Research on Dendritic Neural Computation and Differential Evolution Algorithm

by

Chen Wei

A dissertation

submitted to the Graduate School of Science and Engineering for Education

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Engineering

University of Toyama

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

2016

(Submitted December 1, 2016)

Acknowledgements

First of all, I want to thanks to all those people who have advanced and helped me during my Ph.D. course, and especially the process of writing this thesis. Any progress that I have made is the result of their profound concern and selfless devotion. Particularly, I am deeply indebted to Prof. Zheng Tang at University of Toyama, my supervisor, who has offered me valuable suggestions in the academic studies. He kindly provided many incisive comments, useful suggestions, and constructive criticism to contribute greatly to the completion of this thesis. High tribute should be paid to Prof. Shangce Gao, who devotes a considerable portion of his time to reading my manuscripts and making suggestions for further revisions. Truly, without his painstaking efforts in revising and polishing my drafts, the completion of the present thesis would not have been possible. I am also greatly indebted to all my fellow classmates and my friends, who have helped me in the past few years, whether in study or lives. Finally, I should be indebted to my parents and girlfriend for their consistent support and encouragement.

Abstract

Artificial neural networks(ANNs) is the popular and promising area of artificial intelligence research. it has many learning methods and a variety of network architectures and it can be connected with different computational capabilities to produce neural networks. ANN is a purely computational model based on the human brain's organizational structure. In the past few years, the research of ANNs has made great progress, and successfully solved many modern computer practical problems in the fields of automatic control, intelligent robot, biology, economics, and pattern recognition and prediction estimation. Basically, ANNs always show good intelligence. Prediction is one of the main applications of ANN. We know that ANN's traditional prediction method has excellent classification and pattern recognition ability. Some special features make ANN a better predictive tool. Distinctive features also make predictions valuable and attractive. However, due to the high volatility, irregular motions and non-stationarity of the travel time series, traditional methods often suffer from prediction accuracy problems. In this study, with the rapid development of international tourism, it has become a growing industries with very fast speed in this world. Therefore, the prediction of tourism demand has been a challenge to the international tourism market. A new single dendritic neuron model (SDNM) is proposed to perform tourism demand forecasting. First, we use phase space reconstruction to analyze the characteristics of tourism and reconstruct the time series into appropriate phase space points. The maximum Lyapunov exponent is then used to identify the chaotic properties of the time series used to determine the prediction

limit. Finally, we use SDNM for short-term forecasting. The experimental results of monthly foreign tourists arriving in Japan show that the proposed SDNM model is more efficient and accurate than other neural networks including multilayer perceptrons, neuro-fuzzy inference systems, Elman networks and haploid neurons. On the other hand, multi-objective processing using -Doher differential evolution based on adaptive mutation will be described. Differential evolution (DE) is a well-known and robust population-based stochastic real parameter optimization algorithm in continuous space. DE has recently been shown to be superior to several well-known stochastic optimization methods in solving multi-objective problems. However, its performance is still limited in finding a uniform distribution and approaching the optimal Pareto front. To mitigate this limitation and reduce the computational cost, an adaptive mutation operator is introduced to avoid premature convergence by adaptively adjusting the mutation scale factor F and using the -dominance strategy to update the archives that store non-dominated solutions. Experiments based on five widely used multi-objective functions were performed. The simulation results demonstrate the effectiveness of our proposed approach in solving the Pareto frontier convergence and diversity aspects. The organizational structure of this paper is as follows. In Chapter 1, we give a brief description of ANNs for prediction. Chapter 2 describes some evolutionary computation (EC), such as Genetic Algorithm (GA), Differential Evolution (DE) algorithm and Particle Swarm Optimization (PSO) algorithm. Chapter 4 discusses the performance of SDNM in prediction. The adaptive mutation operator based on the multi-objective DE algorithm is reported in Chapter 5. Finally, the conclusions and future research will be discussed in Chapter 6.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Artificial Neural Networks

The brain consists of about 1011 neurons, which have more than 1015 connections. Neurons constitute the cell body (or soma), axons and dendrites, which will be shown in Fig.1.1. The branch receives the input as a synapse of the synapses, and then sends the information to the somatic cells. When the net stimulus reaches the threshold, neurons heat and send information through the axons to other neurons. Neurons can suppress or excite signals.

About 1011 neuron units are in the human nervous system. Neurons are information processing components with dendrites, cell bodies and axons. Synaptic transmission involves complex chemical and electrical processes. Changes in synaptic potentials triggered by sensory or chemical stimuli. Is used for the input signal to the neuronal dendritic surface and usually passively accept their soma. Dendrites are highly branched structures. It contributes to the initiation site of synaptic or receptor potential impulse through self propagation. The axon converts these signals into commonly known as spike trains.

ANN is the first McCulloc and Pitts proposed in 1943, they launched a research hotspot of artificial intelligence.ANN neural network algorithm and mathematical model based on the 1.2 is a research hotspot in the field of artificial intelligence, first

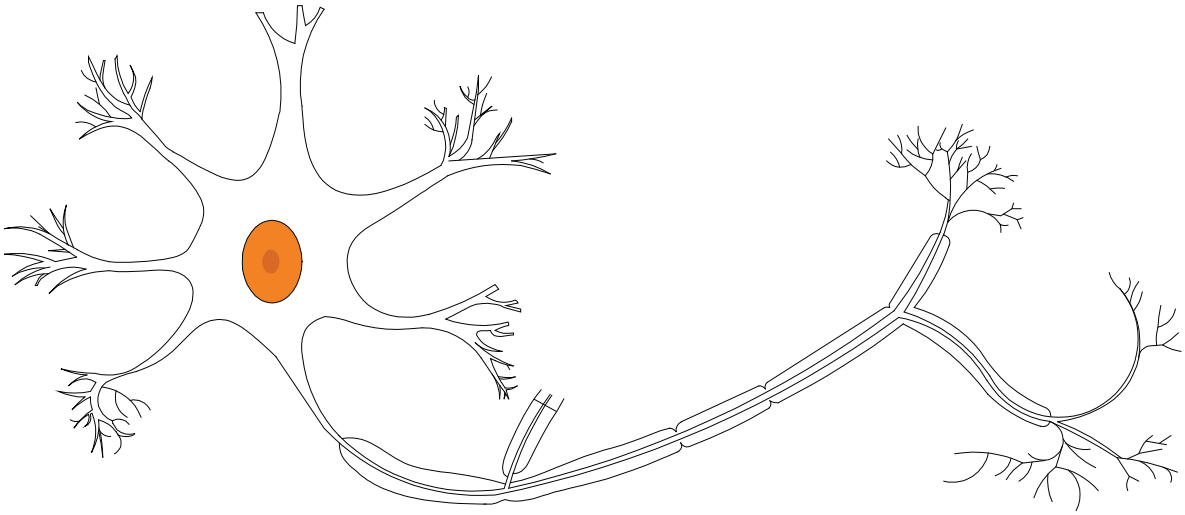


Figure 1.1: Structure of neurons

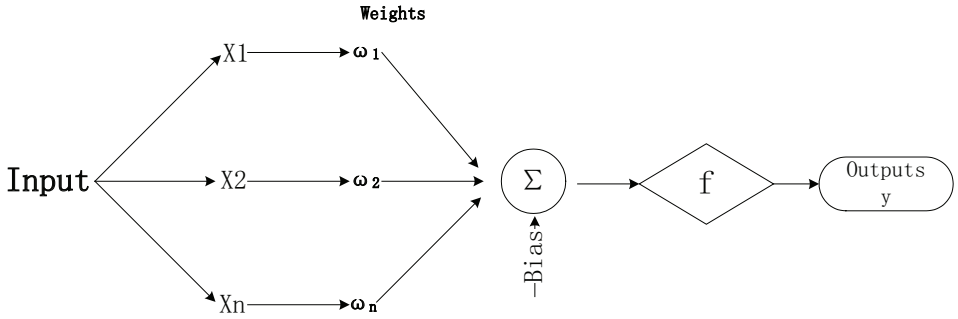


Figure 1.2: Structure of artificial neural networks

by McCulloc and suggest that they launched in 1943 based Pitts calculation model in mathematics and neural network algorithm. The Fig.1.2 shows the structure of the artificial neural network.

This model provides two different approaches to neural network research. Focus on the biological processes of the brain, another focus on the field of application.

ANN is based on the biological neural network of the operation of the biological neural system simulation. In the past few years, great progress has been made in the research of artificial neural network, and successfully solves the automation control, intelligent robot, biology, economics and practical problems in the field of pattern recognition and estimation of many modern computers. Basically, artificial neural networks have been shown to have good intelligence. In the next section, I will tell

you a number of artificial neural networks, such as MLP, ANFIS, Ellman network, single neuron model (SMN).

1.1.1 The Multi-layered Perceptron

Since 80s, a large number of artificial neural network models have been put forward. The most familiar patterns should be multi layer perception and Hopfield networks. In this section, we will focus on the MLP that has been used for many problems. Due to the ability of any input-output mapping, the MLP is well suited for prediction.

MLP is typically composed of nodes with several layers. For example, input layer, hidden layer and output layer. The input layer is used to receive the external information, and is used for the output layer to solve the problem. The input and output layers are divided between different intermediate layers, which we refer to as hidden layers. In Fig.1.3, I'll give you a complete connection to the MLP, and the model has only one hidden layer of the model.

