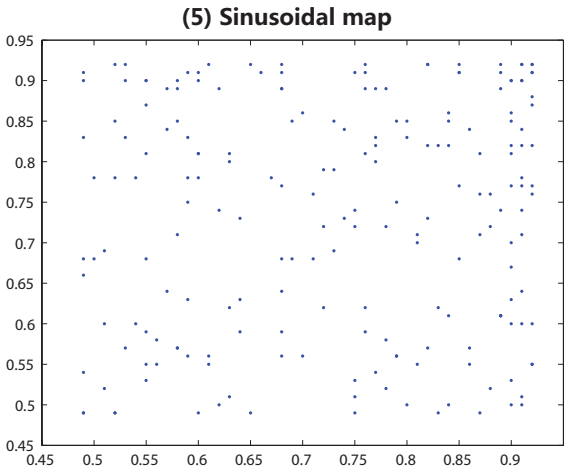


Figure 4.17: Chaotic 2-Dpoint distribution of Sinusoidal map

(6) Tent map: this map is similar to the well-known Logistic map, and displays some specific chaotic effects. These two maps can be converted to each other, and there is a relationship of topological conjugacy between them. Tent map can be defined by the following equation.

$$z_{k+1} = \begin{cases} z_k/\beta, & 0 < z_k \leq \beta \\ (1 - z_k)/(1 - \beta), & \beta < z_k \leq 1 \end{cases} \tag{4.16}$$

We set $\beta = 0.4$ and $z_0 = 0.152$.

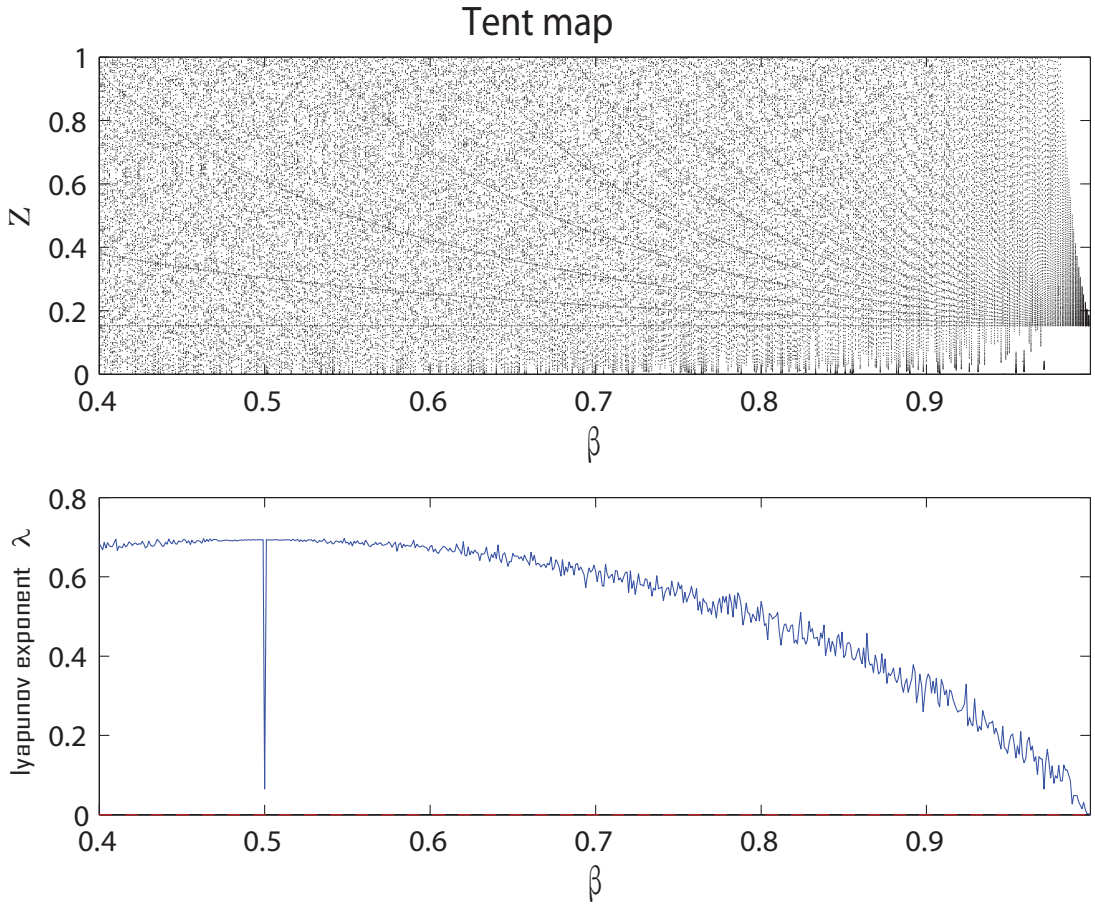


Figure 4.18: Lyapunov exponent of Tent map

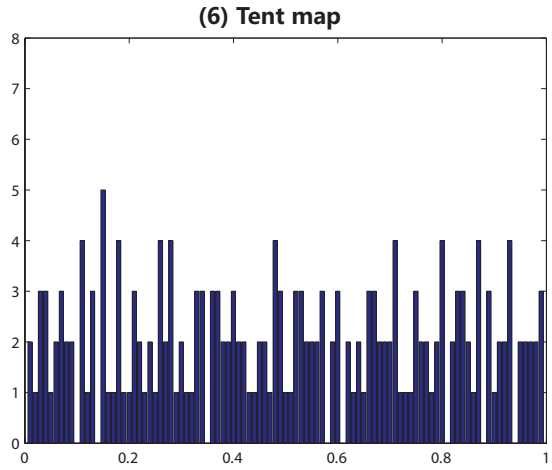


Figure 4.19: Chaotic PDF graph of Tent map

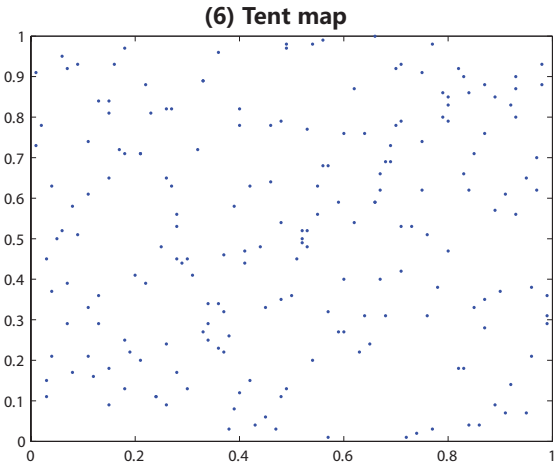


Figure 4.20: Chaotic 2-Dpoint distribution of Tent map

(7) Bernoulli shift map: this map belongs to the class of piecewise linear maps similar to the Tent map. It is formulated as follows

$$z_{k+1} = \begin{cases} z_k/(1-\lambda), & 0 < z_k \leq 1-\lambda \\ (z_k - 1 + \lambda)/\lambda, & 1-\lambda < z_k < 1 \end{cases} \tag{4.17}$$

We set $\lambda = 0.4$ and $z_0 = 0.152$.

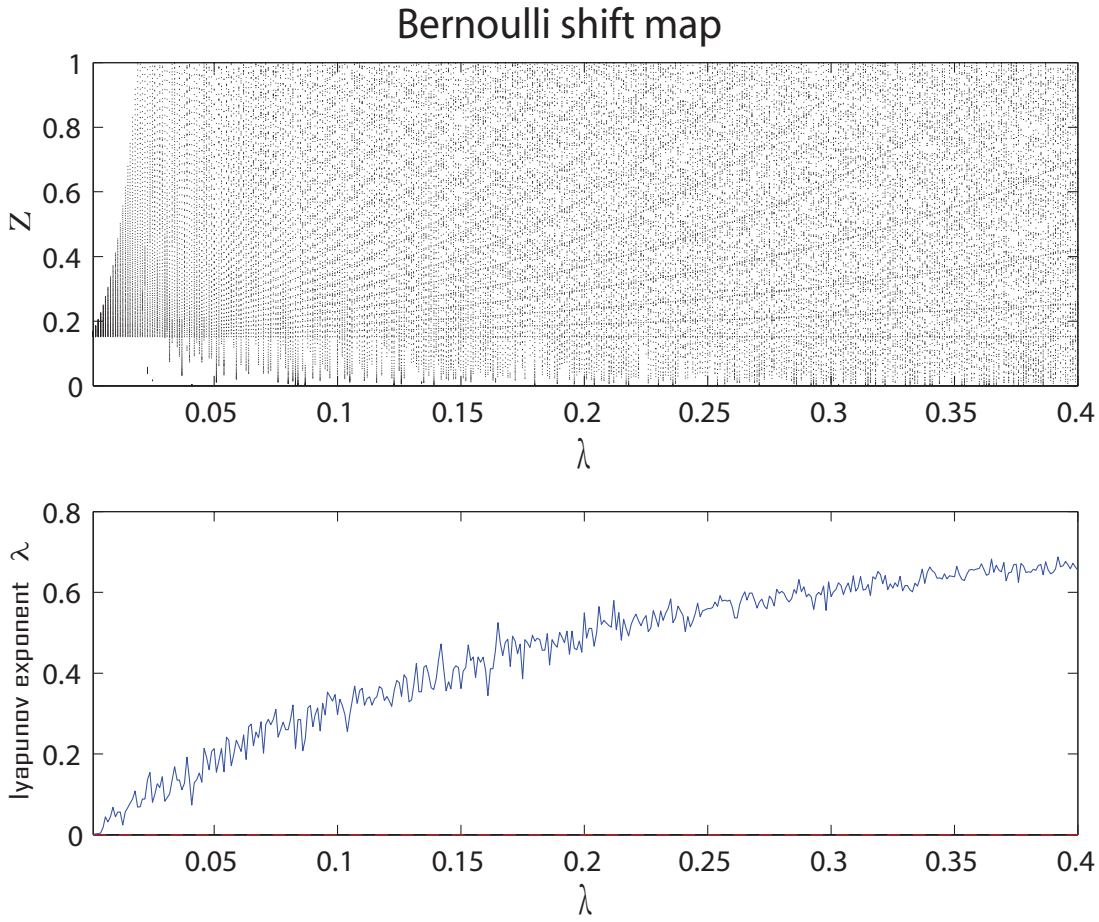


Figure 4.21: Lyapunov exponent of Bernoulli shift map

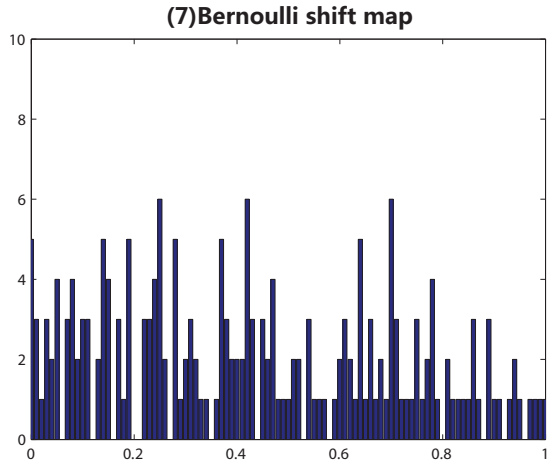


Figure 4.22: Chaotic PDF graph of Bernoulli shift map

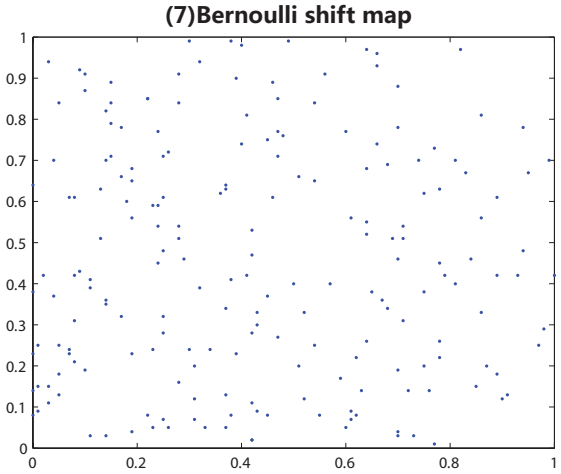


Figure 4.23: Chaotic 2-Dpoint distribution of Bernoulli shift map

(8) Chebyshev map: it is a common chaotic map, and has wide application in the neural network, digital communication and security. Its equation is expressed as

$$z_{k+1} = \cos(\phi \cos^{-1} z_k) \tag{4.18}$$

where the parameter ϕ is set to be 5 and the initial chaotic number $z_0 = 0.152$.

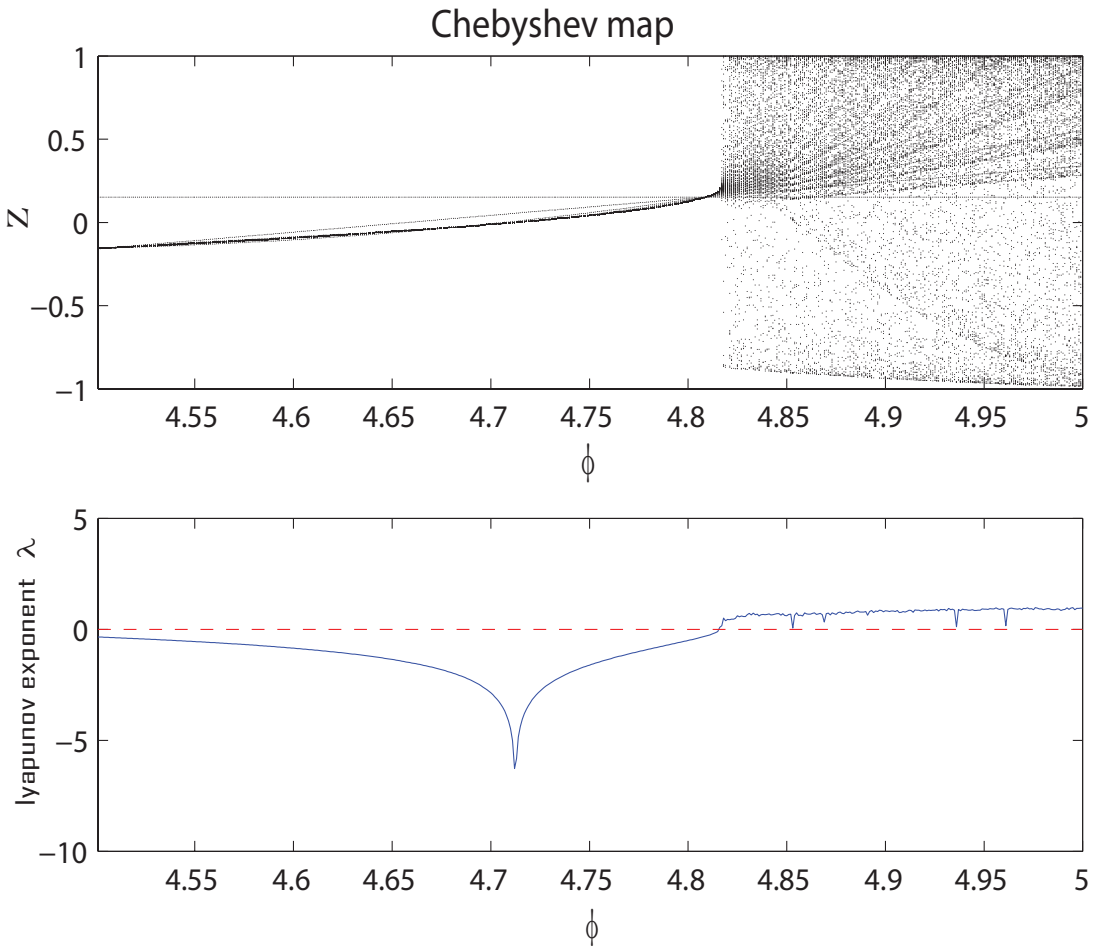


Figure 4.24: Lyapunov exponent of Chebyshev map

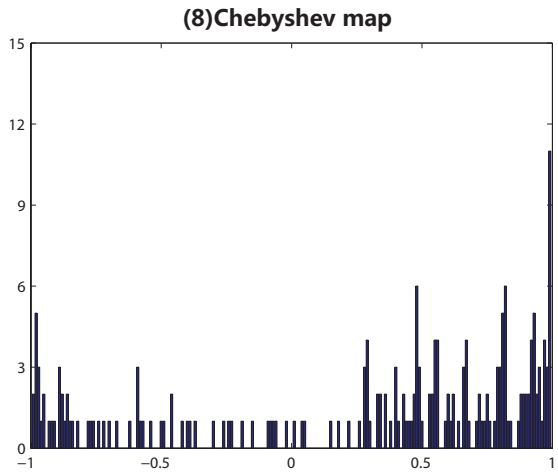


Figure 4.25: Chaotic PDF graph of Chebyshev map

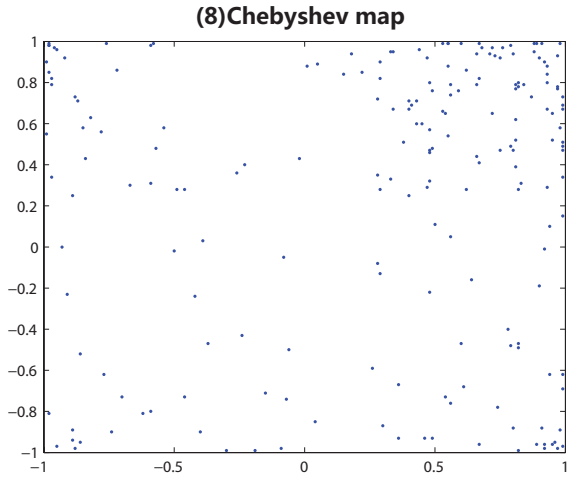


Figure 4.26: Chaotic 2-Dpoint distribution of Chebyshev map

(9) Circle map: this map is a simplified model for both driven mechanical rotors and the phase locked loop in electronics. It is a one-dimensional map which maps a circle onto itself. It is represented by the following equation.

$$z_{k+1} = z_k + a - \frac{b}{2\pi}\sin(2\pi z_k) \mod (1) \tag{4.19}$$

For $a = 0.5$ and $b = 2.2$, it can generate chaotic sequence in $(0, 1)$. Also, we set $z_0 = 0.152$ in the experiment.