For forecasting problems, the inputs to an ANN are usually used as the independent variables or predictor variables. The formula estimated by the ANN can be written as

$$y = f(x_1, x_2, ..., x_p) \tag{1.1}$$

Among them $x_1, X_2, ..., x_P$ is independent variable, y is the dependent variable. In this case, the model functionally equals to the nonlinear regression model. And on the other side, in the extrapolation and time series prediction, the input is usually the past observation of the data sequence, the output is predictable value. ANN performs the following functions mapping

$$y_{t+1} = f(y_p y_{t-L}, ..., y_{t-p}) \tag{1.2}$$

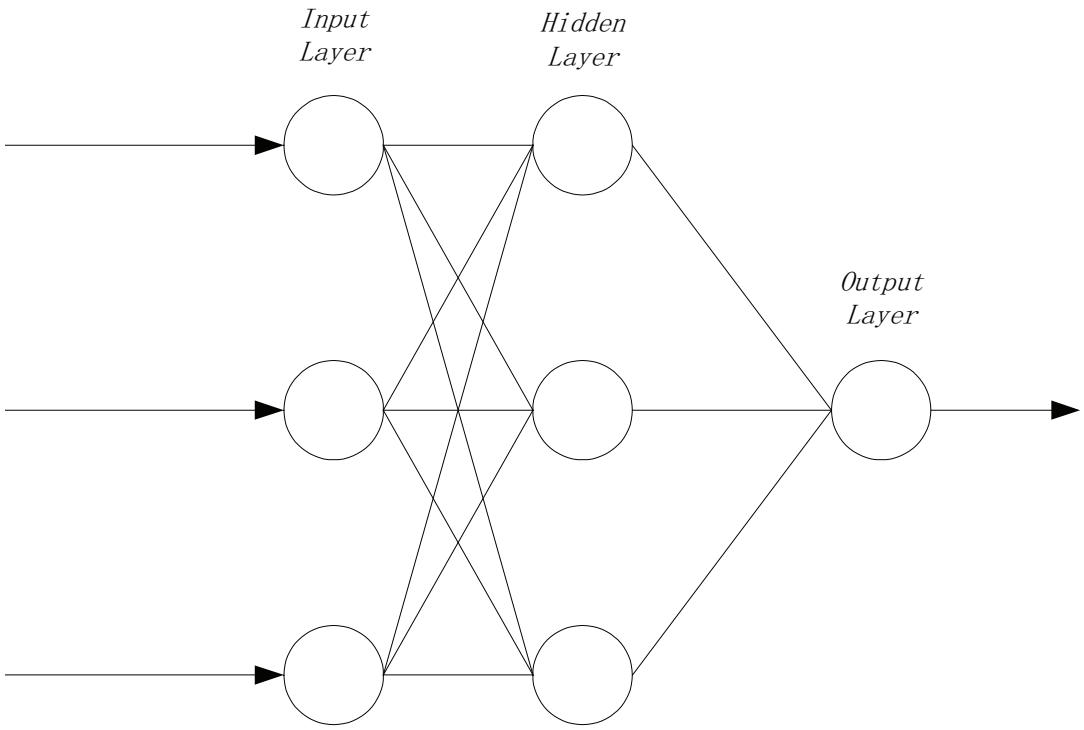


Figure 1.3: Layers in artificial neural network

Where y_t is a observed value at the time t. Therefore, ANN equals to the nonlinear auto regressive model for forecasting the time series. It is very easy to observe and time lag two predictor combined neural network model and it equals to the general transformed function models. ANN must be trained to perform any required tasks. Basically, the training is a process which decides the arc weight as the key element of ANN. The knowledge acquired by network learning is stored in the form of arc weights and nodes and arcs. It should be noted that the connection arc that a neural network may realize complex nonlinear mapping in the period which is from the input node to the output node. MLP training is a desired response (target value) in the network for each input mode (example). It is always available and supervised.

A form of a vector about input variables is training a data. Every element in the connected input vector is an input node according to the network input layer. As a result, the number of input nodes is the same as the input vector in dimensions. For the cause and effect prediction problem, the number of those independent variables

corresponding to the problem is the number of its input nodes. However, for problem of the time series prediction, it is a little difficult to decide the number of the input nodes. No matter what size, the input vector of the time series prediction problem will nearly include all the fixed length of the moving window with the series. The whole number of the available data is often organized into training set and test set (outside the sample or retention sample). Generally, the training set of an ANN is for estimating the arc weights, while the remaining test set is used for measuring the ability of the ANN in generating.

The process of training is typically organized as in the following. Firstly, a sample set about to be is input to input node. The activation value of input node is weighted and put together in every single one node in the hidden layer 1. Then the number in total is converted to the node's activation value by the activation function. If the output activation value is appeared, it becomes the input of the nodes in the next layer. The training algorithm is used to find the weight of the global error measurement of the minimization and (SSE) or mean square error (MSE), for example, the square error. For the time series prediction problem, the training mode is composed of a series of fixed number of hysteresis observations. Assuming that we have N observations in the training set $y_1, y_2, ..., y_N$, and we need a step forward prediction, we have the use of ANN and it has N input nodes of the N-n training model in total. The first model about being training is combined with $y_1, y_2, ..., y_n$, and put as the target output. The first training mode is composed of YN + 1. The next training mode contains $y_2, y_3, ..., y_{n+1}$ as input, y_{n+2} is output. In the last, the final training mode is the target of the input $y_{N-n}, y_{N-n+1}, ..., y_{N-1}, y_N$. Typically, a SSE combined with objective function or cost function is optimized to minimum during the training process.

$$E = \frac{1}{2} \sum_{i=n+1}^{N} (y_i - a_i)^2 \tag{1.3}$$

where a_i is the actual output belonging to the network . $\frac{1}{2}$ is used for simplifying the expression of derivatives evaluated in the training algorithm.

1.1.2 The Artificial Neuro-fuzzy Inference System

ANFIS is a multi-layer feed forward neural network. It takes neural network learning algorithm and fuzzy inference to map the input space to output space. ANFIS uses the ability to combine the linguistic capabilities of a fuzzy system combined with neural network adaptive network and it has been proved that it is robust in simulation of multiple processes.

ANFIS has extraordinary learning, construction, cost and classification ability. It is equipped with the advantage that it is allowed to extract fuzzy rules from digital data or expert knowledge and to establish the rule base adaptively. In addition, it can simulate the human intelligence to the complex fuzzy conversion system. The defect of the ANFIS model is that it takes a lot of time to train the structure and to determine the time required for the parameter.

We suppose that there are two inputs in the fuzzy inference system. They are X and Y as well as the output Z. Equations of the IF-THEN rules can been defined as follows.

$$Rule1 : \text{If } x \text{ is } A_1 \text{ and } y \text{ is } B_1 \text{ then} z_1 = p_1 * x + q_1 * y + r_1 \tag{1.4}$$

$$Rule2 : \text{If } x \text{ is } A_2 \text{ and } y \text{ is } B_2 \text{ then} z_2 = p_2 * x + q_2 * y + r_2 \tag{1.5}$$

Among them, $p_i, q_i and r_i (i = 1 or 2)$ are the linear parameters of the Sugeno fuzzy model in the last part. The construct of the ANFIS combined with five layers

(Fig.1.4). A brief introduction to the model can be seen as follows.

$$O_{1,i} = \mu_{A_i}(x) \quad \text{for} i = 1, 2 \tag{1.6}$$

$$O_{1,i} = \mu_{B_{i-2}}(y) \quad \text{for} i = 3, 4 \tag{1.7}$$

Among them, y, i is the explicit input of node $A_i, and x, B_i$ (small, large and so on) are respectively by the appropriate membership functions. μA_i and μB_i are the characterized by linguistic labels. Gauss and bell shaped membership function is more and more used to specify the fuzzy set, due to the smooth and compact symbol. The bell shaped membership function than Gauss membership function of a parameter, therefore in adjusting the free parameters is close to non fuzzy set. In this study, the use of bell shaped membership function.

$$\mu_{A_i} = \frac{1}{1 + |fracx - c_i a_i|^{2b_i}} \tag{1.8}$$

$$\mu_{B_{i-2}} = \frac{1}{1 + |fracy - c_i a_i|^{2b_i}} \tag{1.9}$$

Among them, a_i, b_i, c_i is the parameter set. The set is the membership function of the fuzzy if-then rule which is changing the shape of the membership function. The parameters are called pre conditions.

The second layer: rule node. The AND operator is applied to obtain the result of the antecedent of the rule, that is, the output of the emission intensity. The emission intensity is the first part of the fuzzy rule to satisfy the degree, and the output function of the fuzzy rule is plastic. As a result, the output of the layer is O_2, and the k is the product of the corresponding degree from layer 1.

$$O_{2,k} = w_k = \mu_{A_i}(x) * \mu_{B_j}(y), \quad k = 1, ..., 4; i = 1, 2; j = 1, 2 \tag{1.10}$$

The third layer: average node. The main objective is to calculate the sum of the emission intensities of each i rule and the emission intensity of all the rules. Therefore, as the standard of the firing strength.

$$O_{3,j} = \bar{w}_i = \frac{w_i}{\sum_{k=1}^{4} w_k}, \quad i = 1, ..., 4 \tag{1.11}$$

The fourth layer: subsequent nodes. The fourth level node function calculates the contribution of each of the first i rules to the total output and the defined function

$$O_{4,j} = \bar{w}_i f_i = \bar{w}_i(p_i x + q_i y + r_i), \quad i = 1, ..., 4 \tag{1.12}$$

Where i is the output comes from the former layer. For p_i, q_i, r_i, they are parameters set in subsequent sections of the Sugeno fuzzy model.

Fifth layer: output node. A single node calculates all of the output under taking all the input signals in a sum. As a result, the process of the solution of the model is to convert the fuzzy result of each rule into the clear output in this layer.

$$O_{5,1} = \sum_{i=1}^{4} \bar{w}_i f_i = \frac{\sum_{i-1}^{4} w_i f_i}{\sum_{i-1}^{4} w_i} \tag{1.13}$$

Network based supervised learning. Therefore, our aim is to train the adaptive network to approximate the function provided by the training data. Finding the exact value of the parameters in next step.

The distinguishing feature of this method is useing the hybrid learning algorithm. ANFIS uses it which calls the gradient descent method to update the parameters. The method is used to adjust nonlinear parameters (a_i, b_i, c_i). And in the least squares method it is used to identify the subsequent linear parameters $p_i, q_i and r_i$. As shown

in figure 1. As shown in Figure 1, a circular node is not a fixed (i.e. non adaptive) node with a parameter variable, and the square node has a parameter variable (which changes the parameters during training). The request of learning process is divided into two steps: in step one, the least squares method is used to identify the parameters of the consequences. And the premise parameters (membership function) is assumed under the fix of the current cycle through the training set. After that, the error signal is transmitted back. The gradient descent method can update the prerequisite parameters by optimizing the total two cost function.

The distinguishing feature of this method is that the ANFIS is updated with hybrid learning algorithm, gradient descent method and least square method. The gradient descent method is for adjusting the premise nonlinear parameters (a_i, b_i, c_i). In the mean time, least squares method is used to identify the subsequent linear parameters $p_i, q_i and r_i$. As shown in figure 1. As shown in Figure 1, the circular node is a fixed (i.e., non adaptive) node with no parameter variables. While the square node has parameter variables (during training to change the parameters). The target of learning process is divided into two phases: in the first phase, the least squares method to identify the consequence parameters, and the premise parameters (membership function) hypothesis through the training set of the current cycle is fixed. Then, the error signal is propagated backwards. The gradient descent method is prepared for updating the premise parameters by minimizing the total two function, and the parameters are fixed.

1.1.3 The Elman Network

There are two main types of neural networks, feedforward and recurrent neural networks, based on the category of a structure viewpoint. The core model of the algorithm is the Elman neural network model, one of the recurrent neural network. Elman network first introduced in the work of Elman (Elman, 1990). Elman network

model has faster convergence speed, high accuracy and good generalization ability. Fig.1.5 describes the original Elman neural network. From this figure, we can clearly find that Elman net is an ANN with three layers of neurons. The first input layer is composed of two different groups of neurons. One group includes external input neurons, while the other group contains internal input neuron. Each kind of neuron plays different role in the information processing procedure. In addition, the external neurons are also known as the context unit. To be more specific, the input to the context unit is the output of the hidden neuron that forms the second layer or the hidden layer. On the other hand, the output of the context unit and the external input neuron is fed to the hidden neuron. The context units are also called memory cells because they store the previous output of the hidden neurons, thus makes the Elamn neural network has more capacity of using associated memories to deal with complex problems.

The training function of the network uses the Levenberg-Marquardt optimization to update the weight and the deviation value. Adaptive learning function is a gradient descent with momentum weight and deviation learning function. Performance function to measure the performance of the network based on the mean of the squared error. The transfer function of each neuron is the traditional Tan-Sigmoid transfer function.

Generally speaking, in the feedforward network unit and a learning algorithm using hidden, hidden within the unit input mode, the input to enable a network to acquire the output in a way to those encoding mode. In the Elman construct, the context unit remembers the former internal state. Therefore, the hidden unit needs to map the external input and the former internal state to some desired output. As the mode in hidden unit is a mode that is saved in the context, hidden units should implement this mapping, and also generate a representation of the useful encoding of the properties

of the serial input.

Therefore, the internal representation is sensitive to time context. And the influence of time is implicit in these states. Note, however, that the time context of these representations does not have to be literal. They represent a high level of task and stimulus dependent memory. Theoretically, the original Elman neural network with all the feedback connections from hidden layer to neuron can be represented by any n order system, where n is the number of the context units.

1.1.4 The Single Multiplicative Neuron Model

The model of artificial single multiplication neuron was proposed by Yadav et al. (2007). By using the statistical learning theory of Vapnik (Vapnik, 1998), the nonlinear generalization, noise tolerance of the model and the input output mapping have been studied. There are more details is in et al Yadav. (2007). The single neuron model is sufficient for the application of conventional neural networks with multiple neurons in different layers.

SMN model can be used to solve engineering problems, demonstrating its advantages compared with other neural network models with simpler structure and lower computational complexity. In addition, the nonlinear filter can deal with the additional noise, and can update the model parameters in the new observation data due to the structure of the iterative algorithm. The model is shown as follows. A single doubling neuron with learning algorithm is illustrated in Fig.1.6.

Among them $(x_1, X_2, ..., x_n)$ is the input mode. (w_1, w_2, w_n) and $(b_1, b_2, ..., b_n)$ are the weights and biases of the model. Operator ω such as multiplication in Eq.1.14 u is equal to ω.

$$\omega = \prod_{i=1}^{n} (w_i x_i + b_i) \tag{1.14}$$

The output function is the logsig function defined as Eq.1.15.

$$y = \frac{1}{1 + e^{-u}} \tag{1.15}$$

The error between the output of the model and actual values can be minimized by the learning algorithm.

1.1.5 Back Propagation Algorithm

In this study, the Elman network learning has been implemented using the reverse propagation (BP) algorithm. The BP algorithm is a widely used algorithm in ANN learning, which is used to modify the weights and thresholds of the connections among neurons. It thus can map nonlinear processes. This well-known BP algorithm is generally a systematic method, which is usually used for training multilayer neural networks, and has shown many advantages as listed in the following. First of all, BP has a strong mathematical foundation based on gradient descent learning. In addition, it is the most widely used algorithm in the traditional neural network training. In addition, BP algorithm also can be used in a new generation of neural network model, for example, you can use the BP learning algorithm to train the Elman network. Finally, the implementation of BP algorithm is very easy and can be realized in hardware directly. Nevertheless, there are also many evidences that explain why BP algorithm is inefficient in training most ANNs, especially due to its inherent local minima trapping problems.