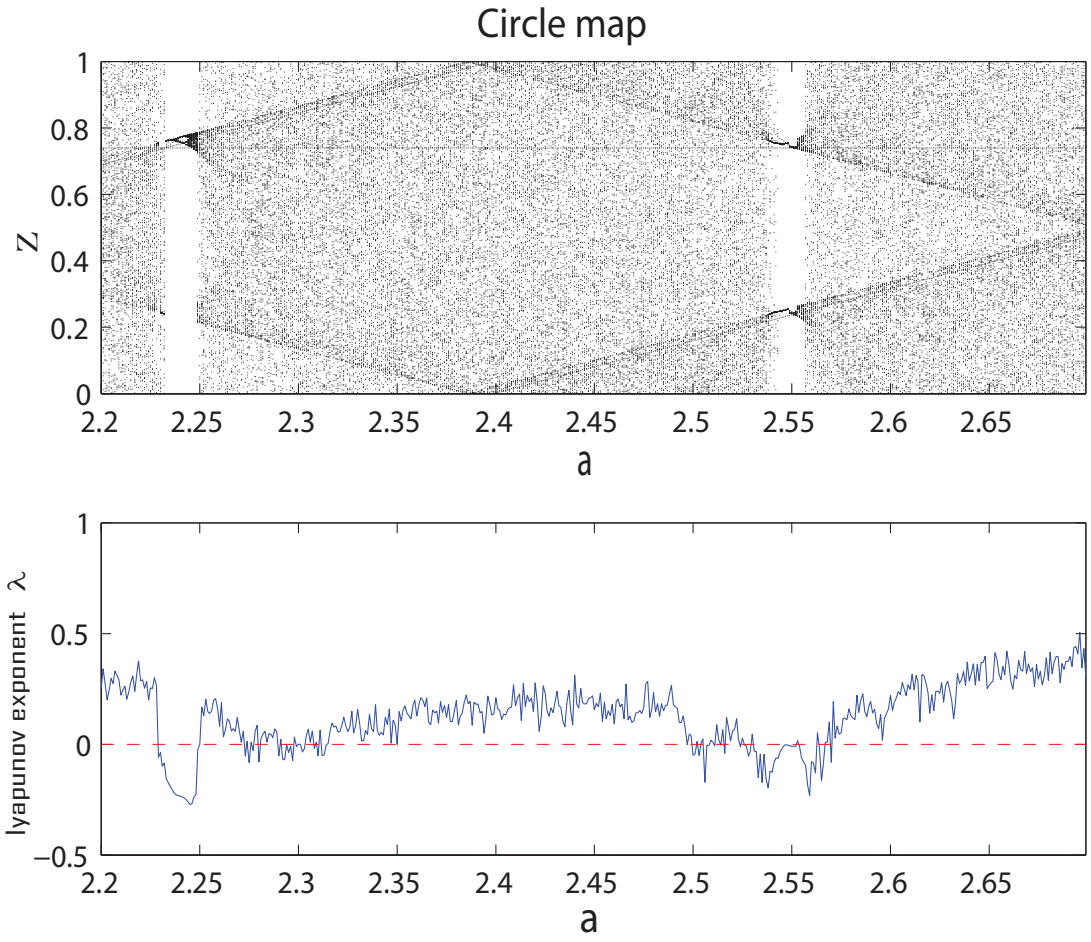


Figure 4.27: Lyapunov exponent of Circle map

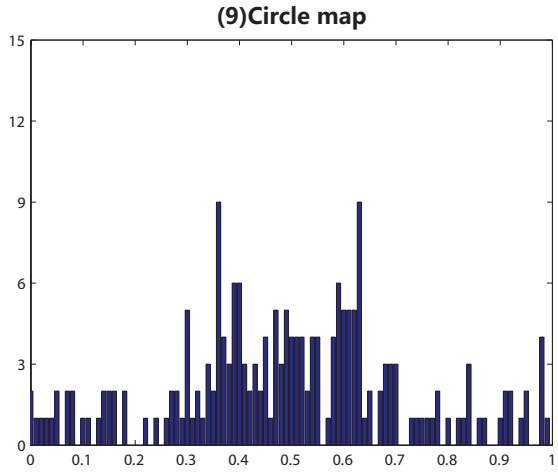


Figure 4.28: Chaotic PDF graph of Circle map

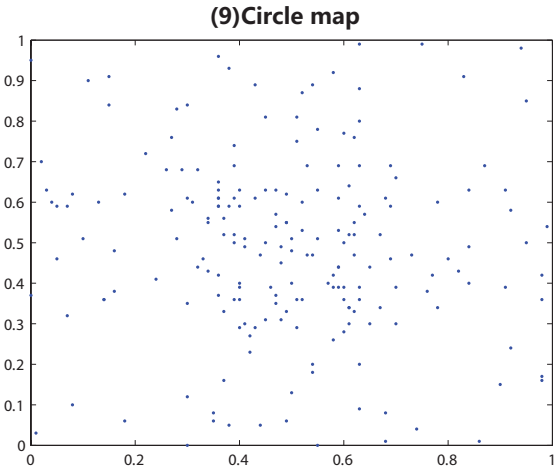


Figure 4.29: Chaotic 2-Dpoint distribution of Circle map

(10) Cubic map: it is one of the most commonly used maps in generating chaotic sequence in various applications like cryptography. It can be formally defined by

$$z_{k+1} = \rho z_k (1 - z_k^2) \tag{4.20}$$

We set $\rho = 2.59$ and $z_0 = 0.242$.

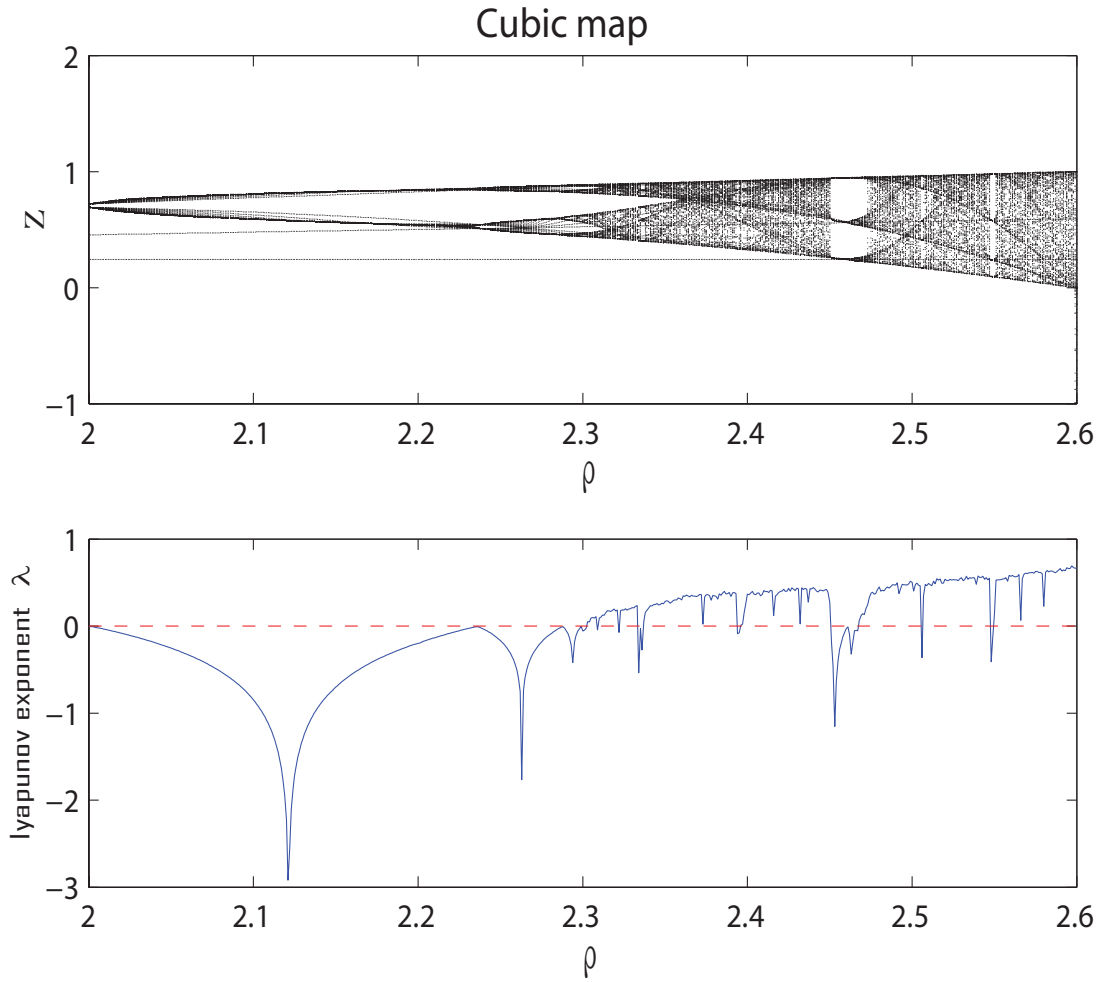


Figure 4.30: Lyapunov exponent of Cubic map

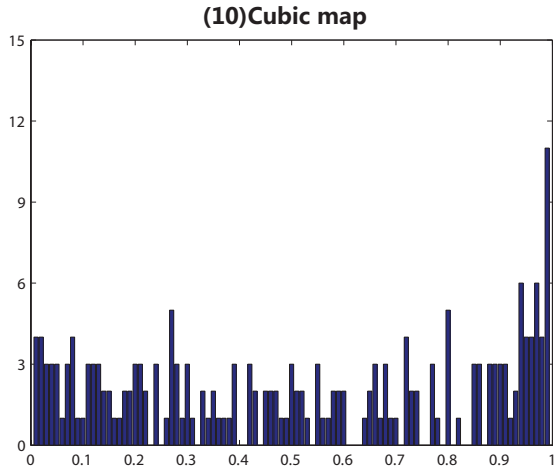


Figure 4.31: Chaotic PDF graph of Cubic map

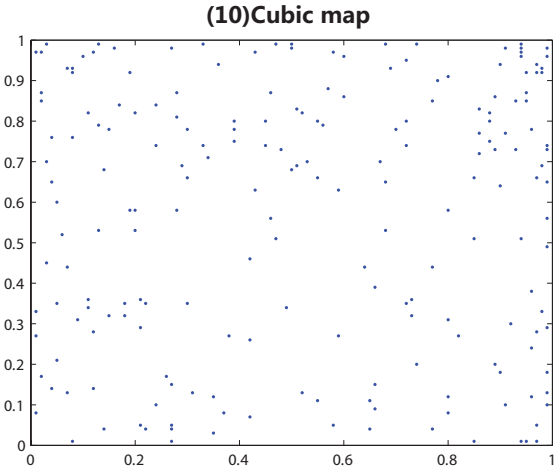


Figure 4.32: Chaotic 2-Dpoint distribution of Cubic map

(11) Gaussian map: it is represented using the following equation

$$z_{k+1} = \begin{cases} 0, & z_k = 0 \\ (\mu/z_k) \mod (1) & z_k \neq 0 \end{cases} \tag{4.21}$$

We set $\mu = 1$ and $z_0 = 0.152$.

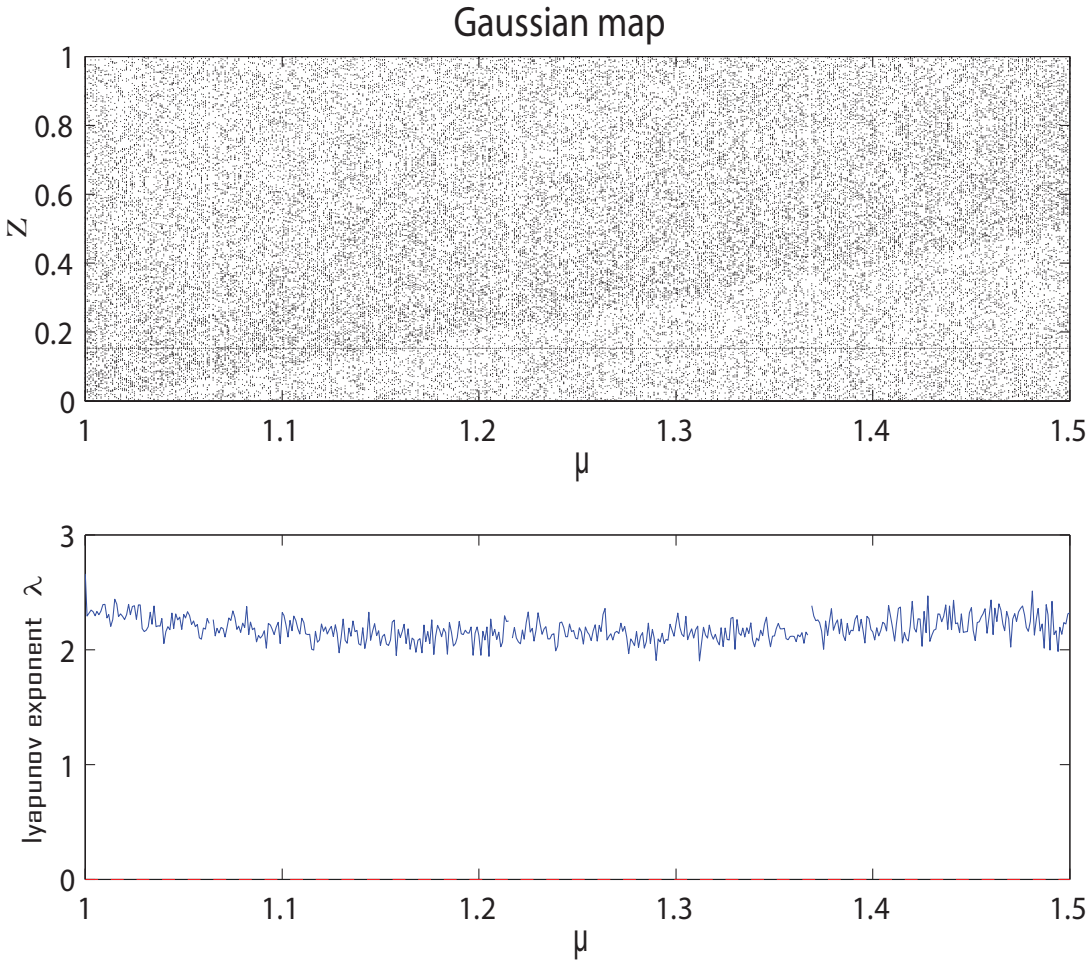


Figure 4.33: Lyapunov exponent of Gaussian map

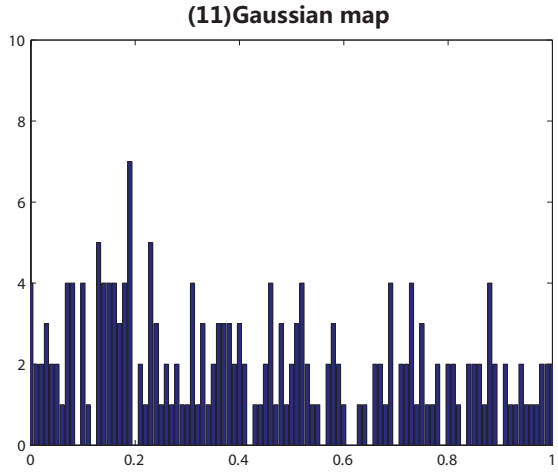


Figure 4.34: Chaotic PDF graph of Gaussian map

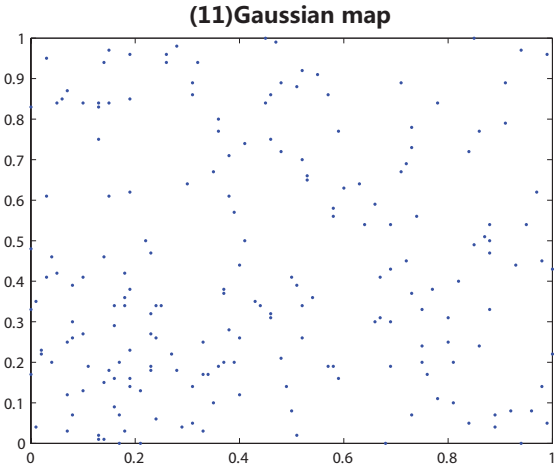


Figure 4.35: Chaotic 2-Dpoint distribution of Gaussian map

(12) Iterative chaotic map with infinite collapses (ICMIC): this map has infinite fixed points, and can be defined using

$$z_{k+1} = \sin(a/z_k) \tag{4.22}$$

where $a \in (0, +\infty)$ is an adjustable parameter, and we set $a = 70$ in our experiment. It is clear that ICMIC generates chaotic sequence $z \in [-1, 0) \cup (0, 1]$.

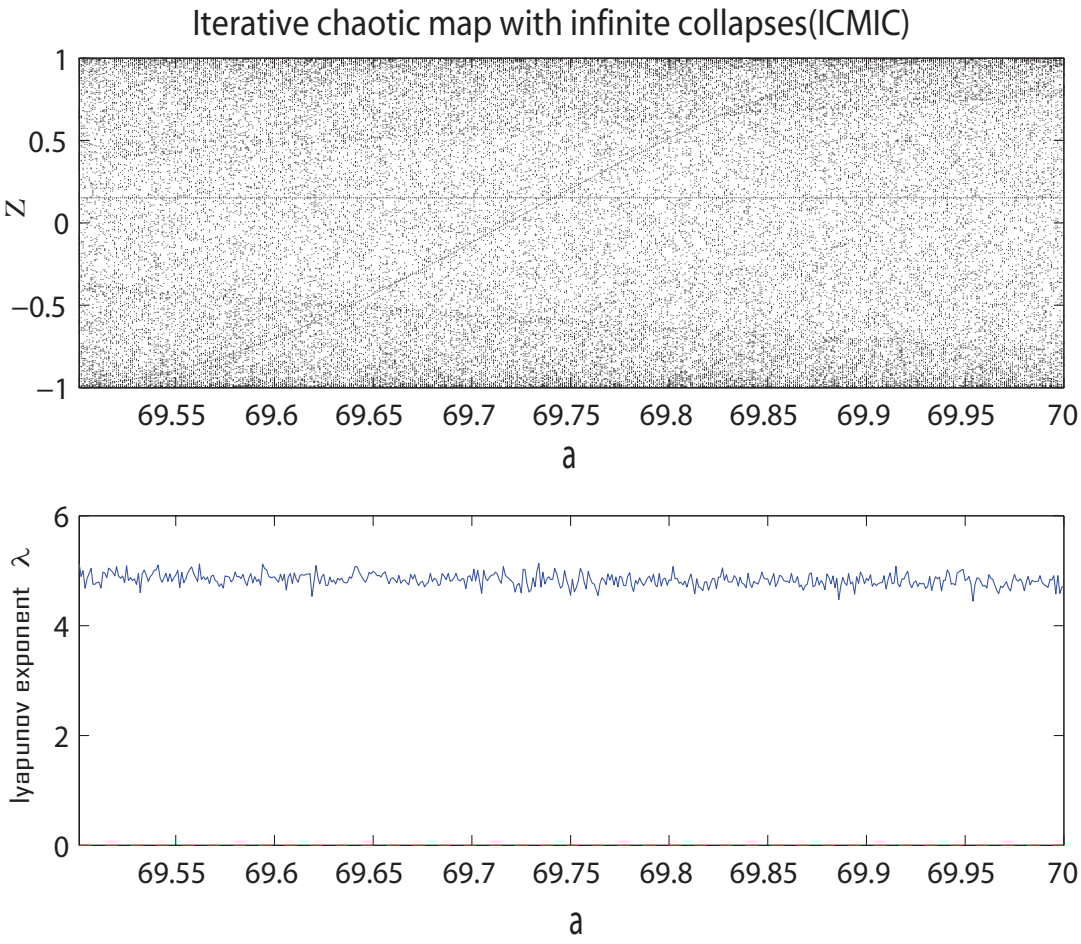


Figure 4.36: Lyapunov exponent of Iterative chaotic map with infinite collapses(ICMIC)

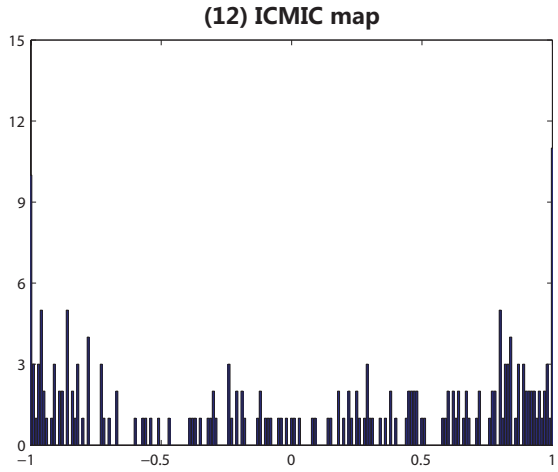


Figure 4.37: Chaotic PDF graph of Iterative chaotic map with infinite collapses

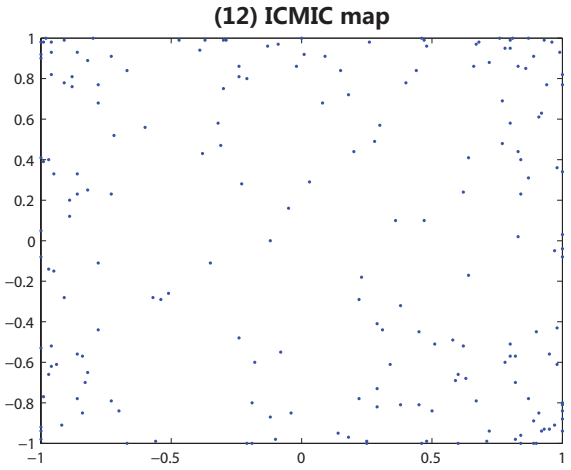


Figure 4.38: Chaotic 2-Dpoint distribution of Iterative chaotic map with infinite collapses

4.4 Chaotic Gravitational Search Algorithm (CGSA)

Chaos is a kind of a characteristic of nonlinear dynamic system which exhibits bounded dynamic unstable, pseudo random, ergodic, non-period behavior depended on initial value and control parameters [51]. Due to its ergodicity and randomicity, a chaotic system changes randomly, but eventually goes through every state if the time duration is long enough. This characteristic of chaotic systems can be utilized to build up a search operator for optimizing objective functions. Nevertheless, chaos optimization works well in a small search space but generates unacceptable optimization time in a large search space [52]. The main idea of chaotic local search in 2 dimensions is shown in Fig. 4.39. Therefore, chaotic search is often incorporated into other global optimizers such as evolutionary algorithms to enhance their search ability [37–48].