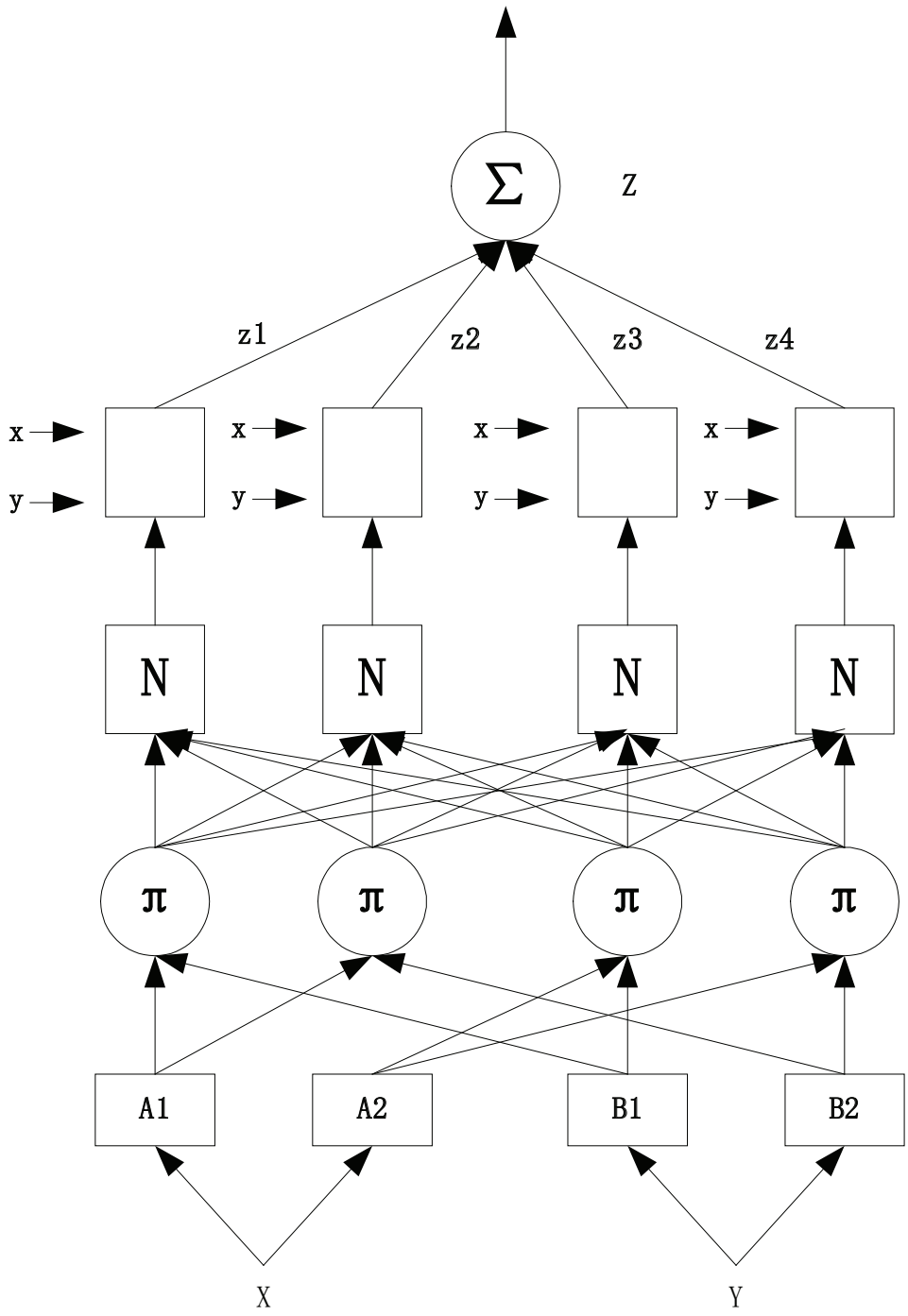


Figure 1.4: Architecture of the ANFIS

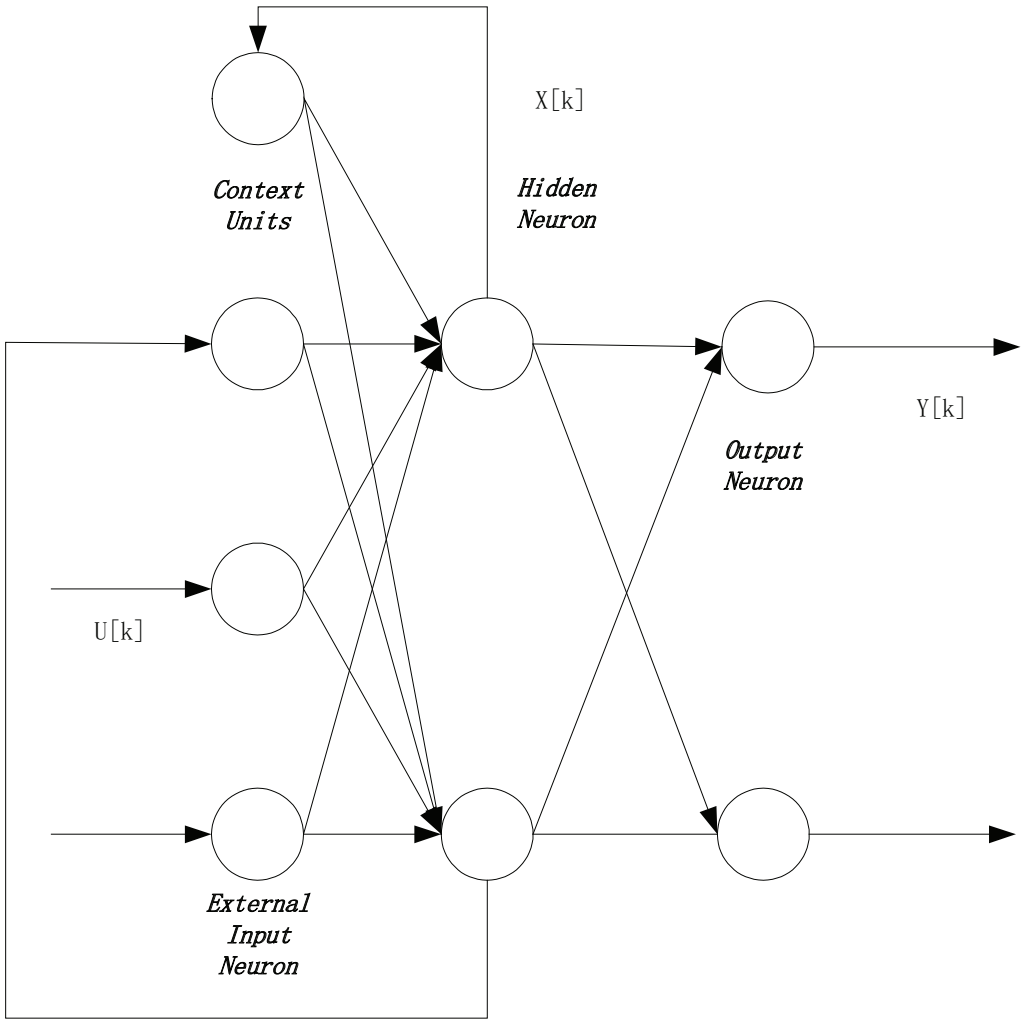


Figure 1.5: The Elman Network

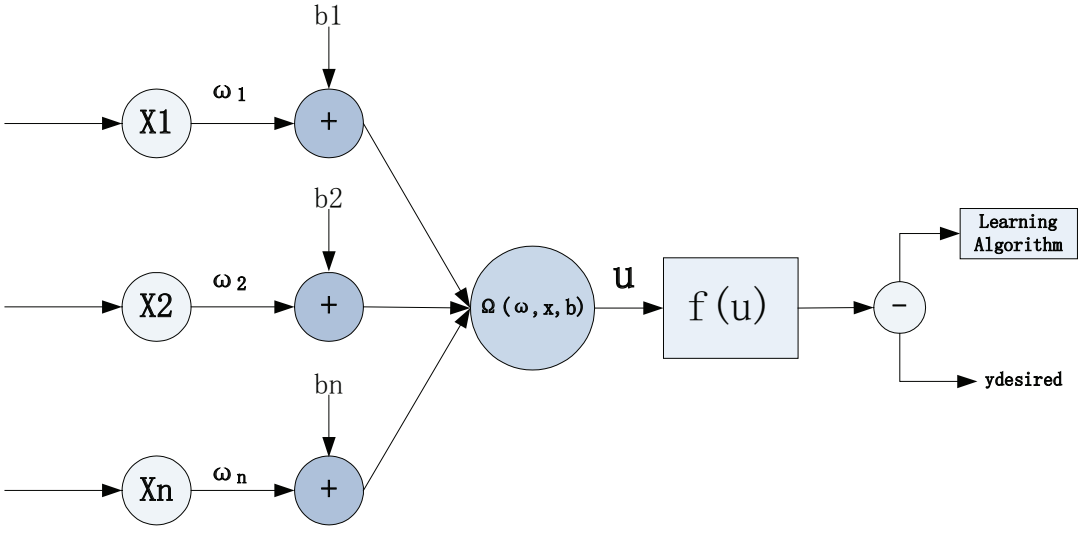


Figure 1.6: Generalized single doubling neuron with learning algorithm

Chapter 2

Evolutionary Computation

The computational model of evolutionary computation is used as a key element in soling problems. The existence of a lot of evolutionary computation models has been proposed. We call it an evolutionary algorithm. They have a common conceptual basis to simulate the evolution of nature animals through the process of selection and regeneration. The first evolutionary algorithms can be traced back to 1950s years (e.g., Fraser, 1957; Box, 1957). For simplicity, we won't focus on this earlier work. We will discuss in detail the three methodologies that have been proposed in the past years: "evolutionary programming" (et al Fogel. (1966), "evolutionary strategy" (Rechenberg, 1973) and "genetic algorithm" (Holland, 1975). Although each of these variants is implemented in a different way. Many applications have been used by evolutionary algorithms. Such as data mining, pattern recognition and complex functions. In this context, many features are added to the EA and it can enhance the quality of the solutions. In this chapter, we provide an overview of EA, and discuss the field that EA is commonly used in.

2.0.6 Genetic Algorithms

By Holland (1975) genetic algorithm (GA) on the development of the traditional use of more domain independent said on the throne. However, many of GA's recent applications focused on the other said, such as graph (neural network), Lisp expression

vector and an ordered list. It outlines a typical genetic algorithm (GA). After initialization, the father is selected under the probability function by the calculated fitness. In other words, these individuals with high calculated fitness are more likely to be chose as parents. The child mutates and compares with the parent, if it is better, then replaces the parent in the population. It is remarkable that the operations of mutation and crossover is opposite to that of EP. In the GA , a number of small probability turns over bits, and it can be considered as the background operator. In other words, the reorganization is underlined as the search agent which has important influence. GA is the optimizer, although its universal adaptability may have more importance for some researchers (De Jong, 1992).

2.0.7 The Particle Swarm Optimization

PSO belongs to stochastic optimization algorithms. The idea of building PSO is not inspired by the evolutionary mechanisms that are encountered in verbal choice but gets the inspiration of the social behavior of groups of creatures such as birds and fish. It has been observed that the behavior of the individual to constitute a group of basic rules, such as the nearest neighbor velocity matching and the distance acceleration. In this regard, PSO has claimed that the conscience of the mutation. The initial purpose is to use the graphical simulation of a flock of elegant but unpredictable movement.

PSO optimizes based on the population of the algorithm, which uses individual groups to search the promising area. By this case, the population is called warm, and the individual is called particle. Each particle moves with a individual speed in the search space, and maintains its best position in its memory. In the global variant of PSO, the best position for all individuals in the population is transmitted to all particles. In a local variant, each particle is assigned to a neighborhood consisting of a predetermined number of particles. In this case, the best position of the particles

that form the neighborhood is transferred between them.

PSO is a algorithm which uses the evolutionary techniques just like the genetic algorithms. In PSO, each potential solution is treated as a particle with its own speed in the search space. PSO is widely used in feature selection because particle swarm will find the best feature combination when they are flying in the search space. GA requires less operators in the operation of crossover and mutation than PSO. It only needs the original and simplified operators. Also it has lower computational time. The proposed algorithm has been compared with GA and other algorithms by the experiments of UCI data.

PSO is a new natural inspired optimization algorithm implemented evolutionary computation technique proposed by Kennedy and Kennedy (Eberhart and Eberhart, 1995a and Kennedy, and Eberhart, 1995b). The concept of particle swarm is simulated from the social behavior of biological organisms. The first is to imitate birds flocking elegant but unpredictable movement. PSO algorithm to simulate the behavior of birds and the exchange of information means to solve the optimization problem. Each possible solution is assumed as a particle with its own velocity and is "flying" around in the problem space". Each particle will update its velocity according to its own flight experience and other particles' flight experience. The best region of complex search space is found by the interaction of the individual in the particle swarm. PSO has been proved that it can successfully solve a mountain of optimization problems.

PSO has some random solutions. The solutons are known as the "particle" initialization. Each particle is assumed as a point in a search space with S dimensions. The first I particle is $Xi = (Xi_1, Xi_2, ..., xi_S)$. The best position obtained so far of any particle (best p, given the location of the optimal fitness value), and expressed as $Pi = (Pi_1, Pi_2, ..., Pi_S)$. The index of the best particle in the population of all

particles is called the global best so far "gbest". The velocity of each particle to change its position is expressed as $Vi = (Vi_1, Vi_2, ..., Vi_S)$ like the following equation operator:

$$v_{kl} = w * v_{id} + c_1 * rand * (p_{kl} - x_{id}) + c_2 * rand * (p_{gd} - x_{id}) \tag{2.1}$$

$$x_{id} = x_{id} + v_{id} \tag{2.2}$$

Where $d = 1, 2, ..., S, w$ is the inertia weight, which is based on the time variation of the generation of the positive linear function. Selecting the inertia weight properly provides a balance to switch between exploration and exploitation, and leads to an average less iteration to find the optima. C1 and C2 in Eq.2.1 is the acceleration constant. They represent the adjustment of the weight in the pbest and gbest positions of each particle. A low value allows the particles to wander off the target area before being dragged. The high values result in a quick movement toward the target area. Rand is the range of [0, 1] in the two random functions.

Maximum speed of Max V is the limitation velocity of each particle. It limits the size of each particle allowed to use the solution space. If Max V is too small, the particle could not have a chance to fully explored beyond the local good region. And it may lead to a result that particles trapped in local optima. But in reverse, the Vmax is too high, the particles may fly over a good solution.

The Eq.2.1 provides a certain degree of memory for the "flying particles". It allows the particle to explore wider space. It is the "cognition" in the second part of the equation, which is the private thoughts of each particle. The "society" is the cooperation between the particles. Eq.2.2 is used to update the velocity of each particle by the distance between the previous speed of the particle and its current position and its best position obtained so far. Then, the particles flew to the new

position. The performance of each particle is calculated by the predefined function.

2.0.8 Differential Evolution

Differential evolution (DE) can be said to be one of the stochastic optimization algorithm parameter arguments currently in use the most powerful. DE operates in a similar calculation step with the standard EA. However, different from the conventional DE, EA variables disrupt the current generation of population members randomly selected and different population members of the scaling differences. Therefore, it is not necessary to use a separate probability distribution to generate offspring. Many researchers around the world have paid their attention to DE. Many optimization algorithms have been proposed and their performance have been improved a lot since the inception of DE in 1995.

The new millennium has witnessed the development of information technology as a driving force behind the progress, especially in the field of design related computing. The increasing complexity of computer programs, the availability of computing speed, and the cost of continuous reduction have had a major impact on Civil Engineering, which can be considered as a paradigm change.

Until recently, the design of the structure of the computer is mainly used for the analysis purpose of the detailed design stage. Nowadays, the role of the design of the structure of computers has becoming more and more different and diverse. Generally, they are utilized in all stages of the design process, from the formation of conceptual design (design, layout or topology) through preliminary design (shape design specification), and finally to the detailed design process (structure dimension). This requires a new intellectual and computational framework to fully benefit from advances in information technology. In the computing paradigm, evolutionary computation (EC) is now considered to be particularly suited to a variety of traditional and novel computational applications in structural engineering.

The evolutionary computation uses the computational techniques based on evolutionary principles in nature. The evolutionary principles are the guidance of flora and fauna to make them suitable for living in the planet. They also use this principle to connect with each other. DE uses this concept to implement into optimization algorithm to find the solutions.

The first work to solve the problem of the use of evolutionary heuristics can be traced back to the late 1950s. By Berg and Skei Vee Phil Reagan on evolutionary strategy for independent research and almost at the same time, the research on genetic algorithms and evolutionary programming for Fogel Holland, the research and application of evolutionary technique.

Three elemental mechanisms driving the evolution of nature: reproduction, mutation, and selection. They ultimately affect chromosomes that contain individual genetic information, rather than the individual. Breeding is the process of introducing new individuals to get into groups. Recombination (or crossover) occurs during sexual reproduction, which is transferred to the progeny chromosome, which is a mixture of parental genetic information. Mutation causes small changes in the genetic chromosome; it is often due to duplication of errors during reproduction. Choice is a process of guiding the survival of the one which is suitable for the environment by Darwin's survival of the fittest. The most suitable for their environment, so the survival and reproduction.

In general, evolutionary techniques have been widely considered as the search mechanism or optimization techniques. As Michalewicz writes: "any abstract tasks to be completed can be thought of as a solution to a problem, which can be viewed as a search space for a potential solution. Since we are usually in the "best" solution, we can view this task as an optimization process."