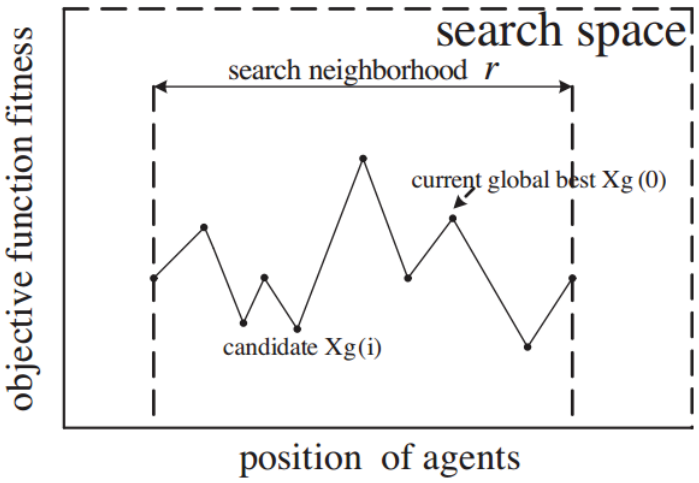


Figure 4.39: The main idea of chaotic local search in 2 dimensions

Compared with the methodology which uses chaotic sequences to substitute random values of the controlling parameters in GSA, chaotic local search has been demonstrated to be more effective for improving the performance of GSA [49]. As a matter of fact, chaotic local search is often adopted in related researches [38–48]. Thus, we employ the chaotic local search in this study.

The framework of CGSA is illustrated in Algorithm and each variant of CGSA differs from each other by specifying the chaotic local search procedure.

Algorithm:
01: **for all** agent i $(i = 1, 2, ..., N)$ **do**
02: initialize position X_i randomly in search space $[L, U]$
03: **end-for**
04: **while** termination criteria not satisfied **do**
05: **for all** agent i **do**
06: compute overall force $F_i^d(t)$ according to Eqs. (4.1)-(4.7)
07: compute acceleration $a_i^d(t)$ according to Eq. (4.8)
08: update velocity according to Eq. (4.9)
09: update position according to Eq. (4.10)
10: **end-for**
11: find out the global best agent X_g
12: implement the chaotic local search approach
13: decrease the chaotic search radius
14: **end-while**

4.4.1 Single chaos embedded CGSA

The chaotic local search that utilizes only a single chaotic map is defined as follows.

$$X_{g'}(t) = X_g(t) + r(t)(U - L)(z(t) - 0.5) \tag{4.23}$$

where $X_g(t)$ denotes the position of the current global best agent in the population at the t-th iteration number. $X_{g'}(t)$ is indicated as the new agent generated by the chaotic local search. U and L are the upper bound and lower bound of the search space, respectively. $z(t)$ is a chaotic variable generated from one of the considered chaotic maps. $r(t) \in (0, 1)$ is a chaotic search radius which is used to control the exploitation range of the search. It is worth pointing out that Eq. (4.23) actually denotes a batch local search manner. In a generation, the same chaotic variable $z(t)$ is used to update all components of the vector X_g (i.e., for all D dimensions). Without the loss of generality, we suppose the optimization problem is a minimization one. After the local search is performed, an agent updating procedure is carried out according to the following equations.

$$X_g(t+1) = \begin{cases} X_{g'}(t) & \text{If} \quad \text{fit}(X_{g'}(t)) \leq \text{fit}(X_g(t)) \\ X_g(t) & \text{Otherwise} \end{cases} \tag{4.24}$$

$$X_i(t+1) = X_i(t) \quad \text{For} \quad i = 1, 2, ..., N \quad \text{And} \quad i \neq g \tag{4.25}$$

The newly generated solution $X_{g'}$ will replace the current global best agent if the fitness is improved, while the others survive to enter into the next iteration.

Regarding the single chaos embedded CGSA, some remarks are given in the following.

1. The local search is performed on the global best agent, not only aiming to improve the search performance of GSA, but also being able to save computational time when compared to the scheme that applies the local search to all agents;

2. Once the acquired values of $X_{g'}$ in Eq. (4.23) locate out of the search bound,

these values will be reset to the closest boundary value;

3. Considering the fact that chaotic search is efficient in small range, a shrinking scheme is used to narrow the search neighborhood by lapse of iteration using $r(t+1) = 0.988 \times r(t)$.

The variants of single chaos embedded CGSA using the chaotic map in Eqs. (4.11) $\sim$ (4.22) are called CGSA-1 $\sim$ CGSA-12, respectively.

4.4.2 Multiple chaos embedded CGSA

Different chaotic maps exhibit different and distinct dynamic properties [53,54]. Multiple chaos are supposed to provide more opportunities for a meta-heuristic to help it jump out of the local minima via the ergodicity and randomicity of chaos. The method of incorporating multiple chaos into meta-heuristics remains challenging and fascinating. In the prior researches no sophisticated scheme has been proposed. Thus, we innovatively propose three novel multiple chaos embedding schemes in this paper. The twelve chaotic maps in Eqs. (4.11) $\sim$ (4.22) are (1) randomly, (2) parallelly, and (3) memory-selectively incorporated into GSA, respectively.

4.4.2.1 CGSA-R

The chaotic local search that randomly makes use of multiple chaos is defined in the following.

$$X_{g'}(t) = X_g(t) + r(t)(U - L)(z^j(t) - 0.5) \tag{4.26}$$

where $z^j(t)$ is a chaotic variable generated from the j-th chaotic map, and j is an uniformly distributed number generated from the set $\{1, 2, ..., 12\}$. In each iteration, only a single selected chaotic map is used. All twelve chaotic maps are used during the whole iterations and each one is implemented for approximately $t_{max}/12$ times. Thereafter, the updating procedure shown in Eqs. (4.24)(4.25) is performed.

4.4.2.2 CGSA-P

The chaotic local search that parallelly uses multiple chaos can be defined as follows.

$$X_{g'}^{j}(t) = X_g(t) + r(t)(U - L)(z^j(t) - 0.5) \tag{4.27}$$

where $X_{g'}^{j}$, $j = 1, 2, ..., 12$ presents a candidate solution temporarily generated by the chaotic local search and it indicates that twelve candidate solutions are simultaneously generated using twelve different chaotic maps. Thereafter, the best one among the twelve candidate solutions is taken to compare with the current global best solution $X_g(t)$. If the fitness can be improved, then replace the original one; otherwise remain the same. The updating rule can be formally expressed as:

$$X_g(t+1) = \begin{cases} X_{g'}^{j_{min}}(t) & \text{If} \quad \text{fit}(X_{g'}^{j_{min}}(t)) \leq \text{fit}(X_g(t)) \\ X_g(t) & \text{Otherwise} \end{cases} \tag{4.28}$$

$$j_{min} = j \in \{1, 2, ..., 12\} \quad \text{s.t.} \min_{j=1,2,...,12} \text{fit}(X_{g'}^{j}(t)) \tag{4.29}$$

4.4.2.3 CGSA-M

The basic idea of CGSA-M is derived from the adaptive trail vector generation strategy for differential evolution [55]. Similarly, we use this memory-based strategy for adaptively selecting different chaotic maps, and hereby named memory-selectively incorporation scheme. The implementation of the memory-selectively incorporation scheme can be described in the following.

In CGSA-M, with respect to each current global best agent X_g, one chaotic map is selected from twelve chaotic maps according to the probability learned from the success rate and failure rate in generating improved solutions within a certain number (i.e. LP) of previous iterations. The selected strategy is applied to the current global best agent X_g, to generate a new agent X'_g for comparing the fitness after utilizing the j-th chaotic map with X'_g to decide whether X_g would be replaced by X'_g, as shown in Eq. (4.23).

Initially, the probability of selecting each chaotic map is set to be 1/12, suggesting that all chaotic maps have the equal probability to be selected. With the lapse of iteration, the selection probabilities are updated according to the following rules.

$$p_{j,t} = \frac{S_{j,t}}{\sum_{j=1}^{12} S_{j,t}} \tag{4.30}$$

$$S_{j,t} = \frac{\sum_{g=t-LP}^{t-1} ns_{j,g}}{\sum_{g=t-LP}^{t-1} ns_{j,g} + \sum_{g=t-LP}^{t-1} nf_{j,g}} + \phi, \quad (j = 1, 2, ..., 12; t > LP) \tag{4.31}$$

where $p_{j,t}$ denotes the probability of selecting the j-th chaotic map at the t-th iteration. $ns_{j,t}$ indicates the number of new individuals generated by the j-th chaotic map and successfully entering the next iteration within the previous LP iterations with respect to generation t, and $nf_{j,t}$ denotes the number of these new individuals which failed to enter into the next iteration. $S_{j,t}$ represents the success rate, and $\phi = 0.01$ is set to avoid the null success rate. It is apparent that the larger the success rate for the j-th chaotic map, the higher the probability of applying it to generate new individual at the current iteration.

4.5 Experimental Results

To evaluate the performance of the proposed multiple chaos embedded gravitational search algorithms, i.e., CGSA-R, CGSA-P, CGSA-M, we make a comparison with the original gravitational search algorithm [28] and twelve single chaos embedded gravitational search algorithms using different chaotic maps (i.e., CGSA-1 $\sim$ CGSA-12). The experiment is conducted using Matlab on a personal PC.

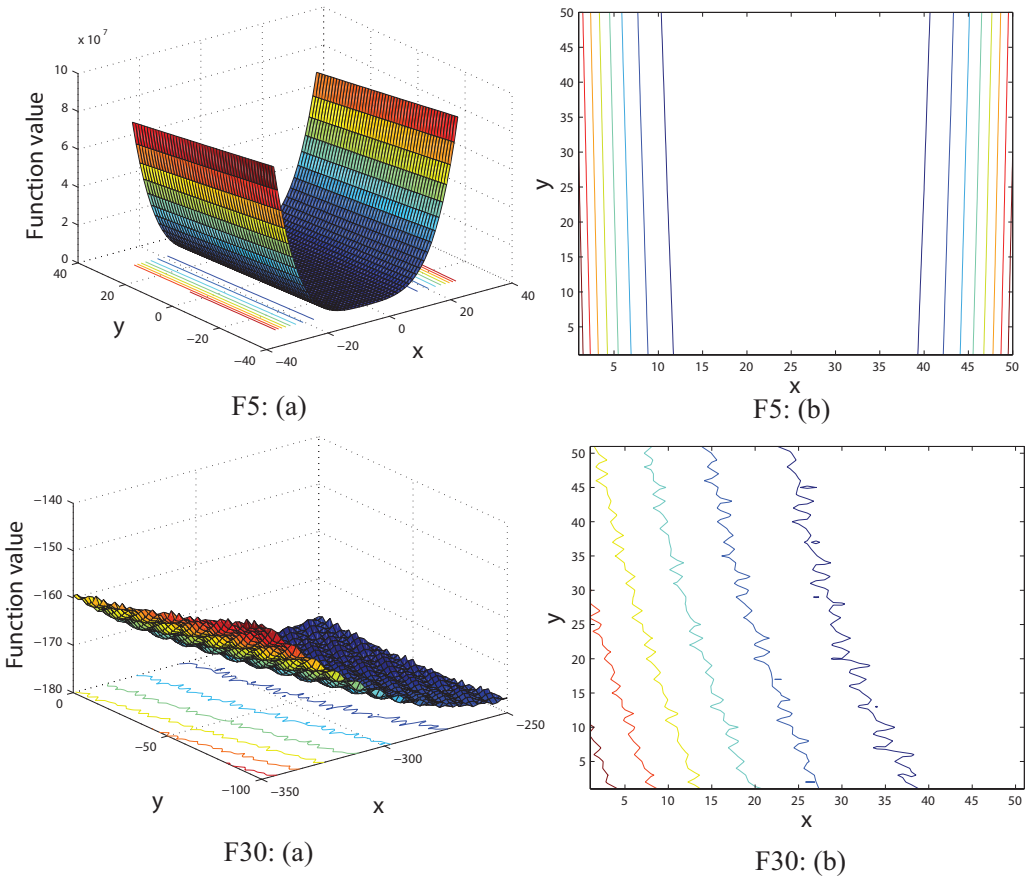


Figure 4.40: The 2-dimensional sketch (a) and the contour (b) for the unimodal function F5 and the shifted rotated Griewank's function F30, respectively.

In order to make a statistical analysis, all compared algorithms are implemented 30 times based on a total number of 48 benchmark functions. These benchmark functions are taken from [56, 57]. F1 $\sim$ F23 are the most commonly used benchmark numerical functions [56], where F1 and F5 are unimodal functions; F6 is a step function which has only one minimum and is discontinuous; F7 is a noisy quartic function; F8 $\sim$ F13 are multimodal functions with plenty of local minima and the number of the local minima in these functions increase exponentially with the dimension of the function; F14 $\sim$ F23 are low dimensional functions which only have a few local minima. These functions can successfully test the searching capacity of algorithms in terms of convergence speed and global exploration ability. In other words, unimodal functions are able to reflect the convergence speed of the algorithm in a direct man-

ner, and multimodal ones are likely to estimate the algorithms' ability of escaping from local minima. Nevertheless, these traditional 23 benchmark functions suffer from two problems: (1) global minima lie at the center of the search range (usually at $\overrightarrow{0}^D$), which might be easily utilized as a prior knowledge; (2) local minima lie along the coordinate axes or no linkage among the variables exists [55]. Shifted or rotated functions proposed in CEC'05 [57] can solve these problems in traditional 23 benchmarks functions. F24 $\sim$ F48 are CEC'05 functions, where F24 $\sim$ F28 are shifted unimodal functions; F29 $\sim$ F37 are shifted multimodal functions; F38 $\sim$ F48 are rotated hybrid composition functions. Fig. 4.40 illustrates the characteristics of the unimodal function F5 and the shifted rotated Griewank's function F30 respectively, in terms of the two-dimensional sketch and the contour. The user-defined parameters in GSA and CGSAs are set as follows. The population size N is 50. The maximum iteration number t_{max} is 1000. ε in Eq. (4.4) is set to be 1.0E-100 to make sure that it exerts little influence on the gravitational force. The shrinking constant α in Eq. (4.5) controls the decrease speed of $G(t)$ and is set to be $0.02t_{max}$. The initial value of the gravitational parameter $G_0 = 100$. The attraction scope parameter β in Eq. (4.7) is set to be 2%. As suggested in [55], we adopt $LP = 50$ for CGSA-M.

Tables 1 $\sim$ 8 summarize the experimental results of GSA, CGSA with 12 different chaos, CGSA-R, CGSA-P and CGSA-M for 48 tested benchmark functions. The recorded results are shown in the form of $Ave. \pm Dev.$, where $Ave.$ denotes the average of the optimization error (final best-so-far solution) of 30 independent runs for each algorithm, and $Dev.$ represents its standard deviation. The best result among the compared 16 algorithms is shown in bold. From Tables 1 $\sim$ 8, we can find that

1. The best results are always obtained by one of the variants of chaotic GSA rather than GSA, which suggests that the chaotic local search definitely improves the search performance of GSA.

2. The proposed MCGSA (including CGSA-R, CGSA-P and CGSA-M) can acquire the best solutions for 31 out of 48 benchmark functions.

3. On the other hand, all twelve variants of single chaos embedded GSA can per-

form the best for only 18 benchmark functions.

4. Thus, it can be stated that the multiple chaos incorporation scheme is generally better than the single one for improving the performance of GSA.

To give some insights into the search performance of compared algorithms, Figs. 4.41 and 4.42 depict the convergence graphs and distributions of the final solutions for functions F5 and F30 respectively. Two kinds of convergence graphs are utilized: one is the average best-so-far solutions versus the iteration number, and the other is the ratio of the best-so-far solution versus the iteration number. In Figs. 4.41(a) and 4.42(a), the horizontal axis in a linear scale indicates the generation (i.e., iteration number) of the algorithm, while the vertical axis in a logarithmic scale represents the average fitness of the best-so-far solutions generated by the algorithm. From these two sub-figures, it can be found that all algorithms converge quickly in early phases of the iteration, and trapped into local minima at the later phases. The best final solutions are obtained by CGSA-P for both F5 and F30.

On the other hand, the ratio of the best-so-far solutions found by chaotic GSAs to those found by GSA is depicted in Figs. 4.41(b) and 4.42(b), aiming to verify the effects of chaotic local search on the GSA. From Fig. 4.41(b), we observe that all chaotic GSAs can find better solutions than GSA in earlier phases, suggesting that the chaotic local search is able to improve GSA in an exploitation manner. However, some of chaotic GSAs generate worse solutions than GSA in the latter search phase and cannot improve solutions' qualities any further. It reveals that the chaotic local search cannot always improve the performance of GSA for all the time. Its effects strongly depend on the used chaotic map, which is usually the common case in chaotic meta-heuristics. In this figure, it is clear that the proposed MCGSA (including CGSA-R, CGSA-P and CGSA-M) can generate better solutions than the other compared algorithms. A same phenomenon that MCGSA performs better than the others can also be observed in Fig. 4.42(b).

Due to the stochastic feature of all compared 16 algorithms, a box-and-whisker diagram is used to depict the distribution of final obtained best-so-far solutions in Figs.

4.41(c) and 4.42(c). In these figures, five characteristic values including the smallest observation, lower quartile, median, upper quartile, and the largest observation are illustrated. Symbol + indicates outliers. From these sub-figures, it is quite clear that MCGSA outperforms its competitive algorithms. Especially, CGSA-P performs the best among all compared algorithms.

To further demonstrate the effectiveness and robustness of the proposed MCGSA, the average rankings of the algorithms obtained by the Friedman test [58, 59] on all tested 48 benchmark optimization functions are summarized in Tables 4.9 $\sim$ 4.11. The Friedman test is a nonparametric statistical test which applies the *post hoc* method of Iman-Davenport [59]. It can rank the algorithms for each problem separately. The best performing algorithm among all compared algorithms should have rank 1, the second best rank 2, and so on. From these results, it can be found that the best performing algorithm usually changes for a certain optimization function. It is difficult to find such an algorithm which can perform the best for all tested problems (also known as the No Fee Lunch Theorem [60]). Thus, we confirm that the performance of an algorithm not only depends on its searching capacity, but also relies on the fitness structure (shown as in Fig. 4.40) of the solved function.

Furthermore, the values in the last column of Table 4.11 record the average ranking of 48 functions for all compared 16 algorithms. CGSA-P gets the smallest value of 7.47, which means that it averagely performs the best for all functions. The second smallest value 7.58 is acquired by CGSA-R, while CGSA-M gets the third one. It is worth pointing out that GSA gets the largest ranking value, indicating that all chaotic GSAs performs better than GSA. In addition, it is mostly desired that a general well-performing algorithm should be designed. From this practical problem-solving perspective, we can conclude that the proposed multiple chaos incorporation scheme is effective for improving the performance of GSA.