From the beginning, we randomly generate a vector to create the initial population,

which will be modified throughout the execution of the algorithm. In fact, in each generation, we deal with the current group represented by the PC, which has been found to be acceptable as the initial point, or by comparison with other vectors:

$$P_{C,i}^{g} = (C_i^g), \text{with} i = 0, 1, ..., N_p - 1 \tag{2.3}$$

$$C_i^g = (c_{j,i}^g), \quad j = 0, 1, ..., D - 1 \tag{2.4}$$

$$g = \text{index of current generation} \tag{2.5}$$

Index g and I respectively represent the generation and population of the vector. i, [0, Np-1], Np is the population size. In addition, J represents the index of each element in the solution vector containing the D elements. The main feature of this evolutionary algorithm, namely DE, is the exploration of feasible space. This is mainly through the wisdom of the group size Np and the appropriate choice to ensure that. In other words, it should not be too small to avoid stagnation and provide adequate exploration. Np increased the number of induced functional evaluation; that is, the convergence rate of the algorithm has a negative impact.

2.0.9 Mutation

Similar to genetic algorithms, DE begins with two parents and creates a child. Then we randomly sample three vectors to re combine them. Eq.2.6 shows how to combine three different vectors to produce a mutant vector:

$$M_{j,i}^{g} = C_{j,0}^{g} + A * rand_{j,i}^{g}(C_{j,i}^{g} - C_{j,2}^{g}) \tag{2.6}$$

The scaling factor A is positive control group evolutionary rate selection. Although there is no upper limit on the A, the RMS value is less than "1", and is a random number between 0 and 1.

2.0.10 Crossover

Once generated, DE is associated with vectors from the current population of cross mutant vectors. Results are obtained by using the following procedure:

$$T_{j,i}^{g} = \begin{cases} M_{j,i}^{g} & \text{if}(r_{j,i}^{g} \leq Cr \text{ or } j = j_r) \\ C_{j,i}^{g} & \text{otherwise} \end{cases} \tag{2.7}$$

C_r and [0,1] cross cross factor values of different elements based on vector. By comparing the output of the factor and the uniform random number generator, we determine the source of each element of the test vector. If the random number is less than or equal to C_r, then the test parameters are inherited from the mutant; otherwise, the parameters are copied from the current vector.

2.0.11 Selection

In the selection step of the algorithm, the test vectors are compared with the corresponding target vectors supposed that they are connected with the objective function values of the test vectors. The fitness of two individuals is generated in the next generation Eq2.8.

$$C_{i}^{g+1} = \begin{cases} M_{i}^{g} & \text{if } f(M_{i}^{g}) \leq f(C_{i}^{g}) \\ C_{i}^{g} & \text{otherwise} \end{cases} \tag{2.8}$$

Once the current population is updated, it is through mutation, crossover and selection to evolve again until the optimal value is found, or a predefined termination criterion is reached.

Chapter 3

Using A Single Dendritic Neuron to Forecast Tourist Arrivals to Japan

3.1 Introduction

In the past few decades, the significant growth of international tourism has been achieved in Japan, and the tourism industry has become a crucial contribution to Japan's economic development. According to the Japanese National Tourism Organization [1], there are nearly 1.85 million tourists will visit Japan in January 2016, and it will record the highest figure for January on a monthly basis [2]. Chinese tourists are going wild on a shopping spree in Japan, resulting a new word "Bakugai" in Japanese. It is highly important for Japanese tourism agencies including government bodies and the private sector to understand the trends affecting monthly tourist arrivals. Thus, the visitors planning a tourism to Japan may need to make more detailed plan to avoid fastigium.

Linear parametric time series are usually used to construct forecasting models which are applied by the traditional tourism demand researches. The most famous are the autoregressive integrated moving average models [3–5], the naive method [6,7], and the exponential smoothing model [8]. However, the predictions made by these models are not precise enough to reach people's expectation. Using these models to

simulate the tourism time series is also not a easy work [9, 10].

Recently, more and more nonlinear forecasting models are proposed to address the above issues in the time series prediction. A piecewise linear method is proposed to model and forecast the demand for the tourism, and the experimental results indicate that the piecewise linear model is significantly more accurate than those autoregressive models [11]. A regime switching detection and forecasting model is proposed in [12]. However, the performance of these models is limited to the problems of proper model selection and data dependency [10, 13].

On the other hand, machine learning techniques are developed for time series forecasting, such as support vector machines [14–16], fuzzy time-series methods [17], rough set approaches [18, 19], genetic programming [20], artificial neural networks (ANNs) [21–28] and their hybridizations [29–32]. These complex non-linear models overcome the limitation of linear models as they are able to capture non-linear pattern of data, thus improving their prediction performance.

Among them, ANNs are receiving increasing interests due to their ability to adapt to imperfect data, functions of self-organizing, self-study, data-driven, associated memory, and arbiter function mapping [9]. ANNs can learn from patterns and capture hidden functional relationships in a given data even if the functional relationships are not known or difficult to identify [33, 34]. Using the training methods, an ANN can be trained to identify the underlying correlation based on the inputs and outputs, and finally to generate appropriate outputs. A number of researchers have utilized ANNs to predict tourism demand [24, 25, 27, 31, 35, 36]. Kon and Turner [24] announced a research about using ANN to predict in tourism. Empirical evidences show that ANNs is better than the classical models in the performance of forecasting the tourism. For example, the best performance was obtained by an ANN method in [22] when compared it with the traditional models.

Although various ANNs have been proposed for tourism time series, it couldn't bahave a excellent performance when comparing with other algorithms in all problems [37] because each ANN has many properties and limitations which deteriorates prediction results. For example, multiple-layered perceptron (MLP) has a good capacity in applications, the back-propagation-based MLP has drawback in patterns which have time independence. [38] proves that the time-delayed ANN can be suitable for mapping the past and present values. But ANNs still has some limitations after initialization. [39]. The Elman recurrent ANN [40] has a better performance comparing with the MLP as it has more time dependencies in data. However, the conventional ANN algorithms based on the gradient descent approach have many drawbacks such as slow convergence speed and time costs [41]. These drawbacks make it difficult to be applied into practical problems.

In this paper, we propose a realistic single dendritic neuron model (SDNM) with synaptic nonlinearities in a dendritic tree for tourism forecasting. The distinct characteristic of SDNM is that the sense of locality of dendrites can be represented and manipulated. For a specific given task, SDNM is able to detect the synapse which are needed or not. [42, 43]. This is realized by modeling the synaptic nonlinearity implemented with a sigmoid function. Thus enabling single neuron to have the ability of computing functions and approximating any complex continuous function [44, 45]. On the other hand, although the tourist arrivals time series apparently is one-dimension, it actually contains high-dimensional information and is a result of many factors such as the tourism policy which strongly influences the number of tourists, and thus making the tourism data nonlinear, irregular, and difficult to be predicted. To address this problem, we employ the phase space reconstruction (PSR) technique based on the Takens's embedding theorem [46] to handle the chaotic properties of the tourism time series before using SDNM to perform the prediction. By doing so, single obser-

vations from the tourist arrivals can be transformed into a set of multiple dimensional vectors with two parameters of time delay and embedding dimension. The acceptable dimensions and time delay of the attractors in the tourism time series can be obtained, and the data can obtain dynamic behavior and structural topology. According to the Lyapunov exponent of the reconstructed phase points of tourism time series, SDNM is then used to perform short-term predications. Experimental results of the forecasting of the monthly foreign tourist arrivals to Japan indicate that the proposed SDNM is more efficient and accurate than other neural networks including the MLP, the artificial neuro-fuzzy inference system (ANFIS), the Elman network (Elman), and the single multiplicative neuron model (SMN).

The rest of the paper is organized as follows. Section 2 describes the SDNM in details. Section 3 elaborates more about the prediction method by using PSR and SDNM. Experimental results and discussions are organized in Section 4. Finally, concluding remarks are presented in Section 5.

3.2 Single Dendritic Neuron Model

Compared with ANNs which utilize more than one neurons in information processing procedure, many attentions have been paid to propose single neuron models, such as the single multiplicative neuron model [47, 48] and the sigma-pi unit [49]. However, these single neuron based models are based on the architecture of the McCulloch-Pitts neuron which uses weights to represent the degree of clustering in different synapses. Therefore, all sense of locality in dendrites is lost, and models may loss the ability to present local interaction within a fixed dendritic tree. Moreover, the nonlinear computational capabilities of these McCulloch-Pitts based single neuron models are limited to solve complex problems, especially the non-linearly separated problems [50].

Different from the McCulloch-Pitts neuron based models which do not consider the dendritic structure in the neuron, it has been recently conjectured by a series of theoretical studies that individual neurons could act more powerfully as computational units by considering synaptic nonlinearities in a dendritic tree [51,52]. The various types of synaptic plasticity and nonlinearity mechanisms allow synapses to play a more important role in computations [53]. Synaptic inputs from different neuronal sources can be distributed spatially on the dendritic tree and the plasticity in neuron can result from changing in synaptic strength or connectivity, and the excitability of the neurons themselves [54]. Moreover, a slight morphological difference can just cause great functional variation, acting as filters to determine what signals a single neuron receives and then how these signals are organized into a interger [55].

By taking the nonlinearity of synapses into consideration, a single dendritic neuron model (SDNM) has been proposed in our previous researches [42,44,45]. In [42], an unsupervised learning method was proposed for SDNM to learn two-dimensional eight-directionally selective problems. In [44], an error back-propagation (BP) method was used for training SDNM to perform cancer classification tasks. In [45], we demonstrated that SDNM could be approximately realized by using logic NOT, AND and OR operations. it is just the same as dendritic morphology, and thus was suitable for a simple hardware application. In this study, we apply SDNM to perform the tourism arrivals forecasting. The details of SDNM are described in the following and its architecture is shown in Fig. 3.1.

SDNM is constituted by four layers including a synaptic layer which performs sigmoid functions, a dendrite layer which acts as a multiplicative function for the outputs of synapses, a membrane layer which actually is an addition function for the outputs of all dendritic branches, and a soma function which uses another sigmoid function to output the result of the entire single neuron.

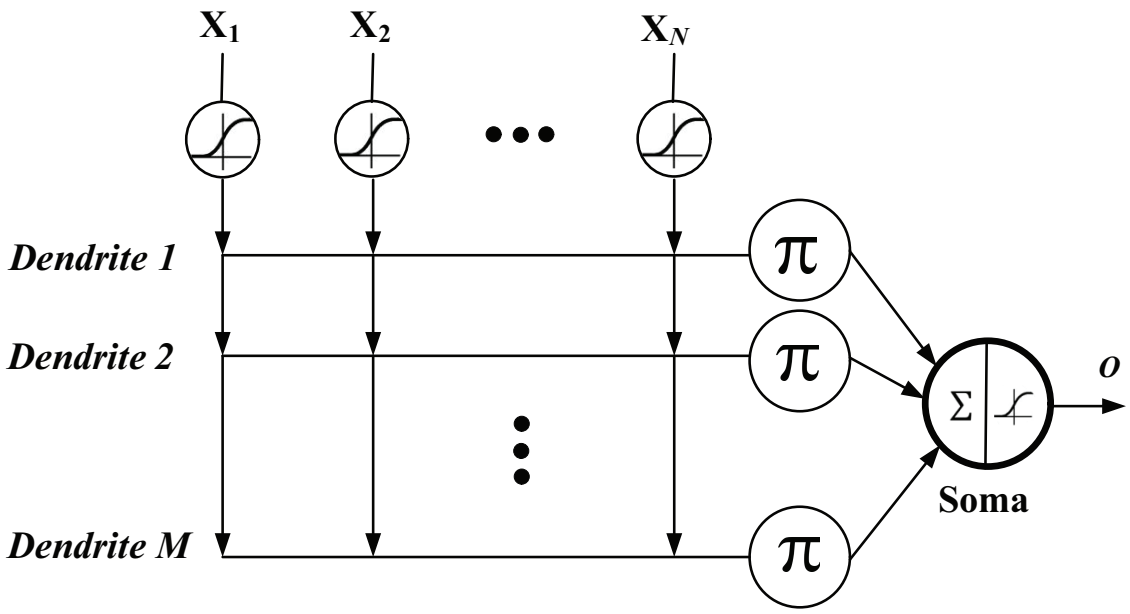


Figure 3.1: The architecture of the single dendritic neuron model (SDNM).

3.2.1 Synaptic Layer

A synapse refers to the connection between neurons at a terminal bouton of a dendrite to another dendrite/axon or the soma of another neural cell. The direction of information flow is feedforward, from the presynaptic neuron to postsynaptic neuron. The synapse can be either excitatory or inhibitory which depends on changes in the postsynaptic potential caused by ionotropic. The function connecting the i-th ($i = 1, 2, ..., N$) synaptic input to the j-th ($j = 1, 2, ..., M$) synaptic layer is expressed by Eq. (3.1). The value k is a positive constant, the weight w_{ij} and the threshold θ_{ij} are the connection parameters.

$$Y_{ij} = \frac{1}{1 + e^{-k(w_{ij}x_i - \theta_{ij})}} \tag{3.1}$$

The sigmoid function becomes a step function when the initial value of k is too large. Depending on the values of w_{ij} and θ_{ij}, there are four connection examples: (1) A constant 0 connection (when $w_{ij} < 0 < \theta_{ij}$ or $0 < w_{ij} < \theta_{ij}$). If the input is changed from 0 to 1, the output is 0. (2) A constant 1 connection (when $\theta_{ij} < w_{ij} < 0$ or

$\theta_{ij} < 0 < w_{ij}$). If the input is changed from 0 to 1, the output will become 0. (3) Excitatory connection (when $0 < \theta_{ij} < w_{ij}$). If the input is changed from 0 to 1, the output equals the input and the synapse will be an excitatory type. (4) Inhibitory connection (when $w_{ij} < \theta_{ij} < 0$) where the synapse will be an inhibitory type and the output will reverse input in this case.

3.2.2 Dendrite Layer

The dendrite layer is the nodes between each branch. It should be noted that a soft-minimization operator was utilized in our previous dendritic neuron model [42] to deal with binary input classification problem, while the multiplicative operation adopted in this study can address real number input problems. The multiplication is just the same as the logic AND operation whenever the value of inputs and outputs of the dendrites are 1 or 0. The output equation for the j-th branch can be given as follows.

$$Z_j = \prod_{i=1}^{N} Y_{ij} \tag{3.2}$$

3.2.3 Membrane Function

The branch results will be summed by an operation, which is similar to a logic OR operation in the binary case. The output is approximated as follows.

$$V = \sum_{j=1}^{M} Z_j \tag{3.3}$$

3.2.4 Soma Function

The results of the output can be calculated by follows.

$$O = \frac{1}{1 + e^{-k_{soma}(V - \theta_{soma})}} \tag{3.4}$$

The parameter k_{soma} is set as a positive constant and the threshold θ_{soma} is variable from 0 to 1.

3.2.5 BP-like Learning Method

The equation of error is as follows.

$$E = \frac{1}{2}(T - O)^2 \tag{3.5}$$

The w_{ij} and θ_{ij} are the connection during learning. The output vector produced by the input vector is compared to the target vector. It can lower the error between output vector and teaching signal T vector by correcting w_{ij} and θ_{ij}. The method of calculating the value of error can be shown as follows:

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}} \tag{3.6}$$

$$\Delta \theta_{ij}(t) = -\eta \frac{\partial E}{\partial \theta_{ij}}, \tag{3.7}$$

where η represents the learning constant and it is a positive constant. The updating rules for w_{ij} and θ_{ij} are defined as:

$$w_{ij} = w_{ij} + \Delta w_{ij}(t) \tag{3.8}$$

$$\theta_{ij} = \theta_{ij} + \Delta \theta_{ij}(t), \tag{3.9}$$

where t is the learning epoch. Moreover, the differentials of E with respect to w_{ij} and θ_{ij} can be calculated as follows.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial O} \cdot \frac{\partial O}{\partial V} \cdot \frac{\partial V}{\partial Z_j} \cdot \frac{\partial Z_j}{\partial Y_{ij}} \cdot \frac{\partial Y_{ij}}{\partial w_{ij}} \tag{3.10}$$

$$\frac{\partial E}{\partial \theta_{ij}} = \frac{\partial E}{\partial O} \cdot \frac{\partial O}{\partial V} \cdot \frac{\partial V}{\partial Z_j} \cdot \frac{\partial Z_j}{\partial Y_{ij}} \cdot \frac{\partial Y_{ij}}{\partial \theta_{ij}} \tag{3.11}$$

3.2.6 Remarks regarding Characteristics of SDNM

- The architecture of SDNM is similar to those of multiplicative neuron models and sigma-pi models. They are multiple-layered and signals are transferred in a feedforward manner. As a result, the functions used in these models can be reciprocated. For example, the radial basis functions using Gaussian kernels, a simplified fuzzy logic formulation and kernel-regression models are able to be represented by a variation of sigma-pi formulation [56]. Furthermore, some of them are isomorphic (e.g. the augmented two-layer neuron model 2LM is isomorphic to a traditional ANN [57]).

- Multiplication is both the simplest and one of the most widespread of all nonlinear operations in the nervous system [58]. Taking advantage of the multiplication operation which is essential and important to the information processing in a neuron [59], the computation in synapses is innovatively modelled using sigmoid functions. Depending on the values of the parameters in synapses, the output of synapses can successfully represent excitatory, inhibitory, constant 0 and constant 1 signals, which is benefit for identifying the morphology of a neuron [42].

- SDNM has been successfully applied on a number of classification problems, such as XOR [60], cancer diagnosis [44], Iris and Glass datasets [45]. On the contrary, some other dendritic neuron models are not able to solve such nonlinearly separated problems [50] (e.g., the Legenstein-Maass model [61]). More importantly, the classifier resulted from SDNM can be easily implemented in hardware [45]

using logic circuits.

3.3 Forecasting Framework for Tourist Arrivals

The framework for forecasting the tourist arrivals based on PSR and SDNM is shown in Fig. 3.2, where PSR is utilized to analyze the behavior of tourism time series based on the Takens's embedding theorem and SDNM is used to perform the predication. Following Fig. 3.2, the procedures of the forecasting method are summarized as in the following.

3.3.1 Input Time Series Data

Let x_t be the one-dimensional tourism time series at time t, $(t = 1, 2, ...)$. First of all, x_t is input and processed using a normalization method to the range of [0, 1] according to Eq. (3.12).

$$y_t = \frac{x_t - MIN(x_t)}{MAX(x_t) - MIN(x_t)} \tag{3.12}$$

where y_t is the normalized data to alleviate the problem of inconsistent measures for different time series data, and MAX (MIN) returns the maximal (minimal) value of the vector.

3.3.2 Phase Space Reconstruction

As we can assume that tourism is chaotic and unpredictable. And because of the properties, the forecasting of it is very difficult. Thus, making predictions in the phase space based on PSR is easier than using a one-dimensional time series. PSR is regarded as the basis of chaotic time series and widely used in non-linear system analysis. It is a theory for inferring geometrical and topological information related to a dynamical attractor based on observations. Takens [46] proposed the delay

coordinates method of PSR for time series analysis, and proved that PSR can unfold the time series into an m-dimensional embedding space while retaining the topology of the higher dimensional dynamic system with the chaotic attractor.

Two parameters of the time delay τ and the embedding dimension m are very important in PSR. Theoretically any value of τ can be easily found in delay time. But the appearance of the transformed attractor relays on the choice of embedding lag. And the standard to judge the value is suitable for τ or not is that the value must bear the function to separate the data in the time series as to have a smooth reconstruction of the attractor. In this study, we use an appropriate embedding dimension m and time delay τ to transform the phase space. The Grassberger-Procaccia algorithm [62] is used to determine the embedding dimension m and the mutual information function [63] is used to calculate the time delay τ. More details regarding the implementation of these two methods are interpreted in Section 4. As a result, a reconstructed phase space can be represented by a matrix $(P, T)'$ for the normalized time series y_t, $t = 1, \ldots, N$, where

$$P = \begin{pmatrix} y_1 & y_2 & \cdots & y_{N-1-\tau(m-1)} \\ y_{1+\tau} & y_{2+\tau} & \cdots & y_{N-1-\tau(m-2)} \\ & & \cdots & \\ y_{1+\tau(m-1)} & y_{2+\tau(m-1)} & \cdots & y_{N-1} \end{pmatrix} \tag{3.13}$$

$$T = (y_{2+\tau(m-1)}, y_{3+\tau(m-1)}, \ldots, y_N) \tag{3.14}$$

In the training and forecasting process of SDNM, P is used as the input data, while T is treated as the target data.