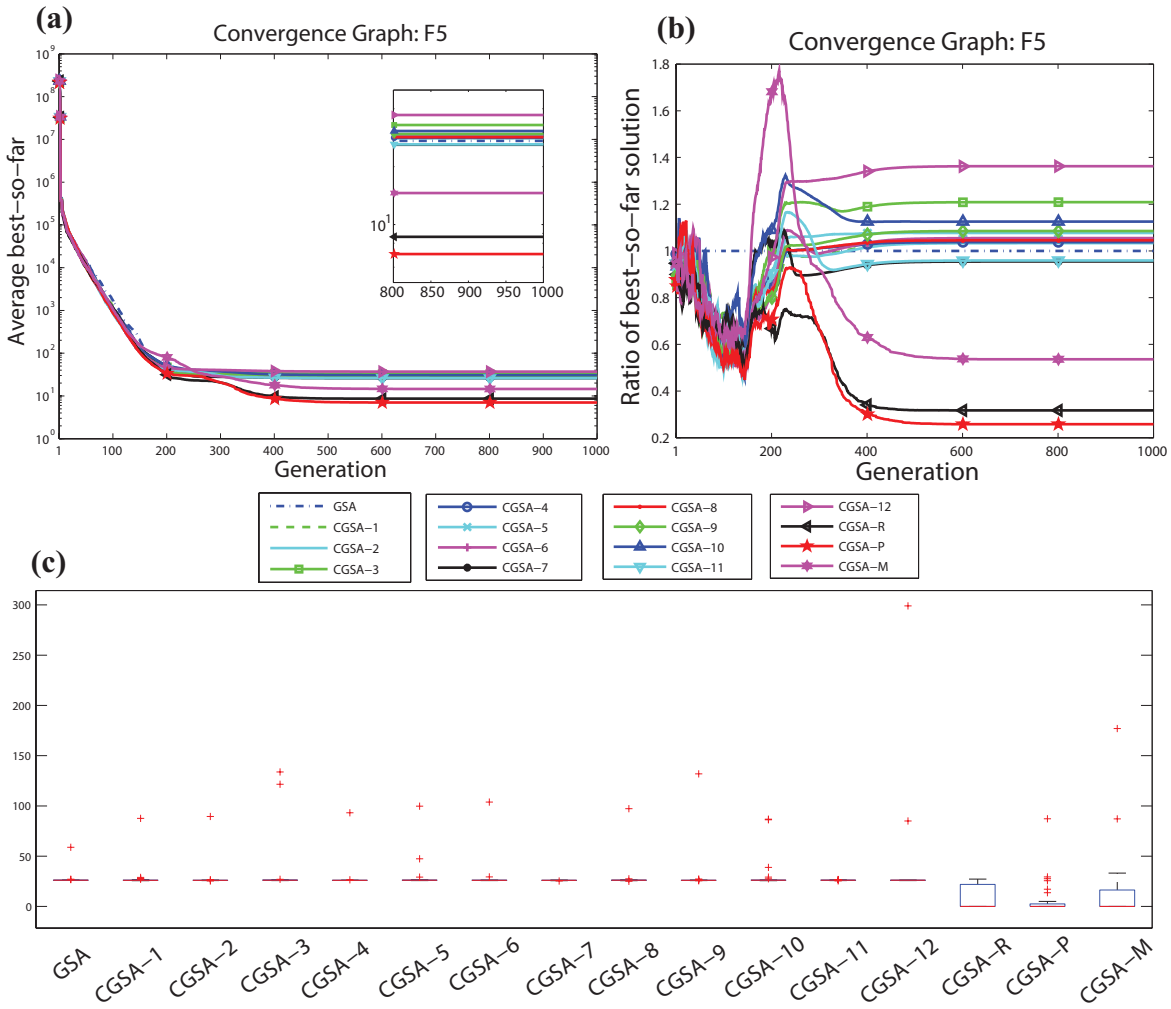


Figure 4.41: Search performance of the algorithms for comparison on F5

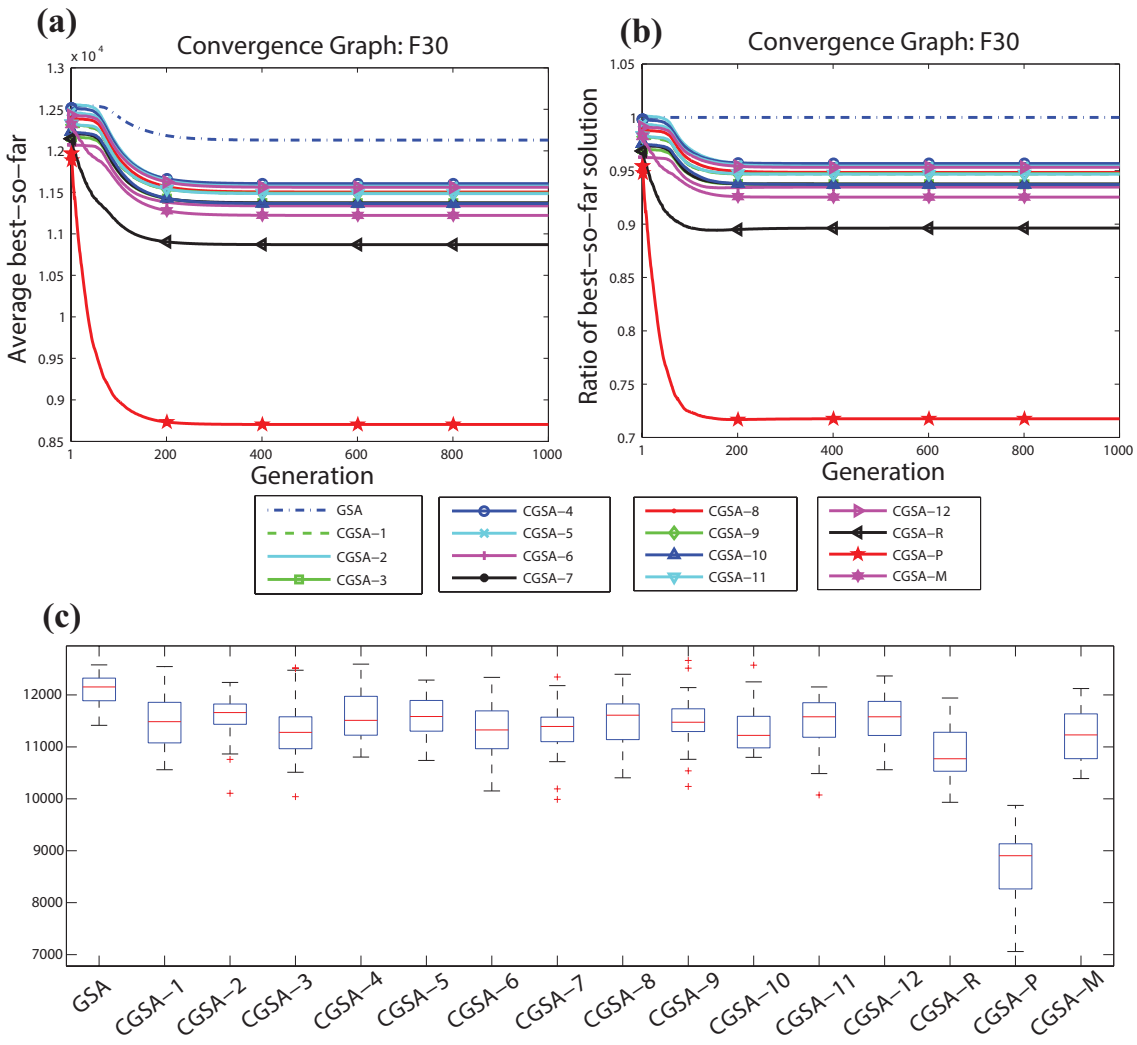


Figure 4.42: Search performance of the algorithms for comparison on F30

Table 4.1: Experimental results of benchmark functions (F1-F6) using traditional GSA, CGSA with 12 different chaos, CGSA-R, CGSA-P, and CGSA-M.

Algorithm	F1	F2	F3	F4	F5	F6
GSA	2.04E-17± 6.21E-18	2.31E-08± 3.49E-09	2.46E+02± 9.31E+01	3.23E-09± 7.46E-10	2.61E+01± 2.63E-01	0.00E+00± 0.00E+00
CGSA-1	1.16E-17± 3.11E-18	1.60E-08± 1.87E-09	2.25E+02± 9.03E+01	**2.73E-09± 6.76E-10**	2.82E+01± 1.12E+01	0.00E+00± 0.00E+00
CGSA-2	9.99E-18± 2.93E-18	1.59E-08± 2.35E-09	2.07E+02± 8.51E+01	3.07E-09± 6.73E-10	2.81E+01± 1.16E+01	0.00E+00± 0.00E+00
CGSA-3	1.08E-17± 2.86E-18	1.64E-08± 2.18E-09	2.24E+02± 7.96E+01	3.05E-09± 5.90E-10	3.29E+01± 2.58E+01	0.00E+00± 0.00E+00
CGSA-4	1.09E-17± 2.77E-18	1.67E-08± 2.73E-09	2.19E+02± 7.41E+01	1.04E-02± 5.72E-02	2.83E+01± 1.23E+01	0.00E+00± 0.00E+00
CGSA-5	1.21E-17± 3.11E-18	1.58E-08± 2.26E-09	1.97E+02± 6.53E+01	5.98E-04± 3.27E-03	2.93E+01± 1.39E+01	0.00E+00± 0.00E+00
CGSA-6	1.10E-17± 3.38E-18	1.65E-08± 2.25E-09	2.16E+02± 8.26E+01	3.63E-03± 1.99E-02	2.87E+01± 1.42E+01	0.00E+00± 0.00E+00
CGSA-7	1.07E-17± 2.78E-18	**1.54E-08± 1.82E-09**	2.16E+02± 8.32E+01	2.85E-09± 4.93E-10	2.60E+01± 2.00E-01	0.00E+00± 0.00E+00
CGSA-8	1.12E-17± 3.21E-18	1.63E-08± 3.16E-09	**1.92E+02± 7.68E+01**	4.57E-03± 2.50E-02	2.85E+01± 1.30E+01	0.00E+00± 0.00E+00
CGSA-9	1.12E-17± 2.88E-18	1.60E-08± 2.23E-09	2.19E+02± 9.91E+01	5.68E-03± 3.11E-02	2.95E+01± 1.93E+01	0.00E+00± 0.00E+00
CGSA-10	9.98E-18± 3.00E-18	1.62E-08± 2.03E-09	2.33E+02± 9.46E+01	4.10E-03± 2.24E-02	3.06E+01± 1.54E+01	0.00E+00± 0.00E+00
CGSA-11	1.26E-17± 5.33E-18	1.66E-08± 2.74E-09	2.48E+02± 1.05E+02	2.80E-09± 5.74E-10	2.61E+01± 2.86E-01	0.00E+00± 0.00E+00
CGSA-12	1.11E-17± 3.24E-18	1.59E-08± 2.78E-09	2.12E+02± 8.33E+01	3.11E-02± 1.70E-01	3.71E+01± 5.06E+01	0.00E+00± 0.00E+00
CGSA-R	1.17E-17± 4.12E-18	1.66E-08± 2.50E-09	2.11E+02± 8.92E+01	2.85E-09± 5.10E-10	8.64E+00± 1.18E+01	0.00E+00± 0.00E+00
CGSA-P	**9.10E-18± 2.87E-18**	1.66E-08± 2.50E-09	2.11E+02± 7.30E+01	2.91E-09± 7.14E-10	**7.02E+00± 1.75E+01**	0.00E+00± 0.00E+00
CGSA-M	1.19E-17± 3.82E-18	1.58E-08± 2.48E-09	2.32E+02± 7.41E+01	2.85E-09± 5.00E-10	1.46E+01± 3.55E+01	0.00E+00± 0.00E+00

Table 4.2: Experimental results of benchmark functions (F7-F12) using traditional GSA, CGSA with 12 different chaos, CGSA-R, CGSA-P, and CGSA-M.

Algorithm	F7	F8	F9	F10	F11	F12
GSA	1.97E-02± 1.23E-02	-2.66E+03± 4.17E+02	1.53E+01± 4.67E+00	3.60E-09± 4.55E-10	4.15E+00± 2.29E+00	3.76E-02± 1.32E-01
CGSA-1	1.16E-02± 4.62E-03	-2.89E+03± 4.95E+02	1.98E+01± 4.45E+00	2.68E-09± 4.81E-10	4.28E+00± 1.75E+00	4.33E-02± 9.29E-02
CGSA-2	1.11E-02± 5.43E-03	-2.81E+03± 4.97E+02	1.89E+01± 4.21E+00	2.63E-09± 4.26E-10	3.70E+00± 1.25E+00	3.44E-02± 9.33E-02
CGSA-3	1.12E-02± 6.63E-03	-2.98E+03± 4.87E+02	2.12E+01± 6.12E+00	2.69E-09± 3.44E-10	**3.61E+00± 1.38E+00**	1.07E-01± 3.79E-01
CGSA-4	1.34E-02± 4.72E-03	-2.96E+03± 5.55E+02	2.22E+01± 5.43E+00	2.74E-09± 3.00E-10	4.14E+00± 1.71E+00	3.91E-02± 7.32E-02
CGSA-5	1.18E-02± 6.83E-03	-2.82E+03± 5.33E+02	1.99E+01± 5.10E+00	2.65E-09± 3.60E-10	3.82E+00± 1.26E+00	3.45E-02± 6.76E-02
CGSA-6	1.26E-02± 4.55E-03	-2.90E+03± 4.23E+02	2.05E+01± 4.86E+00	2.67E-09± 3.44E-10	4.32E+00± 1.25E+00	1.62E-02± 4.61E-02
CGSA-7	1.21E-02± 6.41E-03	-3.01E+03± 3.85E+02	2.19E+01± 5.51E+00	2.67E-09± 4.13E-10	3.77E+00± 1.57E+00	6.28E-02± 2.14E-01
CGSA-8	1.10E-02± 5.58E-03	-2.68E+03± 3.60E+02	2.26E+01± 5.34E+00	2.65E-09± 3.76E-10	4.03E+00± 1.25E+00	4.30E-02± 9.36E-02
CGSA-9	**1.07E-02± 4.40E-03**	-2.74E+03± 4.43E+02	2.17E+01± 7.08E+00	2.75E-09± 4.76E-10	3.95E+00± 1.17E+00	4.15E-02± 9.66E-02
CGSA-10	1.08E-02± 4.58E-03	-2.94E+03± 4.91E+02	2.06E+01± 5.40E+00	2.65E-09± 3.47E-10	3.98E+00± 1.42E+00	3.99E-02± 9.56E-02
CGSA-11	1.20E-02± 5.24E-03	-2.89E+03± 4.98E+02	2.14E+01± 6.89E+00	2.60E-09± 3.36E-10	4.41E+00± 1.80E+00	5.09E-02± 1.15E-01
CGSA-12	1.10E-02± 4.52E-03	-2.72E+03± 4.58E+02	2.18E+01± 6.90E+00	2.88E-09± 4.76E-10	4.27E+10± 1.81E+00	1.73E-02± 3.93E-02
CGSA-R	1.13E-02± 4.65E-03	-3.06E+03± 6.34E+02	**1.46E+01± 3.91E+00**	2.68E-09± 3.14E-10	3.70E+00± 1.61E+00	4.09E-02± 8.16E-02
CGSA-P	1.18E-02± 3.80E-03	-3.07E+03± 5.83E+02	1.53E+01± 3.54E+00	2.63E-09± 3.90E-10	4.15E+00± 1.50E+00	**1.08E-02± 3.29E-02**
CGSA-M	1.10E-02± 4.31E-03	**-3.38E+03± 6.92E+02**	2.17E+01± 6.00E+00	**2.56E-09± 3.64E-10**	3.72E+00± 1.40E+00	3.85E-02± 1.20E-01

Table 4.3: Experimental results of benchmark functions (F13-F18) using traditional GSA, CGSA with 12 different chaos, CGSA-R, CGSA-P, and CGSA-M.

Algorithm	F13	F14	F15	F16	F17	F18
GSA	7.32E-04 ± 2.79E-03	3.61E+00 ± 2.86E+00	1.94E-03 ± 3.48E-04	-1.03E+00 ± 0.00E+00	3.98E-01 ± 1.13E-16	3.00E+00 ± 5.08E-15
CGSA-1	1.23E-03 ± 3.81E-03	1.20E+00 ± 5.46E-01	1.45E-03 ± 9.07E-04	-1.03E+00 ± 0.00E+00	3.98E-01 ± 1.13E-16	3.00E+00 ± 4.29E-15
CGSA-2	7.32E-04 ± 2.79E-03	1.31E+00 ± 6.55E-01	1.22E-03 ± 5.66E-04	-1.03E+00 ± 5.45E-16	3.98E-01 ± 0.00E+00	3.00E+00 ± 2.33E-15
CGSA-3	1.37E-03 ± 4.50E-03	1.84E+00 ± 1.38E+00	1.19E-03 ± 5.53E-04	-1.03E+00 ± 5.68E-16	3.98E-01 ± 0.00E+00	3.00E+00 ± 2.09E-15
CGSA-4	1.09E-03 ± 3.36E-03	1.43E+00 ± 7.65E-01	1.21E-03 ± 6.07E-04	-1.03E+00 ± 6.05E-16	3.98E-01 ± 0.00E+00	3.00E+00 ± 2.39E-15
CGSA-5	**1.12E-18 ± 2.98E-19**	1.30E+00 ± 6.71E+00	1.51E-03 ± 8.29E-04	-1.03E+00 ± 5.53E-16	3.98E-01 ± 0.00E+00	3.00E+00 ± 2.23E-15
CGSA-6	1.12E-18 ± 3.59E-19	1.43E+00 ± 8.90E-01	1.26E-03 ± 6.88E-04	-1.03E+00 ± 6.12E-16	3.98E-01 ± 0.00E+00	3.00E+00 ± 2.17E-15
CGSA-7	1.97E-03 ± 4.53E-03	1.37E+00 ± 6.17E-01	1.24E-03 ± 6.20E-04	-1.03E+00 ± 0.00E+00	3.98E-01 ± 1.13E-16	3.00E+00 ± 4.42E-15
CGSA-8	1.37E-03 ± 3.59E-03	1.61E+00 ± 1.17E+00	1.15E-03 ± 6.32E-04	-1.03E+00 ± 0.00E+00	3.98E-01 ± 1.13E-16	3.00E+00 ± 4.81E-15
CGSA-9	1.61E-03 ± 4.29E-03	1.51E+00 ± 9.80E-01	1.22E-03 ± 4.47E-04	-1.03E+00 ± 5.83E-16	3.98E-01 ± 2.93E-04	3.00E+00 ± 2.22E-15
CGSA-10	9.56E-04 ± 4.10E-03	1.80E+00 ± 9.53E-01	1.36E-03 ± 6.59E-04	-1.03E+00 ± 5.90E-16	3.98E-01 ± 4.66E-06	3.00E+00 ± 1.47E-15
CGSA-11	3.66E-04 ± 2.01E-03	1.45E+00 ± 7.64E-01	1.29E-03 ± 9.15E-04	-1.03E+00 ± 5.68E-16	3.98E-01 ± 0.00E+00	3.00E+00 ± 2.26E-15
CGSA-12	2.60E-04 ± 1.42E-03	1.39E+00 ± 6.67E-01	1.02E-03 ± 4.64E-04	-1.03E+00 ± 5.68E-16	3.98E-01 ± 9.43E-05	3.00E+00 ± 2.02E-15
CGSA-R	3.10E-04 ± 1.70E-03	1.59E+00 ± 1.23E+00	1.30E-03 ± 6.95E-04	**-1.03E+00 ± 0.00E+00**	**3.98E-01 ± 1.13E-16**	**3.00E+00 ± 4.93E-15**
CGSA-P	1.01E-03 ± 3.12E-03	**1.20E+00 ± 4.80E-01**	1.25E-03 ± 5.48E-04	**-1.03E+00 ± 0.00E+00**	**3.98E-01 ± 1.13E-16**	**3.00E+00 ± 5.50E-15**
CGSA-M	3.66E-04 ± 2.01E-03	1.37E+00 ± 1.14E+00	**1.02E-03 ± 3.63E-04**	**-1.03E+00 ± 0.00E+00**	**3.98E-01 ± 1.73E-05**	**3.00E+00 ± 4.97E-15**

Table 4.4: Experimental results of benchmark functions (F19-F24) using traditional GSA, CGSA with 12 different chaos, CGSA-R, CGSA-P, and CGSA-M.

Algorithm	F19	F20	F21	F22	F23	F24
GSA	-3.86E+00 ± 2.71E-15	-3.32E+00 ± 1.36E-15	-7.35E+00 ± 3.44E+00	-1.04E+01 ± 0.00E+00	-1.05E+01 ± 9.03E-15	1.28E+03 ± 5.03E+02
CGSA-1	-3.86E+00 ± 2.71E-15	-3.28E+00 ± 5.70E-02	-7.39E+00 ± 3.11E+00	-9.97E+00 ± 1.67E+00	-1.02E+01 ± 1.37E+00	1.27E+03 ± 6.94E+02
CGSA-2	-3.86E+00 ± 2.48E-15	-3.29E+00 ± 5.54E-02	-7.31E+00 ± 3.20E+00	-9.27E+00 ± 2.34E+00	-1.00E+01 ± 1.64E+00	1.57E+03 ± 8.15E+02
CGSA-3	-3.86E+00 ± 2.54E-15	-3.30E+00 ± 4.51E-02	-7.89E+00 ± 3.12E+00	-9.52E+00 ± 2.01E+00	-9.92E+00 ± 1.90E+00	1.14E+03 ± 7.77E+02
CGSA-4	-3.86E+00 ± 2.45E-15	-3.29E+00 ± 5.54E-02	-8.65E+00 ± 2.84E+00	-9.62E+00 ± 2.06E+00	-1.05E+01 ± 1.78E-15	1.49E+03 ± 9.77E+02
CGSA-5	-3.86E+00 ± 2.36E-15	-3.28E+00 ± 5.83E-02	-7.72E+00 ± 3.12E+00	-9.19E+00 ± 2.52E+00	-9.74E+00 ± 2.10E+00	1.15E+03 ± 5.86E+02
CGSA-6	-3.86E+00 ± 2.37E-15	-3.30E+00 ± 4.84E-02	-7.13E+00 ± 3.19E+00	-9.52E+00 ± 2.01E+00	-1.01E+00 ± 1.68E+00	1.25E+03 ± 7.38E+02
CGSA-7	-3.86E+00 ± 2.71E-15	-3.31E+00 ± 4.11E-02	-8.15E+00 ± 3.18E+00	-9.27E+00 ± 2.34E+00	-9.67E+00 ± 2.30E+00	1.44E+03 ± 7.28E+02
CGSA-8	-3.86E+00 ± 2.71E-15	-3.28E+00 ± 5.70E-02	-6.98E+00 ± 3.53E+00	-9.44E+00 ± 2.21E+00	-1.02E+00 ± 1.36E+00	1.26E+03 ± 8.00E+02
CGSA-9	-3.86E+00 ± 2.45E-15	-3.30E+00 ± 4.51E-02	-8.06E+00 ± 3.10E+00	-9.54E+00 ± 2.28E+00	-9.92E+00 ± 1.90E+00	1.47E+03 ± 7.29E+02
CGSA-10	-3.86E+00 ± 2.46E-15	-3.30E+00 ± 4.51E-02	**-8.89E+00 ± 2.37E+00**	-9.62E+00 ± 2.07E+00	-9.83E+00 ± 2.18E+00	1.45E+03 ± 9.86E+02
CGSA-11	-3.86E+00 ± 2.49E-15	-3.26E+00 ± 6.05E-02	-7.64E+00 ± 3.22E+00	-9.52E+00 ± 2.00E+00	-1.02E+00 ± 1.37E+00	1.49E+03 ± 7.68E+02
CGSA-12	-3.86E+00 ± 2.40E-15	-3.30E+00 ± 4.51E-02	-6.99E+00 ± 3.52E+00	-9.89E+00 ± 1.94E+00	-1.00E+00 ± 1.64E+00	1.33E+03 ± 7.20E+02
CGSA-R	**-3.86E+00 ± 2.71E-15**	**-3.32E+00 ± 1.36E-15**	-8.22E+00 ± 2.62E+00	**-1.04E+01 ± 0.00E+00**	**-1.05E+01 ± 9.03E-15**	**1.13E+03 ± 7.35E+02**
CGSA-P	**-3.86E+00 ± 2.71E-15**	**-3.32E+00 ± 1.36E-15**	-7.64E+00 ± 3.02E+00	-9.87E+00 ± 1.61E+00	**-1.05E+01 ± 9.03E-15**	1.20E+03 ± 7.27E+02
CGSA-M	**-3.86E+00 ± 2.71E-15**	-3.29E+00 ± 5.54E-02	-7.88E +00± 2.91E+00	-9.87E+00 ± 1.61E+00	**-1.05E+01 ± 9.03E-15**	1.15E+03 ± 4.78E+02

Table 4.5: Experimental results of benchmark functions (F25-F30) using traditional GSA, CGSA with 12 different chaos, CGSA-R, CGSA-P, and CGSA-M.