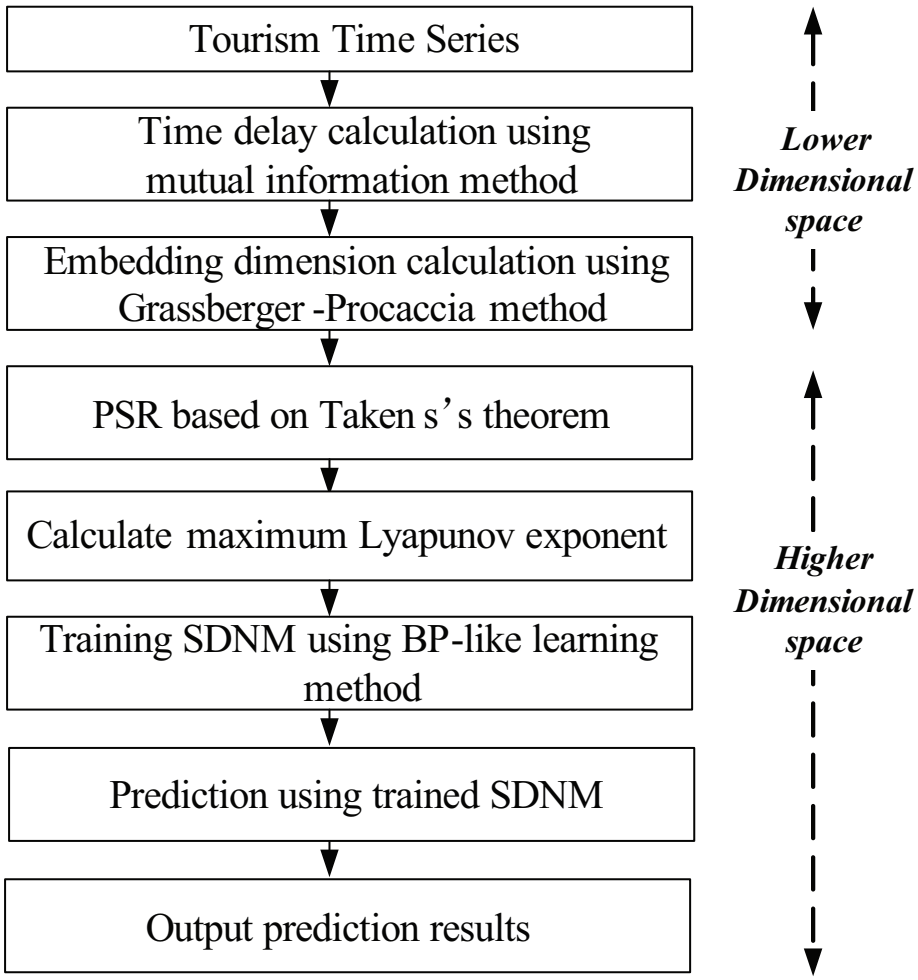


Figure 3.2: Prediction framework based on the proposed SDNM.

3.3.3 Maximum Lyapunov Exponent Calculation

To determine whether a long-term or a short term predication of the trajectory of the tourism can be made, it is necessary to calculate the Lyapunov exponents of the time series which is able to quantitatively characterize the chaotic attractor. When Lyapunov exponent is greater than 0, the time series will become chaotic [64]. The Wolf method [64] is used to calculate the maximum Lyapunov exponent under the condition of chaotic time series. t_0 is set as the initial time and the y_{t_0} is the transformed first phase points, where the minimum length compares y_{t_0} with its adjacent phase points is L_0. The distance $L_0' > \varepsilon$ has a positive variable value when

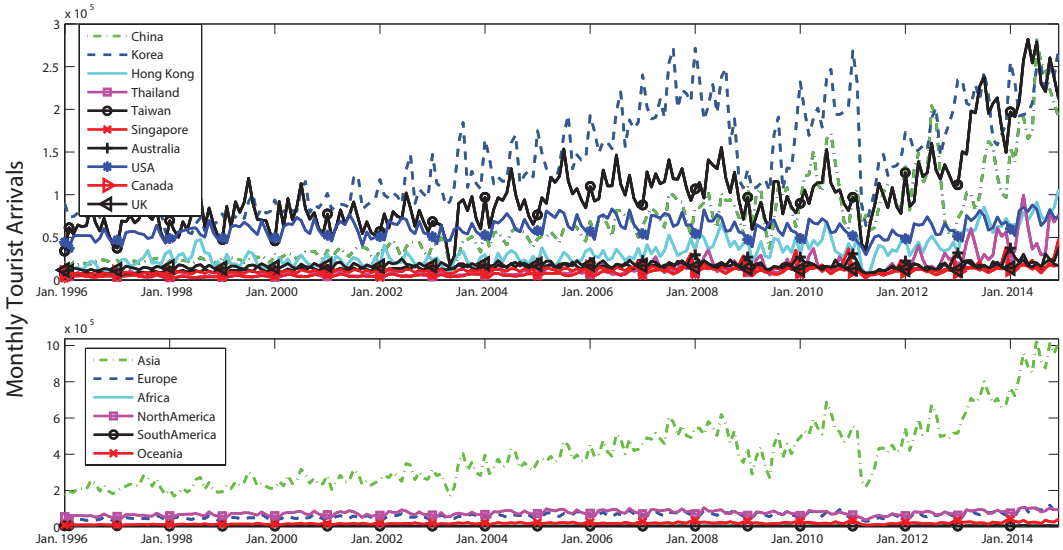


Figure 3.3: The monthly tourist arrivals from ten major source markets and six continents to Japan.

the time is t_1, $L'_0 = ||y_{t_1} - y_{t_0}||$. If another phase point $y^1_{t_1}$ with $L^1 = ||y_{t_1} - y^1_{t_1}|| < L'_0$ is found, then L'_0 will be substituted. The maximum Lyapunov exponent can be calculated as follows.

$$\lambda_{max} = \frac{1}{t_m - t_0} \sum_{i=0}^{m} ln \frac{L'_i}{L_i} \tag{3.15}$$

The prediction is allowed in a short length when the system is chaotic. But the length is unknown unless we use reciprocal Lyapunov exponent in theory [65, 66].

$$\Delta t = \frac{1}{\lambda_{max}} \tag{3.16}$$

When the maximum Lyapunov exponent is greater than 0, the system comes into a chaotic condition. And if the Lyapunov exponent keeps increasing and is greater than one, the sampling frequency will be more compared with the predictable limit. In this condition, the prediction using chaotic time series will not be reliable. So the maximum Lyapunov exponent should be greater than 0 and less than 1. The prediction will be more reliable in a long length when the Lyapunov exponent is

approaching to 0.

3.3.4 Prediction using SDNM

When the reconstruction of phase space and maximum Lyapunov exponent calculation accomplished, we carry out the prediction for tourism arrivals based on the SDNM described in Section 2. First, we divide all time series data into two parts: one is used as training data set and the other is used to verify the prediction accuracy. Then we implement the BP-like learning method to optimize the weights w_{ij} and thresholds θ_{ij} in the synaptic layer of SDNM until a learning termination condition is fulfilled. In this study, a maximum learning epoch L_{max} is used as the termination condition. Finally, we output the prediction results using some assessment methods.

3.4 Experimental Results and Analysis

We use our proposed method to study monthly foreign tourist arrivals to Japan from the eight major markets of China, Korea, Hong Kong, Thailand, Taiwan, Singapore, Australia, USA, Canada and UK, and form six continents of Asia, Europe, Africa, North America, South America, and Oceania, respectively, from January 1996 to December 2014. These data are published by Japanese National Tourism Organization [1]. Fig. 3.3 illustrates these data in one-dimensional time series. For each sequence of the tourism arrival, there are 228 points, where the first 168 (14 years) points are employed for SDNM learning and the remaining 60 (5 years) points for verification. All experiments are conducted using Matlab (R2013) software on a personal PC with Intel(R) Core i5, 1.70GHz and 4GB memory.

3.4.1 Time Delay and Embedding Dimension

The time delay which we set to τ is calculated to take the value for which the mutual information has its first minimum [63]. The mutual information $I(y, y_\tau)$ between two time series $y = \{y_{t_1}, y_{t_2}, ..., y_{t_N}\}$ and $y_\tau = \{y_{t_1+\tau}, y_{t_2+\tau}, ..., y_{t_N+\tau}\}$ is the average bits where y was predicted by the measurement from y_τ. $I(y, y_\tau)$ can be represented as

$$I(\tau) = I(y, y_\tau) = H(y) + H(y_\tau) - H(y, y_\tau) \tag{3.17}$$

where $H(y)$ and $H(y_\tau)$ are the entropy of y and y_τ respectively. $H(y, y_\tau)$ is the mutual entropy between y and y_τ. Generally, the moment of the first minimal mutual information is taken as the optimal delay time for PSR. Fig. 3.4 shows the time delay sequence of the monthly tourism arrivals time series from China to Japan with respect to the mutual information. It is apparent that the time delay is 3 months as the first minimal mutual information appears, namely $\tau = 3$. All the time delays for ten major source markets and six continents are summarized in Table 3.1 and Table 3.2 with the values located in the interval of [2, 7].

Once the time delay is determined, Grassberger-Procaccia algorithm is used to calculate the embedding dimension. First, the correlation integral $C(r)$ is calculated:

$$C(r) = \frac{2}{N_m(N_m) - 1} \sum_{1 \le i \le j \le N_m} \varphi(r - |y_i - y_j|) \tag{3.18}$$

where $N_m = N - \tau(m - 1)$, r is the chosen radius and $\varphi(\cdot)$ is the Heaviside function. The correlative dimension $D(m)$ ($D(m) = ln(C(r))/ln(r)$) increases with the increment of the embedding dimension m, and gradually converges to a saturation value. We plot $ln(C(r))$ vs. $ln(r)$ for different m, which is presented in Fig. 3.5, for the monthly tourism arrivals time series from China to Japan. Intuitively, several nearby parallel line segments exist in the figure, which indicate that when $ln(r)$

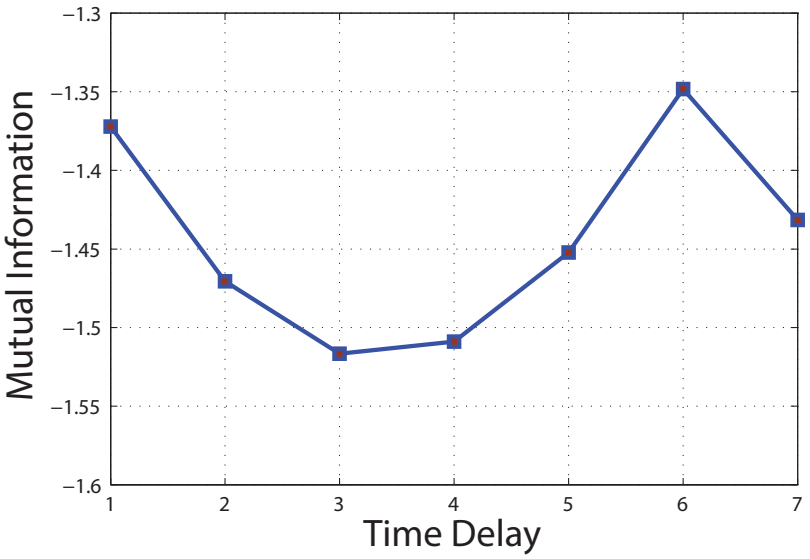


Figure 3.4: The mutual information versus time delay for tourism time series from China to Japan after PSR.

varies in [9.5, 13], the embedding dimension m varies from 1 to 20. The slopes of the line portion can be estimated as the correlation dimension which is shown in Fig. 3.6. The embedding dimension is determined as the value when $D(m)$ first reaches a stable value. Thus, we obtain $m = 8$ for the monthly tourism arrivals time series from China to Japan. The values of the embedding dimension for other time series instances are summarized in Table 3.1 and Table 3.2.