Algorithm	F25	F26	F27	F28	F29	F30
GSA	2.00E+04 ± 2.05E+03	5.09E+07 ± 7.46E+07	6.79E+04 ± 1.34E+04	2.28E+04 ± 2.12E+03	1.26E+08 ± 8.42E+07	1.21E+04 ± 2.97E+02
CGSA-1	1.97E+04 ± 1.98E+03	4.69E+07 ± 4.64E+07	5.36E+04 ± 1.07E+04	2.29E+04 ± 3.02E+03	1.10E+08 ± 6.11E+07	1.15E+04 ± 5.33E+02
CGSA-2	1.95E+04 ± 3.08E+03	4.36E+07 ± 4.97E+07	5.61E+04 ± 1.12E+04	2.32E+04 ± 3.61E+03	1.09E+08 ± 4.28E+07	1.16E+04 ± 4.40E+02
CGSA-3	**1.93E+04 ± 2.73E+03**	4.33E+07 ± 3.22E+07	5.04E+04 ± 8.65E+03	2.27E+04 ± 2.55E+03	1.03E+08 ± 4.56E+07	1.14E+04 ± 6.32E+02
CGSA-4	1.98E+04 ± 2.16E+03	3.81E+07 ± 3.92E+07	5.62E+04 ± 9.62E+03	**2.16E+04 ± 2.44E+03**	1.15E+08 ± 7.45E+07	1.16E+04 ± 4.98E+02
CGSA-5	1.97E+04 ± 2.07E+03	4.18E+07 ± 5.02E+07	5.14E+04 ± 9.68E+03	2.30E+04 ± 2.89E+03	1.13E+08 ± 5.72E+07	1.16E+04 ± 4.34E+02
CGSA-6	1.96E+04 ± 2.32E+03	3.77E+07 ± 2.93E+07	5.55E+04 ± 1.45E+04	2.30E+04 ± 2.61E+03	1.14E+08 ± 5.68E+07	1.13E+04 ± 5.01E+02
CGSA-7	2.03E+04 ± 2.26E+03	3.96E+07 ± 3.64E+07	5.50E+04 ± 1.26E+04	2.30E+04 ± 2.67E+03	1.23E+08 ± 6.93E+07	1.14E+04 ± 5.48E+02
CGSA-8	1.98E+04 ± 2.68E+03	4.94E+07 ± 4.80E+07	5.76E+04 ± 1.57E+04	2.19E+04 ± 2.76E+03	1.09E+08 ± 5.74E+07	1.15E+04 ± 5.05E+02
CGSA-9	2.01E+04 ± 2.02E+03	4.52E+07 ± 5.25E+07	5.35E+04 ± 9.18E+03	2.24E+04 ± 2.99E+03	1.23E+08 ± 6.99E+07	1.15E+04 ± 5.23E+02
CGSA-10	2.07E+04 ± 2.22E+03	3.50E+07 ± 3.50E+07	**4.98E+04 ± 9.69E+03**	2.24E+04 ± 2.75E+03	1.36E+08 ± 9.57E+07	1.14E+04 ± 4.86E+02
CGSA-11	1.98E+04 ± 2.35E+03	4.38E+07 ± 3.35E+07	5.77E+04 ± 1.34E+04	2.29E+04 ± 2.57E+03	1.33E+08 ± 7.86E+07	1.15E+04 ± 5.05E+02
CGSA-12	1.96E+04 ± 2.36E+03	4.32E+07 ± 4.17E+07	5.58E+04 ± 1.33E+04	2.22E+04 ± 2.71E+03	1.22E+08 ± 6.04E+07	1.16E+04 ± 4.12E+02
CGSA-R	1.95E+04 ± 2.22E+03	3.33E+07 ± 2.73E+07	5.93E+04 ± 1.68E+04	2.23E+04 ± 2.57E+03	1.15E+08 ± 7.58E+07	1.09E+04 ± 4.99E+02
CGSA-P	2.01E+04 ± 2.74E+03	**3.18E+07 ± 2.41E+07**	5.53E+04 ± 1.07E+04	2.28E+04 ± 3.18E+03	**1.00E+08 ± 5.17E+07**	**8.70E+03 ± 6.70E+02**
CGSA-M	1.96E+04 ± 2.69E+03	4.05E+07 ± 5.30E+07	5.44E+04 ± 1.15E+04	2.21E+04 ± 2.22E+03	1.04E+08 ± 7.30E+07	1.12E+04 ± 4.59E+02

Table 4.6: Experimental results of benchmark functions (F31-F36) using traditional GSA, CGSA with 12 different chaos, CGSA-R, CGSA-P, and CGSA-M.

Algorithm	F31	F32	F33	F34	F35	F36
GSA	2.03E+01 ± 9.60E-02	4.47E+01 ± 8.85E+00	3.58E+01 ± 7.50E+00	1.99E+00 ± 1.35E+00	5.28E+03 ± 6.81E+03	6.04E+00 ± 1.10E+00
CGSA-1	2.03E+01 ± 1.14E-01	5.48E+01 ± 1.14E+01	4.55E+01 ± 1.21E+01	4.16E+00 ± 2.55E+00	1.58E+03 ± 2.80E+03	5.16E+00 ± 1.10E+00
CGSA-2	2.02E+01 ± 9.34E-02	5.51E+01 ± 1.16E+01	5.22E+01 ± 1.20E+01	3.76E+00 ± 2.04E+00	2.53E+03 ± 3.97E+03	5.15E+00 ± 1.23E+00
CGSA-3	**2.02E+01 ± 7.44E-02**	5.21E+01 ± 9.32E+00	4.93E+01 ± 9.41E+00	4.57E+00 ± 3.28E+00	1.74E+03 ± 2.31E+03	5.06E+00 ± 1.19E+00
CGSA-4	2.03E+01 ± 7.26E-02	4.88E+01 ± 9.36E+00	4.88E+01 ± 1.30E+01	4.17E+00 ± 2.52E+00	2.10E+03 ± 2.98E+03	5.27E+00 ± 1.30E+00
CGSA-5	2.02E+01 ± 7.29E-02	5.35E+01 ± 1.30E+01	5.01E+01 ± 1.40E+01	3.98E+00 ± 2.09E+00	2.31E+03 ± 3.95E+03	5.36E+00 ± 1.12E+00
CGSA-6	2.03E+01 ± 9.75E-02	5.08E+01 ± 9.70E+00	5.34E+01 ± 1.16E+01	3.68E+00 ± 2.19E+00	1.67E+03 ± 2.16E+03	5.16E+00 ± 1.21E+00
CGSA-7	2.03E+01 ± 7.66E-02	5.07E+01 ± 1.12E+01	4.79E+01 ± 1.11E+01	3.56E+00 ± 2.34E+00	2.85E+03 ± 3.75E+03	5.28E+00 ± 1.27E+00
CGSA-8	2.03E+01 ± 1.04E-01	5.24E+01 ± 9.35E+00	4.79E+01 ± 9.65E+00	4.19E+00 ± 2.29E+00	3.25E+03 ± 9.69E+03	5.41E+00 ± 1.03E+00
CGSA-9	2.03E+01 ± 9.20E-02	5.59E+01 ± 1.14E+01	5.18E+01 ± 1.37E+01	3.96E+00 ± 2.48E+00	2.63E+03 ± 5.25E+03	5.29E+00 ± 1.02E+00
CGSA-10	2.03E+01 ± 8.85E-02	5.44E+01 ± 9.68E+00	4.78E+01 ± 1.20E+01	3.47E+00 ± 1.98E+00	2.21E+03 ± 2.35E+03	5.18E+00 ± 1.21E+00
CGSA-11	2.02E+01 ± 9.50E-02	5.21E+01 ± 1.12E+01	4.66E+01 ± 1.13E+01	4.04E+00 ± 2.07E+00	2.29E+03 ± 3.07E+03	5.16E+00 ± 1.32E+00
CGSA-12	2.03E+01 ± 1.02E-01	5.12E+01 ± 9.90E+00	5.17E+01 ± 1.34E+01	3.59E+00 ± 2.29E+00	**1.46E+03 ± 1.62E+03**	5.02E+00 ± 1.25E+00
CGSA-R	2.03E+01 ± 8.38E-02	4.19E+01 ± 7.18E+00	3.50E+01 ± 7.43E+00	**1.80E+00 ± 1.43E+00**	1.62E+03 ± 2.54E+03	**4.88E+00 ± 9.47E-01**
CGSA-P	2.03E+01 ± 8.82E-02	**4.05E+01 ± 7.41E+00**	**3.41E+01 ± 8.11E+00**	1.98E+00 ± 1.56E+00	2.92E+03 ± 4.00E+03	4.95E+00 ± 1.27E+00
CGSA-M	2.03E+01 ± 1.03E-01	5.79E+01 ± 1.34E+01	5.10E+01 ± 1.03E+01	4.04E+00 ± 2.85E+00	2.26E+03 ± 4.35E+03	5.28E+00 ± 1.21E+00

Table 4.7: Experimental results of benchmark functions (F37-F42) using traditional GSA, CGSA with 12 different chaos, CGSA-R, CGSA-P, and CGSA-M.

Algorithm	F37	F38	F39	F40	F41	F42
GSA	1.43E+01 ± 1.30E-01	3.23E+02 ± 4.85E+01	2.40E+02 ± 2.17E+02	3.51E+02 ± 2.60E+02	9.66E+02 ± 3.40E+01	9.55E+02 ± 5.03E+01
CGSA-1	1.38E+01 ± 2.57E-01	3.45E+02 ± 5.86E+01	2.07E+02 ± 1.69E+02	2.16E+02 ± 1.79E+02	9.70E+02 ± 4.99E+01	9.69E+02 ± 3.73E+01
CGSA-2	1.38E+01 ± 3.06E-01	3.50E+02 ± 6.74E+01	1.70E+02 ± 1.54E+02	2.82E+02 ± 2.11E+02	9.72E+02 ± 3.77E+01	9.64E+02 ± 5.77E+01
CGSA-3	1.38E+01 ± 2.28E-01	3.66E+02 ± 8.37E+01	2.55E+02 ± 1.93E+02	3.04E+02 ± 2.13E+02	9.69E+02 ± 4.84E+01	9.70E+02 ± 4.81E+01
CGSA-4	1.38E+01 ± 3.32E-01	3.55E+02 ± 8.21E+01	1.93E+02 ± 1.68E+02	**1.63E+02 ± 1.36E+02**	9.71E+02 ± 3.83E+01	9.59E+02 ± 6.24E+01
CGSA-5	1.39E+01 ± 2.79E-01	3.44E+02 ± 6.06E+01	2.21E+02 ± 1.77E+02	2.37E+02 ± 1.83E+02	9.73E+02 ± 3.56E+01	**9.44E+02 ± 6.90E+01**
CGSA-6	1.37E+01 ± 3.13E-01	3.72E+02 ± 7.22E+01	2.44E+02 ± 1.83E+02	2.71E+02 ± 2.02E+02	**9.41E+02 ± 7.80E+01**	9.66E+02 ± 4.73E+01
CGSA-7	1.38E+01 ± 2.59E-01	3.55E+02 ± 6.21E+01	1.26E+02 ± 9.50E+01	2.36E+02 ± 1.92E+02	9.74E+02 ± 5.02E+01	9.78E+02 ± 1.78E+01
CGSA-8	1.38E+01 ± 3.12E-01	3.31E+02 ± 8.11E+01	**1.14E+02 ± 9.44E+01**	2.72E+02 ± 2.17E+02	9.75E+02 ± 3.62E+01	9.64E+02 ± 4.85E+01
CGSA-9	1.38E+01 ± 2.71E-01	3.45E+02 ± 6.12E+01	1.30E+02 ± 1.19E+02	2.27E+02 ± 1.92E+02	9.70E+02 ± 4.86E+01	9.64E+02 ± 5.79E+01
CGSA-10	1.37E+01 ± 3.25E-01	3.27E+02 ± 4.61E+01	1.59E+02 ± 1.57E+02	2.51E+02 ± 1.91E+02	9.71E+02 ± 5.00E+01	9.53E+02 ± 6.40E+01
CGSA-11	1.38E+01 ± 3.06E-01	3.45E+02 ± 7.11E+01	1.33E+02 ± 1.26E+02	2.25E+02 ± 1.87E+02	9.55E+02 ± 6.39E+01	9.68E+02 ± 5.92E+01
CGSA-12	1.38E+01 ± 2.35E-01	3.44E+02 ± 7.20E+01	1.69E+02 ± 1.48E+02	3.06E+02 ± 2.09E+02	9.46E+02 ± 7.45E+01	9.73E+02 ± 4.77E+01
CGSA-R	1.38E+01 ± 3.02E-01	3.43E+02 ± 6.88E+01	2.29E+02 ± 1.67E+02	2.49E+02 ± 2.00E+02	9.76E+02 ± 2.42E+01	9.68E+02 ± 4.76E+01
CGSA-P	1.38E+01 ± 2.48E-01	**3.02E+02 ± 9.91E+00**	1.77E+02 ± 1.38E+02	2.82E+02 ± 1.97E+02	9.51E+02 ± 5.77E+01	9.50E+02 ± 6.09E+01
CGSA-M	**1.37E+01 ± 2.81E-01**	3.56E+02 ± 7.79E+01	2.20E+02 ± 1.90E+02	2.55E+02 ± 1.95E+02	9.83E+02 ± 2.00E+01	9.76E+02 ± 3.78E+01

Table 4.8: Experimental results of benchmark functions (F43-F48) using traditional GSA, CGSA with 12 different chaos, CGSA-R, CGSA-P, and CGSA-M.

Algorithm	F43	F44	F45	F46	F47	F48
GSA	9.66E+02 ± 3.78E+01	6.73E+02 ± 2.92E+02	9.31E+02 ± 1.71E+01	7.58E+02 ± 2.82E+02	2.55E+02 ± 2.11E+02	1.68E+03 ± 1.29E+01
CGSA-1	9.63E+02 ± 5.66E+01	7.83E+02 ± 3.29E+02	9.62E+02 ± 2.61E+01	8.10E+02 ± 3.10E+02	4.94E+02 ± 4.57E+02	1.73E+03 ± 2.59E+01
CGSA-2	9.68E+02 ± 4.90E+01	6.96E+02 ± 3.04E+02	9.63E+02 ± 2.28E+01	8.27E+02 ± 3.01E+02	7.36E+02 ± 4.79E+02	1.72E+03 ± 2.35E+01
CGSA-3	**9.55E+02 ± 7.27E+01**	7.61E+02 ± 3.25E+02	9.65E+02 ± 2.49E+01	8.84E+02 ± 2.91E+02	4.97E+02 ± 4.32E+02	1.72E+03 ± 2.19E+01
CGSA-4	9.64E+02 ± 5.75E+01	7.61E+02 ± 3.26E+02	9.61E+02 ± 2.91E+01	8.79E+02 ± 3.02E+02	3.93E+02 ± 3.94E+02	1.72E+03 ± 1.82E+01
CGSA-5	9.74E+02 ± 3.67E+01	7.62E+02 ± 3.26E+02	9.56E+02 ± 1.88E+01	8.18E+02 ± 2.92E+02	5.39E+02 ± 4.70E+02	1.73E+03 ± 2.85E+01
CGSA-6	9.73E+02 ± 3.76E+01	7.82E+02 ± 3.23E+02	9.57E+02 ± 2.79E+01	7.63E+02 ± 2.94E+02	5.14E+02 ± 4.53E+02	1.73E+03 ± 1.79E+01
CGSA-7	9.71E+02 ± 3.63E+01	8.28E+02 ± 3.34E+02	9.72E+02 ± 2.77E+01	8.17E+02 ± 3.19E+02	4.18E+02 ± 4.05E+02	1.73E+03 ± 2.81E+01
CGSA-8	9.65E+02 ± 4.69E+01	7.84E+02 ± 3.30E+02	9.59E+02 ± 2.63E+01	7.83E+02 ± 3.03E+02	5.51E+02 ± 4.60E+02	1.73E+03 ± 3.08E+01
CGSA-9	9.68E+02 ± 4.73E+01	7.40E+02 ± 3.20E+02	9.52E+02 ± 2.52E+01	8.21E+02 ± 3.02E+02	3.88E+02 ± 3.83E+02	1.73E+03 ± 2.64E+01
CGSA-10	9.57E+02 ± 5.46E+01	9.13E+02 ± 3.20E+02	9.59E+02 ± 2.14E+01	8.58E+02 ± 2.99E+02	4.52E+02 ± 4.26E+02	1.73E+03 ± 2.23E+01
CGSA-11	9.80E+02 ± 3.54E+01	6.52E+02 ± 2.80E+02	9.66E+02 ± 2.31E+01	7.56E+02 ± 2.89E+02	5.76E+02 ± 4.71E+02	1.73E+03 ± 2.54E+01
CGSA-12	9.71E+02 ± 4.89E+01	8.07E+02 ± 3.29E+02	9.70E+02 ± 3.09E+01	8.55E+02 ± 3.10E+02	5.79E+02 ± 4.74E+02	1.73E+03 ± 2.08E+01
CGSA-R	9.63E+02 ± 4.72E+01	7.83E+02 ± 3.30E+02	9.76E+02 ± 2.29E+01	**7.29E+02 ± 2.74E+02**	4.52E+02 ± 4.30E+02	**1.66E+03 ± 1.42E+01**
CGSA-P	9.83E+02 ± 1.87E+01	**6.52E+02 ± 2.80E+02**	**9.29E+02 ± 1.76E+01**	8.03E+02 ± 3.04E+02	**2.37E+02 ± 1.76E+02**	1.75E+03 ± 2.76E+01
CGSA-M	9.80E+02 ± 3.83E+01	8.26E+02 ± 3.32E+02	9.79E+02 ± 3.76E+01	7.74E+02 ± 2.95E+02	5.52E+02 ± 4.72E+02	1.73E+03 ± 2.86E+01

4.6 Discussions

As discussed in detail in Section 5, our proposed multiple chaotic embedded CGSA (MCGSA) can perform better than traditional GSA and 12 single chaos embedded CGSAs under the condition of the same maximum number of iterations. That is to say, under this condition MCGSA outperforms its competitive algorithms in terms of solution accuracy and the capacity of jumping out of local optima. However, as the computational cost in each iteration for the compared algorithms is different, it is necessary to compare these algorithms under the same computational cost. To realize this, we set the termination criteria to be the maximum number of function evaluations (i.e., D*10000) for all algorithms. We select all functions to be the test suit.

On the other hand, to further verify the effect of the usage of chaos, we perform a contrast analysis to answer the following question: why the chaos is effective for perturbation force of the local search. We conduct two variants of CGSA by replacing the $z(t)$ in Eq. (4.23) to be random numbers with a uniform distribution or normal distribution. We designate the newly conducted algorithms to be GSA-UD (i.e., the GSA using a uniform distribution random sequence embedded local search) and GSA-ND (i.e., the GSA using a normal distribution random sequence embedded local search), respectively.

Table 4.12 summarizes the results for GSA, CGSA-R, CGSA-P, CGSA-M, GSA-UD, and GSA-ND for the all benchmark functions over 30 independent runs. Figs. 4.43 and 4.44 depict the convergence graphs (average best-so-far versus number of function evaluation) for two typical functions: F5 and F30. It is clear that MCGSA generally outperforms the compared algorithms in terms of solution accuracy under almost the same computational burden. For F13, CGSA-P and CGSA-R perform significantly better than the others. For F30, although almost the same average value (4.86E+03) are obtained by CGSA-P and GSA-UD, CGSA-P possesses a smaller deviation, which suggests that CGSA-P is more robust to generate promising solutions. However, there are still two (F28,F43) of the 48 functions cannot got the solution

Table 4.9: Rankings of 16 variants of GSAs based on Friedman test for benchmark optimization function F1 - F16.

Algorithm	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
GSA	14.8	15.4	10.4	10.7	10.6	8.5	12.2	11.0	5.0	15.4	8.0	8.6	12.8	13.6	13.8	8.5
CGSA-1	8.8	8.2	8.7	6.5	9.2	8.5	8.5	8.6	8.6	8.0	9.0	9.1	8.3	6.9	9.1	8.5
CGSA-2	6.8	8.0	8.1	10.0	9.1	8.5	7.6	9.2	8.0	7.7	7.6	8.2	8.5	7.1	8.4	8.5
CGSA-3	7.7	8.7	9.2	9.6	10.9	8.5	7.2	7.3	9.2	8.2	7.1	8.5	8.8	10.4	7.8	8.5
CGSA-4	8.1	8.8	9.0	8.3	9.0	8.5	9.8	8.0	10.6	9.0	8.8	8.9	8.6	8.3	8.2	8.5
CGSA-5	9.8	7.4	7.4	9.7	9.3	8.5	8.2	9.6	9.0	7.8	8.2	8.7	7.7	8.1	10.2	8.5
CGSA-6	7.7	8.5	8.2	8.4	9.4	8.5	9.2	8.5	9.3	8.0	9.9	7.9	7.9	7.8	8.0	8.5
CGSA-7	7.7	7.1	8.4	8.2	8.4	8.5	8.6	7.0	10.2	8.0	7.7	8.6	8.7	8.8	8.4	8.5
CGSA-8	8.4	7.8	6.5	9.0	10.3	8.5	7.7	11.1	10.5	7.7	9.1	8.6	9.3	8.7	7.3	8.5
CGSA-9	8.4	7.9	8.2	8.9	8.5	8.5	7.5	9.9	9.8	8.5	8.7	8.3	8.4	8.7	8.4	8.5
CGSA-10	6.8	7.6	9.2	7.7	10.0	8.5	7.9	8.1	9.0	8.1	8.5	8.8	8.0	9.3	8.8	8.5
CGSA-11	9.3	8.8	9.7	7.6	10.0	8.5	8.5	8.6	9.5	7.0	9.5	9.3	7.7	8.9	7.7	8.5
CGSA-12	8.1	8.0	8.0	7.3	9.9	8.5	8.3	10.2	9.0	9.9	9.0	7.8	7.5	7.9	6.5	8.5
CGSA-R	8.8	8.6	7.6	8.1	4.0	8.5	8.2	7.1	3.7	8.4	7.8	9.0	8.0	7.6	8.4	8.5
CGSA-P	5.7	8.4	8.1	8.2	3.1	8.5	8.9	6.7	4.6	7.5	9.2	7.4	8.3	6.7	8.4	8.5
CGSA-M	9.1	7.0	9.2	7.9	3.8	8.5	7.7	5.0	10.1	6.7	8.0	8.3	7.5	7.4	6.7	8.5

Table 4.10: Rankings of 16 variants of GSAs based on Friedman test for benchmark optimization function F17 - F32.

Algorithm	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31	F32
GSA	8.4	8.5	8.5	6.8	8.8	7.5	8.0	8.8	8.9	8.2	12.7	9.3	9.1	14.5	9.8	5.8
CGSA-1	8.4	8.5	8.5	9.5	8.8	8.1	8.5	8.1	7.9	8.7	7.9	9.5	8.2	9.2	8.4	9.7
CGSA-2	8.4	8.5	8.5	9.2	9.0	9.1	8.8	10.4	7.2	9.0	9.0	9.2	8.8	10.3	7.4	10.2
CGSA-3	8.4	8.5	8.5	8.1	8.4	8.9	8.8	7.1	7.6	9.3	6.5	8.7	7.8	8.0	7.2	9.1
CGSA-4	8.4	8.5	8.5	9.2	7.5	8.6	8.0	9.5	8.2	8.4	9.0	6.9	8.3	9.6	8.1	7.9
CGSA-5	8.4	8.5	8.5	9.7	8.6	9.2	9.0	7.5	8.4	7.5	7.0	9.0	8.8	9.9	7.6	9.4
CGSA-6	8.4	8.5	8.5	8.4	9.4	8.8	8.5	7.8	7.9	9.2	8.0	9.3	8.9	7.9	8.0	9.0
CGSA-7	8.4	8.5	8.5	7.9	8.2	9.1	9.1	9.3	9.3	8.8	8.0	9.2	8.8	8.0	9.8	8.3
CGSA-8	8.4	8.5	8.5	9.5	9.3	8.8	8.5	8.3	8.6	8.8	9.1	7.6	8.3	9.5	9.3	9.4
CGSA-9	8.7	8.5	8.5	8.1	8.3	8.6	8.8	9.7	9.3	8.1	8.1	8.1	9.4	9.0	9.8	10.4
CGSA-10	8.7	8.5	8.5	8.1	7.0	8.6	8.8	8.7	10.4	7.8	6.4	8.5	8.7	8.1	8.3	10.2
CGSA-11	8.4	8.5	8.5	10.8	8.7	8.8	8.5	9.9	8.6	9.2	9.6	8.6	9.4	9.7	7.9	8.5
CGSA-12	8.7	8.5	8.5	8.1	9.3	8.1	8.8	8.6	8.2	9.1	8.1	7.7	9.1	9.9	7.7	8.4
CGSA-R	8.4	8.5	8.5	6.9	7.8	7.5	8.0	7.0	8.3	7.8	9.7	7.9	7.9	4.9	8.7	4.4
CGSA-P	8.4	8.5	8.5	6.9	8.6	8.3	8.0	7.4	8.9	7.7	8.6	8.9	7.5	1.0	10.6	4.2
CGSA-M	9.2	8.5	8.5	9.2	8.5	8.3	8.0	8.1	8.1	8.3	8.1	7.6	7.0	6.6	7.5	11.1

Table 4.11: Rankings of 16 variants of GSAs based on Friedman test for benchmark optimization function F33 - F48.

Algorithm	F33	F34	F35	F36	F37	F38	F39	F40	F41	F42	F43	F44	F45	F46	F47	F48	Total Average
GSA	3.9	5.4	10.7	11.6	15.0	9.2	7.3	8.9	6.8	6.7	7.2	7.1	3.6	8.1	6.5	2.2	**9.23**
CGSA-1	7.5	9.3	6.8	8.4	9.1	8.3	9.6	8.0	8.9	8.4	8.5	8.5	9.3	8.4	8.9	10.3	**8.55**
CGSA-2	10.5	9.0	9.3	7.9	8.3	8.8	8.3	9.0	8.6	9.0	8.2	7.7	9.4	8.3	10.8	9.0	**8.68**
CGSA-3	9.5	10.2	8.1	7.2	8.3	9.6	10.6	10.1	9.1	8.9	9.3	8.6	9.6	9.8	8.6	7.7	**8.62**
CGSA-4	9.1	9.9	7.6	8.6	8.1	8.7	9.0	6.7	8.2	8.3	8.8	8.5	8.6	9.9	7.6	7.7	**8.55**
CGSA-5	9.7	9.7	8.9	9.0	9.4	9.4	9.1	8.0	8.6	7.2	8.2	8.5	8.1	8.5	9.2	8.5	**8.65**
CGSA-6	11.2	8.7	8.2	8.1	6.6	10.3	9.9	8.2	7.8	8.6	8.8	8.9	7.8	8.1	8.7	10.2	**8.59**
CGSA-7	9.1	8.8	9.3	8.4	7.7	8.7	7.7	7.6	10.0	8.9	8.2	9.4	10.3	8.4	8.2	9.6	**8.59**
CGSA-8	9.5	10.0	8.3	9.8	7.7	7.1	7.1	8.8	9.0	8.2	6.9	8.8	8.5	8.2	9.0	9.6	**8.66**
CGSA-9	10.0	9.0	8.4	9.0	8.3	8.8	6.9	7.7	9.1	9.2	8.6	8.2	7.3	8.8	7.6	9.7	**8.65**
CGSA-10	9.0	8.5	9.6	8.1	7.5	6.6	7.4	8.5	8.9	7.7	7.0	10.3	8.2	8.8	8.1	9.5	**8.41**
CGSA-11	8.4	9.8	9.0	8.6	8.9	8.5	7.3	7.9	8.2	10.3	10.4	7.1	9.7	7.9	9.3	9.9	**8.83**
CGSA-12	10.5	8.4	7.8	7.8	8.2	8.0	8.2	9.7	8.0	9.5	9.4	9.1	9.7	8.7	9.2	9.5	**8.60**
CGSA-R	4.0	4.5	7.4	7.1	8.0	8.5	9.6	7.8	8.7	9.3	7.0	8.9	11.4	7.4	8.5	1.1	**7.58**
CGSA-P	3.6	5.5	9.1	7.4	8.2	5.5	9.0	10.1	6.4	6.3	9.5	7.0	3.3	8.6	6.5	12.4	**7.47**
CGSA-M	10.6	9.3	7.5	8.9	6.9	10.0	9.0	8.9	10.0	9.7	10.1	9.4	11.0	8.1	9.3	9.1	**8.37**

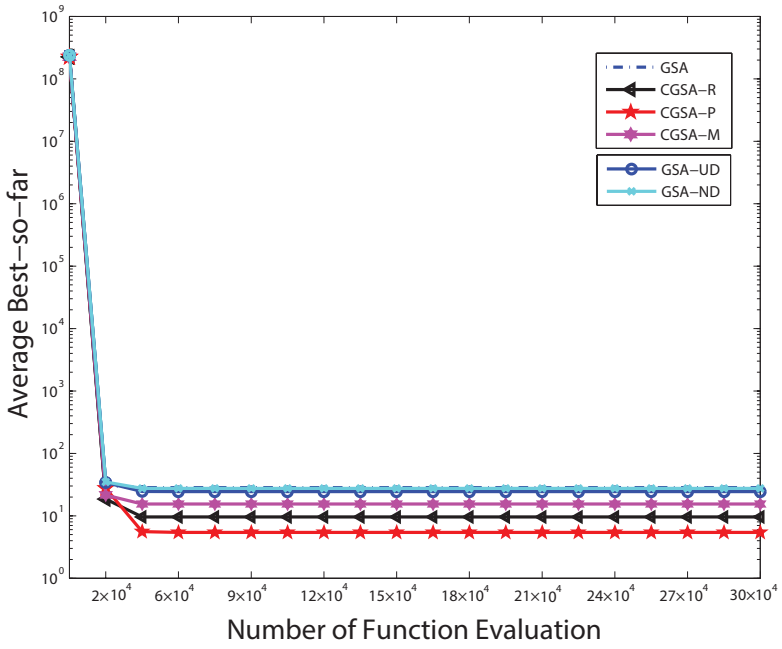


Figure 4.43: Convergence graph of F5: solution along with the number of function evaluation.

better than that by random or normal distribution. In addition, it should be pointed out that the solutions in Table 4.12 are generally better than those in Tables $1 \sim 8$ because more iteration numbers are implemented in this complementary experiment (e.g. 6000 iterations are carried out for GSA). All in all, we can conclude that: (1) the local search induced by chaos is more efficient than that by random or normal distribution numbers; (2) multiple chaos embedded local search generally performs better than single chaos embedded one; and (3) the parallelly embedding strategy is the most effective for improving the performance of GSA.

4.7 Conclusions

In this paper, taking into account the abundant searching dynamics of different chaos we innovatively propose a multiple chaos embedded gravitational search algorithm (MCGSA). To further improve the searching performance of GSA, three kinds of incorporation schemes are investigated. Multiple chaotic maps are randomly, paral-

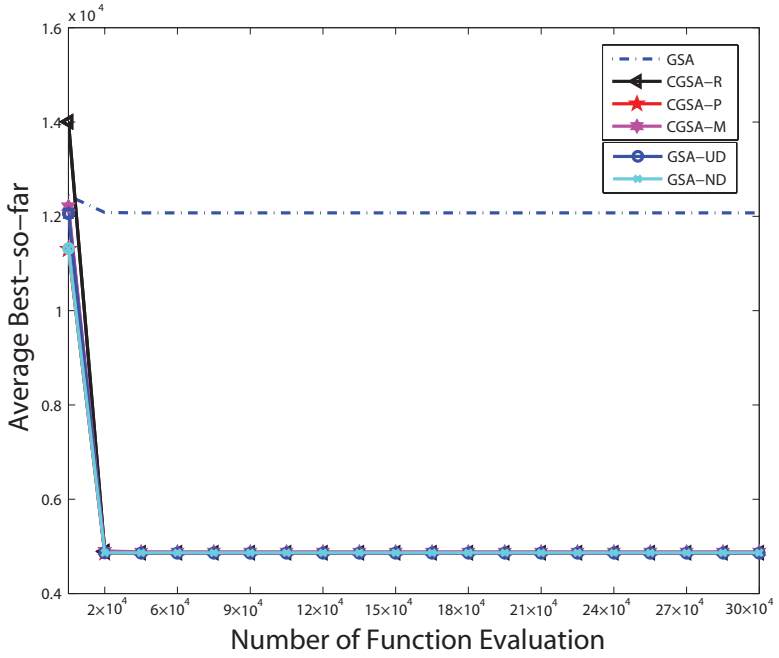


Figure 4.44: Convergence graph of F30: solution along with the number of function evaluation.

lelly, or memory-selectively incorporated into GSA, respectively. Experimental results based on a set of 48 benchmark optimization functions verify the effectiveness and robustness of the proposed MCGSA. Especially, the parallelly embedding scheme for GSA is demonstrated to be the most effective based on the Friedman test. This study opens the door to the following future researches:

1. MCGSA should be verified on other practical problems, especially engineering optimization problems.

2. The effectiveness of the proposed multiple chaos incorporation scheme should be applied on other meta-heuristics to further reveal its effects.

3. Through our experimental results, we find that a certain chaotic map is effective for some specific optimization functions. The influence of the distinct chaotic search dynamics on the algorithm should be further studied.

4. As each chaotic map has an inherent Lyapunov exponent which reflects its

chaotic degree, a Lyapunov exponent based adaptive multiple chaos incorporation scheme should also be designed.

Table 4.12: Experimental results of benchmark functions using traditional GSA, CGSA-R, CGSA-P, CGSA-M, GSA-UD and GSA-ND under the same maximum number of function evaluations.

Algorithm	F1	F2	F3	F4	F5	F6
GSA	1.30E-17 ± 4.33E-18	2.32E-08±3.07E-09	2.70E+02±9.52E+01	3.45E-09±6.91E-10	2.83E+01±1.14E+01	0.00E+00±0.00E+00
CGSA-R	**1.92E-18 ±4.68E-19**	**2.30E-08 ±4.62E-09**	**2.19E+02 ±8.92E+01**	1.21E-01 ±4.80E-01	9.59E+00 ±1.15E+01	**0.00E+00 ±0.00E+00**
CGSA-P	2.31E-18 ±8.88E-19	2.40E-08 ±4.20E-09	2.29E+02 ±8.30E+01	**3.30E-09 ±6.37E-10**	**5.42E+00 ±2.15E+01**	**0.00E+00 ±0.00E+00**
CGSA-M	1.97E-18 ±5.70E-19	2.34E-08 ±3.51E-09	2.40E+02 ±8.47E+01	2.56E-03 ±1.40E-02	1.55E+01 ±1.78E+01	**0.00E+00 ±0.00E+00**
GSA-UD	6.84E-18 ±2.05E-18	2.35E-08 ±3.22E-09	2.55E+02 ±7.11E+01	5.01E-02 ±2.75E-01	2.45E+01 ±4.29E+01	0.00E+00 ±0.00E+00
GSA-ND	5.87E-18 ±2.34E-18	2.32E-08 ±3.31E-09	2.55E+02 ±9.08E+01	7.03E-03 ±3.03E-02	2.75E+01 ±4.08E+01	0.00E+00 ±0.00E+00
Algorithm	F7	F8	F9	F10	F11	F12
GSA	2.18E-02 ±1.04E-02	-2.89E+03 ±4.49E+02	1.60E+01 ±3.84E+00	2.36E-09 ±2.35E-10	4.46E+00 ±2.07E+00	4.58E-02 ±1.17E-01
CGSA-R	1.08E-02 ±4.95E-03	**-3.23E+03 ±5.73E+02**	**1.49E+01 ±3.36E+00**	**1.08E-09 ±1.36E-10**	3.67E+00 ±1.44E+00	4.65E-02 ±1.14E-01
CGSA-P	**1.05E-02 ±4.81E-03**	-2.93E+03 ±4.65E+02	1.52E+01 ±3.56E+00	1.91E-09 ±2.93E-10	4.05E+00 ±1.75E+00	**3.13E-02 ±6.20E-02**
CGSA-M	1.18E-02 ±4.64E-03	-2.90E+03 ±4.97E+02	1.52E+01 ±3.99E+00	1.95E-09 ±4.05E-10	**2.77E+00 ±8.92E-01**	6.04E-02 ±9.90E-02
GSA-UD	1.35E-02 ±5.93E-03	-2.95E+03 ±4.47E+02	1.66E+01 ±3.82E+00	1.98E-09 ±3.15E-10	3.91E+00 ±1.79E+00	4.84E-02 ±8.49E-02
GSA-ND	1.22E-02 ±5.01E-03	-3.11E+03 ±5.54E+02	1.65E+01 ±3.52E+00	1.94E-09 ±3.12E-10	4.08E+00 ±1.63E+00	6.87E-02 ±1.09E-01
Algorithm	F13	F14	F15	F16	F17	F18
GSA	3.66E-04 ±2.01E-03	6.32E+00 ±3.10E+00	6.96E-03 ±4.75E-03	-1.03E+00 ±4.97E-16	3.98E-01 ±0.00E+00	3.00E+00 ±5.89E-15
CGSA-R	6.75E-19 ±1.86E-19	4.80E+00 ±3.24E+00	**4.09E-03 ±2.47E-03**	**-1.03E+00 ±4.70E-16**	**3.98E-01 ±0.00E+00**	**3.00E+00 ±3.39E-15**
CGSA-P	**6.39E-19 ±2.53E-19**	5.35E+00 ±3.74E+00	5.11E-03 ±3.32E-03	-1.03E+00 ±4.79E-16	**3.98E-01 ±0.00E+00**	3.00E+00 ±4.07E-15
CGSA-M	6.90E-04 ±2.63E-03	**4.40E+00 ±3.30E+00**	4.63E-03 ±2.62E-03	**-1.03E+00 ±4.70E-16**	**3.98E-01 ±0.00E+00**	3.00E+00 ±3.59E-15
GSA-UD	3.66E-04 ±2.01E-03	5.00E+00 ±3.45E+00	5.42E-03 ±2.72E-03	-1.03E+00 ±4.70E-16	3.98E-01 ±0.00E+00	3.00E+00 ±4.06E-15
GSA-ND	3.66E-04 ±2.01E-03	4.42E+00 ±3.70E+00	4.76E-03 ±3.27E-03	-1.03E+00 ±4.79E-16	3.98E-01 ±0.00E+00	3.00E+00 ±4.29E-15
Algorithm	F19	F20	F21	F22	F23	F24
GSA	-3.86E+00 ±3.17E-03	-3.32E+00 ±1.49E-15	-6.42E+00 ±3.80E+00	-1.01E+01 ±1.14E-15	-1.05E+01 ±2.68E-15	1.69E+03 ±8.03E+02
CGSA-R	-3.86E+00 ±2.47E-03	**-3.32E+00 ±1.49E-15**	-6.03E+00 ±3.71E+00	-1.01E+01 ±1.39E+00	**-1.06E+01 ±1.32E+00**	**1.10E+03 ±6.69E+02**
CGSA-P	**-3.86E+00 ±1.03E-03**	-3.32E+00 ±1.54E-15	-6.75E+00 ±3.72E+00	-1.02E+01 ±1.22E+00	-1.00E+01 ±1.61E+00	1.30E+03 ±8.96E+02
CGSA-M	-3.86E+00 ±3.24E-03	-3.32E+00 ±2.57E-02	**-7.93E+00 ±3.46E+00**	**-1.02E+01 ±7.49E-01**	-9.87E+00 ±2.05E+00	1.67E+03 ±9.01E+02
GSA-UD	-3.86E+00 ±2.30E-03	-3.32E+00 ±1.57E-15	-7.30E+00 ±3.58E+00	-1.02E+01 ±7.71E-01	-1.03E+01 ±1.48E+00	1.26E+03 ±7.12E+02
GSA-ND	-3.86E+00 ±2.59E-03	-3.32E+00 ±1.57E-15	-7.01E+00 ±2.96E+00	-1.02E+01 ±7.92E-01	-1.00E+01 ±1.67E+00	1.43E+03 ±6.54E+02

Table 4.13: Experimental results of benchmark functions using traditional GSA, CGSA-R, CGSA-P, CGSA-M, GSA-UD and GSA-ND under the same maximum number of function evaluations.

Algorithm	F25	F26	F27	F28	F29	F30
GSA	2.02E+04 ±2.14E+03	4.54E+07 ±5.13E+07	6.60E+04 ±1.39E+04	2.29E+04 ±2.80E+03	1.13E+08 ±5.02E+07	1.21E+04 ±2.51E+02
CGSA-R	1.99E+04 ±2.19E+03	3.09E+07 ±2.72E+07	6.29E+04 ±9.66E+03	2.25E+04 ±2.40E+03	1.11E+08 ±6.50E+07	4.93E+03 ±9.86E+01
CGSA-P	**1.97E+04 ±2.05E+03**	4.63E+07 ±4.96E+07	**6.22E+04 ±1.20E+04**	2.28E+04 ±2.46E+03	**1.08E+08 ±5.79E+07**	**4.86E+03 ±1.17E+01**
CGSA-M	2.04E+04 ±2.16E+03	**2.92E+07 ±2.08E+07**	6.29E+04 ±1.09E+04	2.24E+04 ±2.16E+03	1.36E+08 ±6.82E+07	4.88E+03 ±5.36E+01
GSA-UD	2.03E+04 ±2.48E+03	4.84E+07 ±4.60E+07	6.56E+04 ±1.19E+04	**2.23E+04 ±2.11E+03**	1.13E+08 ±5.00E+07	4.86E+03 ±2.84E+01
GSA-ND	2.01E+04 ±1.78E+03	3.49E+07 ±2.26E+07	7.02E+04 ±1.42E+04	2.35E+04 ±2.12E+03	1.25E+08 ±7.60E+07	4.86E+03 ±5.81E+01
Algorithm	F31	F32	F33	F34	F35	F36
GSA	2.03E+01 ±9.36E-02	3.92E+01 ±6.07E+00	3.53E+01 ±7.74E+00	1.86E+00 ±1.38E+00	3.01E+03 ±3.94E+03	5.55E+00 ±1.23E+00
CGSA-R	2.03E+01 ±1.06E-01	4.15E+01 ±8.41E+00	**3.26E+01 ±7.98E+00**	1.30E+00 ±1.22E+00	2.46E+03 ±2.62E+03	5.39E+00 ±1.40E+00
CGSA-P	**2.03E+01 ±7.79E-02**	**4.05E+01 ±8.96E+00**	3.56E+01 ±5.81E+00	1.86E+00 ±1.17E+00	5.15E+03 ±6.84E+03	5.75E+00 ±1.30E+00
CGSA-M	2.03E+01 ±9.36E-02	4.25E+01 ±7.67E+00	3.57E+01 ±5.67E+00	**1.22E+00 ±1.31E+00**	**2.04E+03 ±3.00E+03**	**5.28E+00 ±1.58E+00**
GSA-UD	2.03E+01 ±9.15E-02	4.14E+01 ±6.20E+00	3.39E+01 ±6.42E+00	1.58E+00 ±1.40E+00	2.48E+03 ±3.36E+03	6.14E+00 ±1.34E+00
GSA-ND	2.03E+01 ±8.02E-02	4.10E+01 ±6.62E+00	3.33E+01 ±6.13E+00	2.38E+00 ±1.14E+00	5.31E+03 ±8.08E+03	5.61E+00 ±1.32E+00
Algorithm	F37	F38	F39	F40	F41	F42
GSA	1.42E+01 ±1.29E-01	3.09E+02 ±3.74E+01	2.94E+02 ±2.12E+02	3.35E+02 ±2.58E+02	9.68E+02 ±6.15E+01	9.59E+02 ±4.44E+01
CGSA-R	1.41E+01 ±1.59E-01	3.08E+02 ±2.57E+01	2.31E+02 ±2.08E+02	2.58E+02 ±2.06E+02	9.61E+02 ±4.55E+01	**9.58E+02 ±4.58E+01**
CGSA-P	1.41E+01 ±1.99E-01	3.07E+02 ±2.54E+01	2.65E+02 ±2.12E+02	**2.08E+02 ±1.75E+02**	9.63E+02 ±5.68E+01	9.59E+02 ±1.42E+07
CGSA-M	**1.40E+01 ±2.71E-01**	**3.04E+02 ±1.83E+01**	**1.76E+02 ±2.16E+02**	2.21E+02 ±1.62E+02	**9.52E+02 ±6.22E+01**	9.61E+02 ±4.65E+01
GSA-UD	1.41E+01 ±2.26E-01	3.04E+02 ±1.83E+01	2.69E+02 ±1.86E+02	2.38E+02 ±1.95E+02	9.59E+02 ±6.89E+01	9.62E+02 ±4.32E+01
GSA-ND	1.41E+01 ±2.19E-01	3.09E+02 ±3.76E+01	2.95E+02 ±2.17E+02	2.42E+02 ±1.87E+02	9.78E+02 ±1.65E+01	9.67E+02 ±1.71E+01
Algorithm	F43	F44	F45	F46	F47	F48
GSA	9.67E+02 ±3.48E+01	8.24E+02 ±3.29E+02	9.32E+02 ±2.21E+01	8.38E+02 ±3.09E+02	2.00E+02 ±7.96E-13	1.68E+03 ±8.44E+00
CGSA-R	9.65E+02 ±3.60E+01	7.17E+02 ±3.12E+02	**9.29E+02 ±1.91E+01**	8.61E+02 ±2.96E+02	**2.00E+02 ±1.16E+00**	1.68E+03 ±1.10E+01
CGSA-P	9.66E+02 ±3.35E+01	**6.74E+02 ±2.94E+02**	9.36E+02 ±1.88E+01	8.56E+02 ±2.95E+02	3.09E+02 ±2.85E+02	**1.67E+03 ±9.35E+00**
CGSA-M	9.61E+02 ±5.61E+01	7.38E+02 ±3.18E+02	9.31E+02 ±1.95E+01	**6.97E+02 ±2.70E+02**	2.56E+02 ±2.13E+02	1.68E+03 ±8.30E+00
GSA-UD	9.55E+02 ±5.41E+01	7.17E+02 ±3.12E+02	9.29E+02 ±1.41E+01	8.00E+02 ±2.93E+02	2.00E+02 ±5.73E-13	1.68E+03 ±6.75E+00
GSA-ND	**9.51E+02 ±5.24E+01**	8.05E+02 ±3.25E+02	9.29E+02 ±1.90E+01	7.94E+02 ±2.94E+02	2.63E+02 ±2.40E+02	1.68E+03 ±1.12E+01

Chapter 5

Training A Dendritic Neural Model with Genetic Algorithm for Classification Problems

5.1 Introduction

In 1943, McCulloch and Pitts proposed the first mathematical model of a neural cell to simulate the computational mechanism of biological neurons [61], and it has been widely used as a basic unit of multilayer neural network (MLP). The MLP has been successfully applied to numerous fields, such as function approximation, pattern recognition, design optimization and associative memories [62–64]. High performance and low computation cost make MLP becoming a remarkably popular computational tool over the last few decades. The structure with multiple hidden layers makes it become capable to solve the non-linear problems. On the other hand, the Error Back Propagation (BP) training algorithm for MLP usually suffers from the problems of easily trapping into local minimum, and becomes an obstacle hindering its further development [65].

With in-depth research in the study of biological neurons' computational organization, a fierce debate has been drawn that a single neuron can solve linearly non-separable problems, for which single-layer perceptron of McCulloch and Pitts' model is unable to realize such a complex computation capacity [66]. Scientists focus on the roles of dendritic structure during the neural computation. It is well-known that

10^5 dendritic trees have been reconstructed in different cephalic regions, which are highly specialized for brain functions [67]. Inspired by these preliminary studies in neuroscience, several single neural models operated by considering dendrite structures have been proposed and applied into intelligent computing [68–71]. Compared to these subjectively constructed neural model, Legenstein and Maass [72] introduced a novel experimentally based on phenomenological model, which just used spike timing dependent plasticity and branch strength potentiation to solve a binding problem successfully, whereas Legenstein and Maass's model is still proved to be disabled to solve the linearly non-separated problems [73].

In this paper, we have innovatively proposed a dendritic neural model focusing on the computational capabilities of single branch of dendrites. Our model is composed of three layers, namely, synaptic layer, dendritic layer, and soma body. The synaptic layer is the intermediary connecting to other neurons, and it is regarded as an input layer in our model. The dendritic layer is constituted by several branches. Each branch has its specified branch threshold. The signals of synaptic layer on that branch will multiply each other, and the results are then undergone to be compared with its branch threshold. The multiplication operation is inspired by the fact of the actual existence in biological neural models, such as visual systems [74] and auditory process [75]. Such calculation mechanism makes each branch of the dendrites correspond to a specified distribution in coordinates, while the number of distributions is determined by the input dimensionality, and the distributions could classify training data effectively.

Since the equation of the dendritic layer is not differential, the standard BP algorithm behaves unsatisfiedly to train our model. Alternatively, we turn our attention to heuristic optimization algorithm, such as Differential Evolution (DE), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). Based on the well-known No Free Lunch theorem (NFL) [76], no heuristic algorithm is capable of suiting for all the optimization problems. After comparing the performance of each algorithm, we finally choose the GA. The effectiveness of GA in training our proposed model are verified in two benchmark classification experiments.

The remaining of the paper is organized as follows. Section II introduces the proposed dendritic neuron model in details. The GA training algorithm is described in Section III. Section IV presents experimental results and discussions. Finally, conclusions and remarks for the future are presented in Section V.

5.2 Architecture and properties

The morphological structure is illustrated in Fig. 5.1, which show that it consists of three layers, namely the synaptic layer, the dendritic layer and the soma body. The synapses are able to receive signals from axons of the other neurons, and each input signal connects to two synapses on one branch of the dendrites with different weights and thresholds. The non-linear calculation in synaptic layer can be formulated as follow:

$$Y_{im}^{j} = \frac{1}{1 + e^{-w_{im}(x_i - \theta_{im})}} \quad (j = 1, 2) \tag{5.1}$$

where x_i indicates the i-th input signal, $w_{i,m}$ and $\theta_{i,m}$ correspond to the weights and threshold values of the synapses on m-th branch of the dendrites. It is notable that since all the inputs x_i are normalized to [0, 1] to eliminate the influence of different measure unites in various problems. On every independent branch of the dendrites, the outputs of synaptic layers would multiply each other and compare the result with the branch threshold θ_{branch}, which is set to be 0.5^{2*I} in our simulation. Then the outcomes are transferred to the soma body as the final classification category. The corresponding equation is expressed below.

$$Z_m = \begin{cases} 0, & \prod_{i=1}^{m}(\prod_{j=1}^{2} Y_{im}^{j}) < \theta_{branch} \\ \\ 1, & \prod_{i=1}^{m}(\prod_{j=1}^{2} Y_{im}^{j}) \geq \theta_{branch} \end{cases} \tag{5.2}$$

The soma body collects the branch signals, then compares the summation of signals with its threshold (usually set to be 0.5 in our experiment), which can be

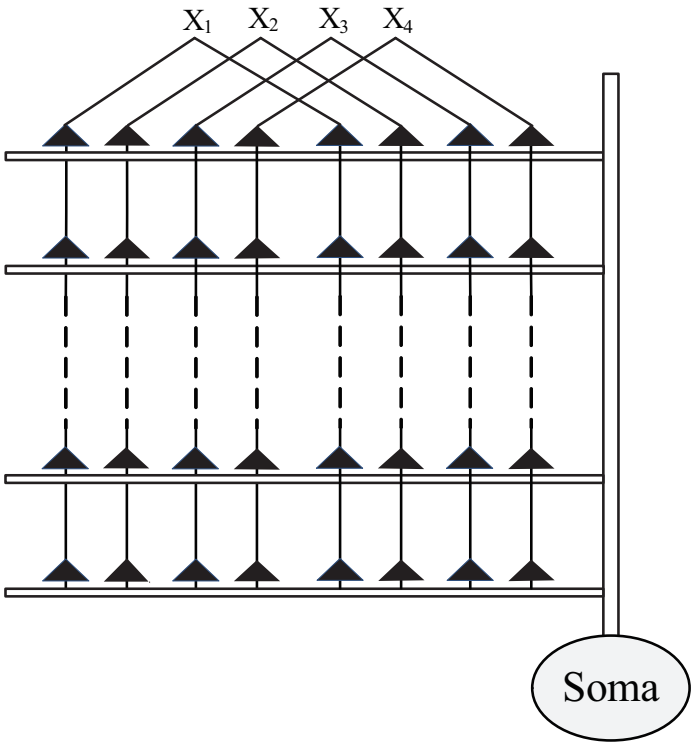


Figure 5.1: The architecture of the proposed dendritic neuron model.

formulated as follows:

$$O = \frac{1}{1 + e^{-5*(\sum_{m=1}^{M} Z_m - 0.5)}} \tag{5.3}$$

Corresponding to different values of weights and thresholds, one branch of the dendrites will correspond to one of these three distribution states, which are shown in Fig. 5.2.

The positive and negative properties of weights determine the shapes of the distribution. The sizes of the absolute values of weights determine the curvature of the arc, while the thresholds determine the position on the axis. Through the observation, these arcs divide the plane into two parts. Hence, based on such distinct property, the model is able to mapping the training data to different classifications.

In Fig. 5.2(a), the first distribution is a closed plane area, which has upper and lower bounds on both axes. It is concluded from the fact that the weights of each input

on this branch have the different positive and negative values. Fig. 5.2(b) depicts that one axis of the second distribution has no boundary. Fig. 5.2(c) illustrates that both axes of the third distribution have no boundary. For n-dimensional training inputs, the number of distribution conditions will be n+1.

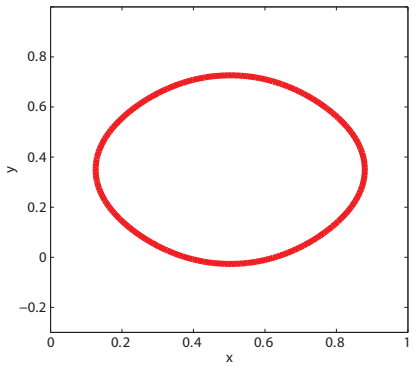
It is worth emphasizing that since we use a summation operation in soma computation, training dots are allocated into the same classification, which makes it only need to satisfy the distribution of one branch of dendrites. Then we need a learning algorithm to train our model to figure out the appropriate parameters, which will be elaborated in the next section.

5.3 Training algorithm

The weights and threshold values of synapses are the specific parameters which need to be trained in our model. Once the parameters are set, the morphological topology would be determined. Genetic Algorithm (GA) is applied to train our model. Before introducing the GA, we firstly present the definition solution encoding and fitness function. One candidate solution is constituted of all the parameters of the model corresponding to a dendritic model. Before training, initialized solution will be encoded as binary string sequences. For a candidate solution, the fitness function is defined as Mean Square Error between the estimated output O and the teacher signal T, which are shown in the following equation.

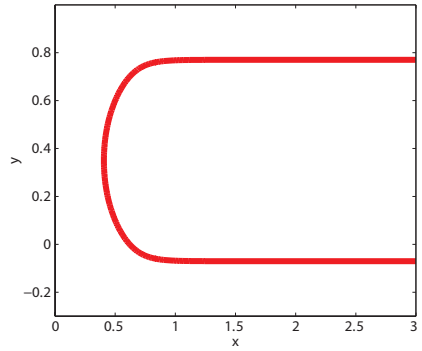
$$\text{MSE} = \sum_{j=1}^{J} (O_j - T_j)^2 \tag{5.4}$$

where J is the number of training samples. The procedure for training our dendritic model could be viewed as a process flow which is shown in Fig. 5.3. First, the evolution starts from a randomly initial population, then three genetic operator are utilized, say selection, crossover and mutation, to generate a new population. The process will continue until a termination condition is fulfilled.



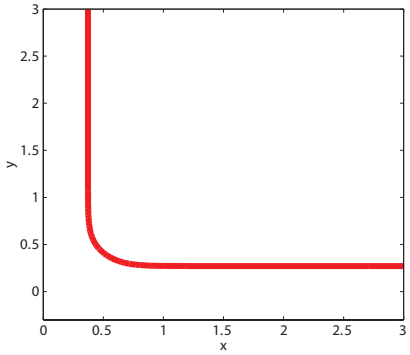
$$\begin{cases} W^1_{m,1} \cdot W^2_{m,1} < 0 \\ W^1_{m,2} \cdot W^2_{m,2} < 0 \end{cases}$$

Fig. 2(a)



$$\begin{cases} W^1_{m,1} \cdot W^2_{m,1} > 0 \\ W^1_{m,2} \cdot W^2_{m,2} < 0 \end{cases}$$

Fig. 2(b)



$$\begin{cases} W^1_{m,1} \cdot W^2_{m,1} > 0 \\ W^1_{m,2} \cdot W^2_{m,2} > 0 \end{cases}$$

Fig. 2(c)

Figure 5.2: The distributions of dendritic structures.

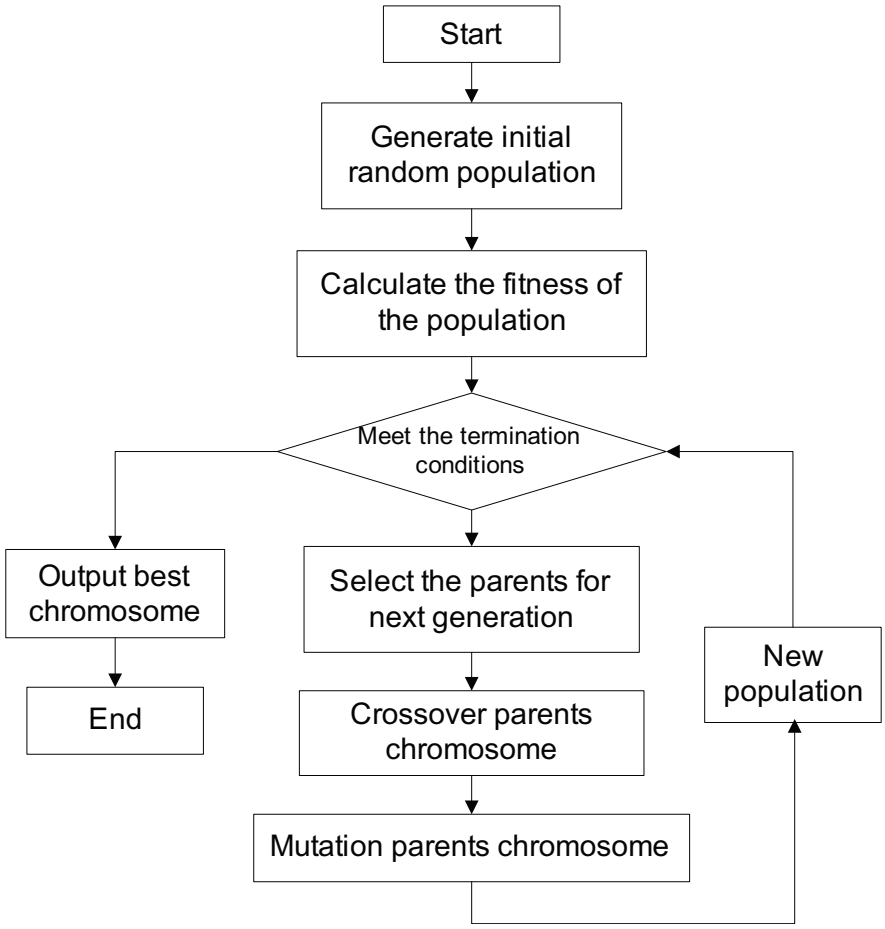


Figure 5.3: The procedure of genetic algorithm for training the dendritic model.

5.4 Simulation

The classification performance of our proposed model has been investigated by two nontrivial benchmark problems: the famous Exclusive OR (XOR) problem and two intertwined spirals problems [77]. Both classification problems are based on two-dimensional synthetic datasets, aiming to demonstrate our model properties visually. The initial values of genetic algorithm for these benchmark problem are shown in Table 2.

5.4.1 Classic XOR problem

The classic XOR problem is a famous non-linear benchmark problem. The training data with two inputs are shown in Table 1.

Table 5.1: Initial parameter of genetic algorithm

Type	Binary Representation
Selection	Roulette wheel
Crossover	Single point
Mutation	Uniform (probability=0.02)
Population size	50(XOR) 100(Spiral)
Maximum number of generations	1000(XOR) 3000(Spiral)

Table 5.2: Exclusive OR problem.

X_2 \ X_1	0	1
0	0	1
1	1	0

5.4.2 Two intertwined spirals problem

The spirals dataset is generated by the following expressions:

$$x_c(\theta) = \frac{2(-1)^{1-c}}{\pi}\theta\cos(\theta) \tag{5.5}$$

$$y_c(\theta) = \frac{8(-1)^{1-c}}{3\pi}\theta\sin(\theta) \tag{5.6}$$

where $[x_c, y_c]$ denotes the position in the planar reference frame, $c \in [0, 1]$ represents the spiral classifications, and θ is the angle in radians. Fig. 5.4 shows the spiral dataset which is trained in our experiment.

The performance of the proposed model is summarized in Table 3, where branch number records the number of the branch of dendrites left in the trained models. MSE is the final fitness function of our training algorithm. Accuracy represents the correct classification rates of the two benchmark problems. The distributions in co-ordinate corresponding to dendritic structures are shown in Fig. 5.5 and Fig. 5.6respectively. After the evolutionary process, there are only one distribution left in the XOR problem, which classifies the four training dots successfully. The dendritic

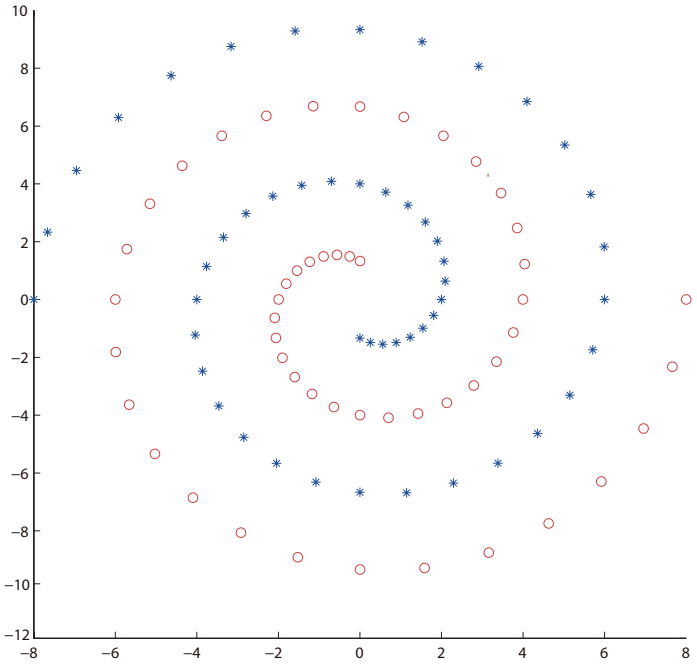


Figure 5.4: Spiral dataset trained in our experiment

Table 5.3: Results of two benchmark problems

Result	XOR	Spiral
Branch number	1	13
MSE	0.0034	0.0436
Accuracy	100%	97%

structure of spiral problem has three kinds of distributions. Most of them are the first distribution, and the accuracy of the spiral problem is 97%. The three missing mistaken points can also be found in Fig. 5.6. Based on these results, we can draw a conclusion that our model is a competitive classifier to solve these benchmark problems.

5.5 Conclusion

In this paper, a novel model with dendritic structure is proposed, which is inspired by the biological neuron model. We choose the genetic algorithm as the training algorithm. Branches of dendrites will correspond to three kinds of distributions in

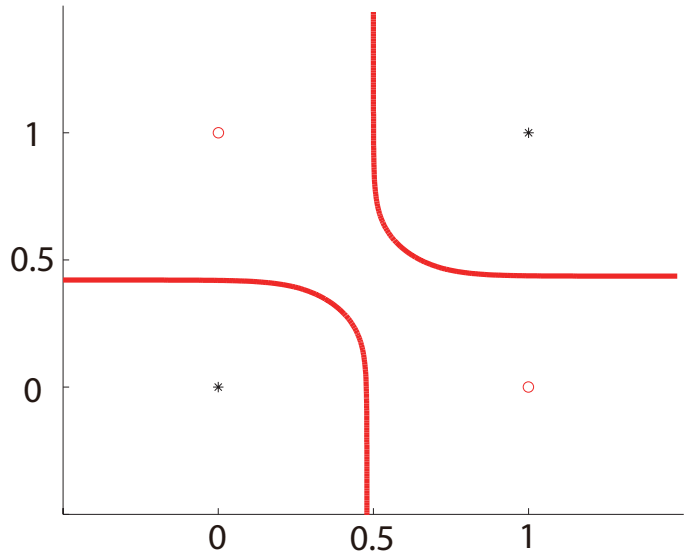


Figure 5.5: The distribution of the trained dendritic structure for XOR problem.

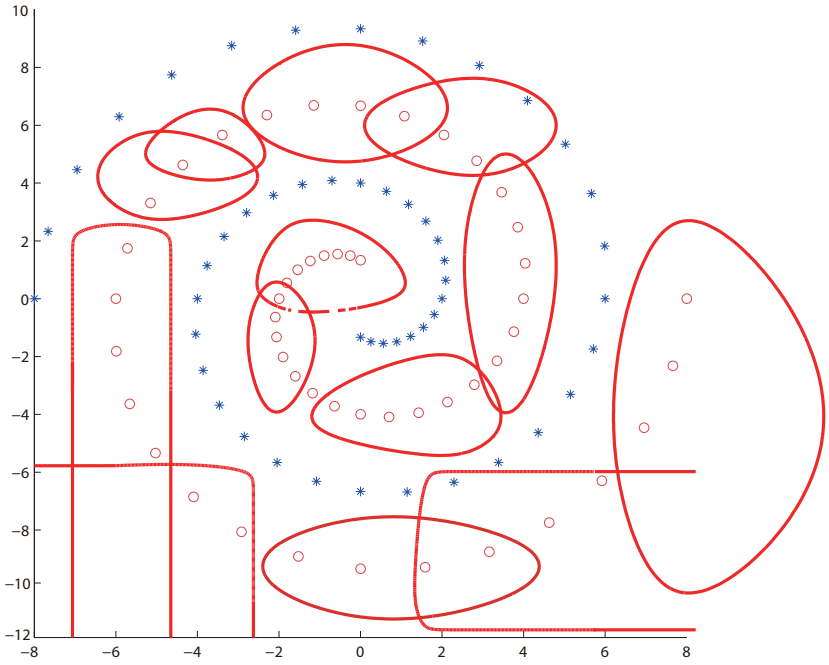


Figure 5.6: The distribution of the trained dendritic structure for spiral problem.

coordinates. This property makes our model solve two benchmark problems effectively, which has been verified in our experiment. More comparative trial, statistical test and theoretical analysis will be investigated in our future research.

Chapter 6

Conclusions

This paper in order to improve the search ability of GSA, we are supplying the GSA with 12 kinds of single chaotic search and 3 kinds of multiple chaotic search, the results show that, whether embedded single chaotic search, or multiple chaotic search are effectively improved the GSA's search ability and get the higher quality solutions. Secondly, the classification problem can be solved more effectively by the fission algorithm, which makes the classification problem more intuitive and quickly.

Evolutionary computation is a kind of robust method, which can be adapted to different environments, and the effective solution can be obtained in most cases. It gives a coding scheme for the whole parameter space of the problem, rather than directly on the specific parameters of the problem, not from a single initial point to start the search, but from a set of initial point to search. Search is used in the objective function value of the information, which cannot use the derivative information of the objective function or specific issues related to the special knowledge. Therefore evolutionary algorithm have widely extensive applicability, highly nonlinear, easy to modify and parallelism.

First chapter in this paper, we introduced the concept of meta-heuristic optimization, from its history of development, birth of the optimization algorithm to the characteristics of heuristic algorithms, we can see reasonable improvement and consolidation of evolutionary algorithm, can effectively improve the ability to solve the problem of optimization algorithm.

Computational intelligence algorithms have a common characteristic is the imita-

tion of certain aspects of human intelligence to simulate the human intelligence, the design optimization algorithm is to achieve biological wisdom, computer programming. However, the computational intelligence of these different research areas have different characteristics, although they have to imitate human and other biological intelligence in common, but there are some differences on the concrete methods.

However at the present stage, the development of computational intelligence is also facing serious challenges, one of the important reasons is computational intelligence is still a lack of solid mathematical foundation, also cannot be subjects such as physics, chemistry, astronomy, and so that freely using mathematical tools to solve the calculation problem. Although the neural network has more perfect theoretical basis, but such as evolutionary computation and other important computational intelligence technology has not been improved mathematical basis. The analysis and proof of the stability and convergence of computational intelligence algorithms are still in the research stage. Through numerical experiment method and the concrete application method testing computational intelligence algorithm is effective and efficiency is the important method of computational intelligence algorithms.

In the third chapter, we introduce the concept of chaos. Chaos is the general term of the system stochastic behavior, and its root lies in the nonlinear interaction. Chaos is not clutter, it is different from the equilibrium state, but a sequence. Movement of the most common form in the nature, is not completely determined, also is not completely random, but somewhere between in two, which is the study of the important significance of the system in the stochastic behavior. To clearly define chaos, it is necessary to discuss the dependence of the system on the minor changes of the initial value.

From the experimental results and data, embedded the chaotic, the ability of the GSA have been improved, especially parallel chaotic CGSA-P in which other algorithms are stuck in a local minimum, it can still to find better solutions, and under the precondition of not consume time cost.

Bibliography

[1] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[2] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing: an international journal*, vol. 8, no. 2, pp. 239–287, 2009.

[3] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[4] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *Journal of statistical physics*, vol. 34, no. 5-6, pp. 975–986, 1984.

[5] M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. Winfield, *Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings*. Springer, 2008, vol. 5217.

[6] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766.

[7] 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.

[8] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied mathematics and computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[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] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[11] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on.* IEEE, 2009, pp. 210–214.

[12] T. Ganesan, I. Elamvazuthi, K. Z. K. Shaari, and P. Vasant, "Swarm intelligence and gravitational search algorithm for multi-objective optimization of synthesis gas production," *Applied Energy*, vol. 103, pp. 368–374, 2013.

[13] R. W. Sperry, "Cerebral organization and behavior," *Science*, vol. 133, no. 3466, pp. 1749–1757, 1961.

[14] N. Ganesan, K. Venkatesh, M. Rama, and A. M. Palani, "Application of neural networks in diagnosing cancer disease using demographic data," *International Journal of Computer Applications (0975-8887)*, vol. 1, no. 26, 2010.

[15] S. H. Kellert, *In the wake of chaos: Unpredictable order in dynamical systems.* University of Chicago press, 1994.

[16] V. G. Ivancevic and T. T. Ivancevic, *Complex nonlinearity: chaos, phase transitions, topology change and path integrals.* Springer Science & Business Media, 2008.

[17] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining lyapunov exponents from a time series," *Physica D: Nonlinear Phenomena*, vol. 16, no. 3, pp. 285–317, 1985.

[18] C. Kyrtsou and W. C. Labys, "Evidence for chaotic dependence between us inflation and commodity prices," *Journal of Macroeconomics*, vol. 28, no. 1, pp. 256–266, 2006.

[19] ——, "Detecting positive feedback in multivariate time series: the case of metal prices and us inflation," *Physica A: Statistical Mechanics and its Applications*, vol. 377, no. 1, pp. 227–229, 2007.

[20] C. Kyrtsou and C. E. Vorlow, "Complex dynamics in macroeconomics: A novel approach," in *New Trends in Macroeconomics*. Springer, 2005, pp. 223–238.

[21] X. Wang and J. Zhao, "An improved key agreement protocol based on chaos," *Communications in Nonlinear Science and Numerical Simulation*, vol. 15, no. 12, pp. 4052–4057, 2010.

[22] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[23] I. BoussaïD, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[24] X. Yao and Y. Xu, "Recent advances in evolutionary computation," *Journal of Computer Science and Technology*, vol. 21, no. 1, pp. 1–18, 2006.

[25] K. Kameyama, "Particle swarm optimization-a survey," *IEICE Transactions on Information & Systems*, vol. 92, no. 7, pp. 1354–1361, 2009.

[26] S. C. 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.

[27] S. Wang, S. C. Gao, Aorigele, Y. Todo, and Z. Tang, "A multi-learning immune algorithm for numerical optimization," *IEICE Trans. on Fundamentals of Elec-*

tronics, Communications and Computer Sciences, vol. 98, no. 1, pp. 362–377, 2015.

[28] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "Gsa: A gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[29] W. Zhang, P. Niu, G. Li, and P. Li, "Forecasting of turbine heat rate with online least squares support vector machine based on gravitational search algorithm," *Knowledge-Based Systems*, vol. 39, pp. 34–44, 2013.

[30] B. Ji, X. Yuan, Z. Chen, and H. Tian, "Improved gravitational search algorithm for unit commitment considering uncertainty of wind power," *Energy*, vol. 67, pp. 52–62, 2014.

[31] C. Lopez-Molina, H. Bustince, J. Fernandez, P. Couto, and B. De Baets, "A gravitational approach to edge detection based on triangular norms," *Pattern Recognition*, vol. 43, no. 11, pp. 3730–3741, 2010.

[32] C. Li, J. Zhou, B. Fu, P. Kou, and J. Xiao, "T–s fuzzy model identification with a gravitational search-based hyperplane clustering algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 2, pp. 305–317, 2012.

[33] N. M. Sabri, M. Puteh, and M. R. Mahmood, "A review of gravitational search algorithm," *Int. J. Advance. Soft Comput. Appl*, vol. 5, no. 3, pp. 1–39, 2013.

[34] C. Li and J. Zhou, "Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm," *Energy Conversion and Management*, vol. 52, no. 1, pp. 374–381, 2011.

[35] B. Shaw, V. Mukherjee, and S. Ghoshal, "A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems," *International Journal of Electrical Power & Energy Systems*, vol. 35, no. 1, pp. 21–33, 2012.

[36] S. Yazdani, H. Nezamabadi-pour, and S. Kamyab, "A gravitational search algorithm for multimodal optimization," *Swarm and Evolutionary Computation*, vol. 14, pp. 1–14, 2014.

[37] R. Caponetto, L. Fortuna, S. Fazzino, and M. G. Xibilia, "Chaotic sequences to improve the performance of evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 289–304, 2003.

[38] T. Xiang, X. Liao, and K.-w. Wong, "An improved particle swarm optimization algorithm combined with piecewise linear chaotic map," *Applied Mathematics and Computation*, vol. 190, no. 2, pp. 1637–1645, 2007.

[39] S. Saremi, S. Mirjalili, and A. Lewis, "Biogeography-based optimisation with chaos," *Neural Computing and Applications*, vol. 25, no. 5, pp. 1077–1097, 2014.

[40] A. A. Heidari, R. A. Abbaspour, and A. R. Jordehi, "An efficient chaotic water cycle algorithm for optimization tasks," *Neural Computing and Applications*, pp. 1–29, 2015.

[41] M. Mitić, N. Vuković, M. Petrović, and Z. Miljković, "Chaotic fruit fly optimization algorithm," *Knowledge-Based Systems*, vol. 89, pp. 446–458, 2015.

[42] 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.

[43] A. H. Gandomi and X.-S. Yang, "Chaotic bat algorithm," *Journal of Computational Science*, vol. 5, no. 2, pp. 224–232, 2014.

[44] 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.

[45] R. Arul, G. Ravi, and S. Velusami, "Chaotic self-adaptive differential harmony search algorithm based dynamic economic dispatch," *International Journal of Electrical Power & Energy Systems*, vol. 50, pp. 85–96, 2013.

[46] A. Kazem, E. Sharifi, F. K. Hussain, M. Saberi, and O. K. Hussain, "Support vector regression with chaos-based firefly algorithm for stock market price forecasting," *Applied Soft Computing*, vol. 13, no. 2, pp. 947–958, 2013.

[47] Y. Li, Q. Wen, L. Li, and H. Peng, "Hybrid chaotic ant swarm optimization," *Chaos, Solitons & Fractals*, vol. 42, no. 2, pp. 880–889, 2009.

[48] S. Talatahari, B. F. Azar, R. Sheikholeslami, and A. Gandomi, "Imperialist competitive algorithm combined with chaos for global optimization," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 3, pp. 1312–1319, 2012.

[49] S. C. 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.

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

[51] J. Gleick, *Chaos: Making a New Science (Enhanced Edition)*. Open Road Media, 2011.

[52] M. S. Tavazoei and M. Haeri, "Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms," *Applied Mathematics and Computation*, vol. 187, no. 2, pp. 1076–1085, 2007.

[53] Y.-Y. He, J.-Z. Zhou, X.-Q. Xiang, H. Chen, and H. Qin, "Comparison of different chaotic maps in particle swarm optimization algorithm for long-term cascaded hydroelectric system scheduling," *Chaos, Solitons & Fractals*, vol. 42, no. 5, pp. 3169–3176, 2009.

[54] D. Yang, Z. Liu, and J. Zhou, "Chaos optimization algorithms based on chaotic maps with different probability distribution and search speed for global optimiza-

tion," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, no. 4, pp. 1229–1246, 2014.

[55] 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, 2009.

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

[57] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," *KanGAL report*, vol. 2005005, p. 2005, 2005.

[58] S. Garcia, A. Fernandez, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.

[59] J. Alcala-Fdez, L. Sanchez, S. Garcia, M. J. Del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, and V. M. Rivas, "Keel: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

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

[61] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[62] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, 2000.

[63] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance," *Neural Networks*, vol. 21, no. 2, pp. 427–436, 2008.

[64] F. Bortolozzi, A. de Souza Britto Jr, L. S. Oliveira, and M. Morita, "Recent advances in handwriting recognition," in *Proceedings of the International Workshop on Document Analysis*, 2005, pp. 1–30.

[65] S. S. Haykin, S. S. Haykin, S. S. Haykin, and S. S. Haykin, *Neural networks and learning machines*. Pearson Upper Saddle River, NJ, 2009.

[66] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[67] R. D. Cazé, M. Humphries, and B. Gutkin, "Passive dendrites enable single neurons to compute linearly non-separable functions," *PLoS Comput.Biol.*, vol. 9, no. 2, p. e1002867, 2013.

[68] G. X. Ritter and G. Urcid, "Lattice algebra approach to single-neuron computation," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 282–295, 2003.

[69] ——, "Learning in lattice neural networks that employ dendritic computing," in *Computational Intelligence Based on Lattice Theory*. Springer, 2007, pp. 25–44.

[70] 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.

[71] 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.

[72] R. Legenstein and W. Maass, "Branch-specific plasticity enables self-organization of nonlinear computation in single neurons," *The Journal of Neuroscience*, vol. 31, no. 30, pp. 10 787–10 802, 2011.

[73] R. P. Costa and P. J. Sjöström, "One cell to rule them all, and in the dendrites bind them," *Frontiers in Synaptic Neuroscience*, vol. 3, p. 5, 2011.

[74] E. Salinas and L. Abbott, "A model of multiplicative neural responses in parietal cortex," *Proceedings of the National Academy of Sciences*, vol. 93, no. 21, pp. 11 956–11 961, 1996.

[75] F. Gabbiani, H. G. Krapp, C. Koch, and G. Laurent, "Multiplicative computation in a visual neuron sensitive to looming," *Nature*, vol. 420, no. 6913, pp. 320–324, 2002.

[76] Y.-C. Ho and D. L. Pepyne, "Simple explanation of the no-free-lunch theorem and its implications," *Journal of Optimization Theory and Applications*, vol. 115, no. 3, pp. 549–570, 2002.

[77] K. J. Lang, "Learning to tell two spirals apart," in *Proc. of 1988 Connectionist Models Summer School*, 1988, pp. 52–59.