3.4.2 PSR and Lyapunov Exponent

Using the obtained time delay τ and embedding dimension m, we reconstruct the phase space by Eqs. (3.13) and (3.14) from the original one-dimension time series. The reconstructed phase space is exhibited using a three-dimensional phase space, in the condition that embedding dimension $m = 8$ in the tourism time series of China, which means that describing the information onto a lower-dimensional space will be very hard. But we contrive a method that locating three vectors in different three-dimensional phase space will not lose the distortion factor because the three

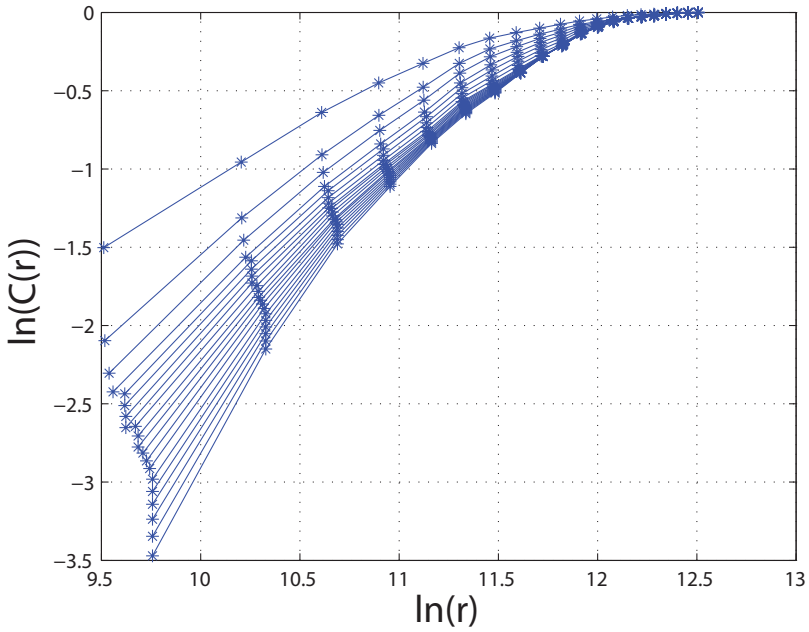


Figure 3.5: The correlation function, ln(C(r)) versus ln(r) of the monthly tourist arrivals from China to Japan.

dimensions contributed to the geometric representation. Because of this, the vectors can also intuitively represent the structure of the attractor. It is shown in Fig. 3.7 that it depicts the results of PSR using two three-dimensional vectors $(y_t, y_{t+3\tau}, y_{t+6\tau})$ and $(y_{t+\tau}, t_{t+4\tau}, t_{t+7\tau})$ for the PSR results of China, respectively. Both three-dimensional vectors show clear chaotic attractors, which suggest that the distributed trace for the tourism exhibits the property of dissipation, and thereby indicating that it is an dynamic system despite possessing the features of a strange attractor. Similar PSR results can also be plotted for the other tourism time series.

The Lyapunov exponents are the average exponential rates of convergence of adjacent orbits in phase space. All maximum Lyapunov exponents (MLE) for the tourism time series are calculated to verify whether the tourism system is chaotic and further to determine the limitation of the predication. The MLE results are also summarized in Table 3.1 and Table 3.2, and these MLE are positive values between [0, 1] for all cases, indicating chaotic behaviors. Besides, as the obtained MLE has relative large

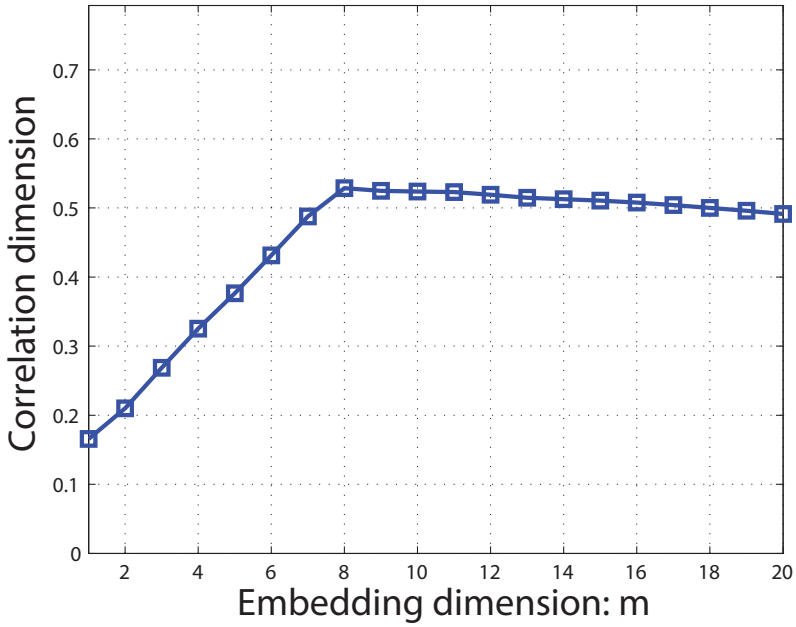


Figure 3.6: Correlation dimension (fitted ln(C(r)/ln(r))) versus the embedding dimension for tourism time series from China to Japan after PSR.

values, it is more reliable to predict the tourism arrivals in a shorter time range (i.e., to perform a short-term forecasting).

3.4.3 Short-term Forecasting and Performance Comparison

Generally, with the length of the time range to be forecasted increasing, the predication accuracy will decrease. In this study, we use five years as the forecasting time length to evaluate the performance of our proposed method. It is worth emphasizing that, within the five years, the former estimated values will be used to forecast the latter values bases on PSR. In addition, user-defined parameters in SDNM influence the prediction performance for the tourism time series. These parameters include the number of dendrites M, the parameter k in synapses (Eq. (1)), the parameters k_{soma} and θ_{soma} in the soma function (Eq. (4)), the BP learning rate η (Eqs. (6) and (7)), and the maximum learning epoch L_{max}. It should be noted that the input number parameter N is set to be the embedding dimension m in SDNM. It is not trivial to

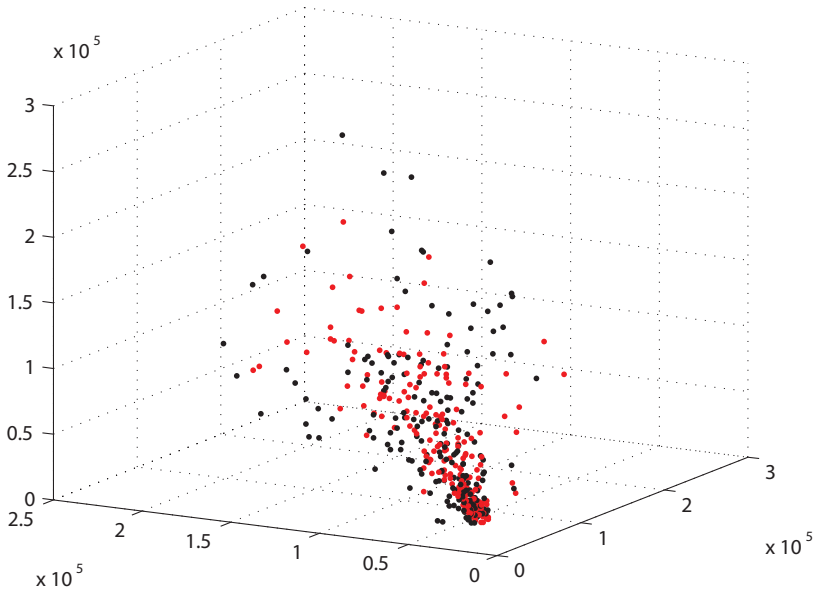


Figure 3.7: Three-dimension phase space for tourism time series from China to Japan after PSR.

set these parameters to obtain the best performance for SDNM, and generally no systematic procedure exists to find out the optimal values for these parameters except the exhaustive method which is very time-consuming. Moreover, it is clear that the parameter M plays a significant influence on the computational time of SDNM.

As a preliminary experiment, we use Taguchi's method [67] to find a reasonable setting combination of these parameters. Taguchi's method tests part of the possible combinations among factors and levels instead of full factorial analysis, and it commits to a minimum of experimental runs and best estimation of the factor main effects over the process [68]. The number of levels for each of the five factors (i.e., the user-defined parameters) is set as follows: four levels for the number of dendrites, that is $M = 1, 3, 5, 10$; four levels for the parameter k, that is $k = 1, 3, 5, 10$; four levels for the parameter k_{soma}, that is $k_{soma} = 1, 3, 5, 10$; four levels for the parameter θ_{soma}, that is $\theta = 0, 0.3, 0.5, 0.9$; and four levels for the BP learning rate η, that is $\eta = 0.005, 0.01, 0.05, 0.1$, respectively. A full factorial design of experiment should

Table 3.1: Results of PSR for the monthly tourist arrivals from **ten major source markets** to Japan: the embedding delay τ, the embedding dimension m, and the maximum Lyapunov exponents MLE.

Country	τ	m	MLE
China	3	8	0.2510
Korea	2	12	0.0779
Hong Kong	3	6	0.3060
Thailand	4	4	0.2121
Taiwan	5	4	0.1416
Singapore	2	18	0.0173
Australia	2	13	0.0333
USA	4	12	0.2520
Canada	2	10	0.1269
UK	2	12	0.0669

Table 3.2: Results of PSR for the monthly tourist arrivals from **six continents** to Japan: the embedding delay τ, the embedding dimension m, and the maximum Lyapunov exponents MLE.

Continent	τ	m	MLE
Asia	2	12	0.0067
Europe	2	14	0.0473
Africa	4	6	0.3339
North America	7	9	0.0146
South America	3	9	0.0691
Oceania	2	14	0.0467

result in a total of $4^5 = 1024$ experiments. In contrast with the full factorial analysis, the Taguchi's method uses the orthogonal arrays reducing the number of experimental runs, and controlling the cost of time, manpower and materials, effectively. Thus, an orthogonal array $L_{16}(4^5)$ which contains only 16 experiments is adopted in the preliminary study.

Table 3.4 summarizes the experimental results based on the orthogonal array and factor assignment, where the MSE values are displayed in the form of "Mean $\pm$ Standard Deviation over 25 runs, and computational times are average values in seconds. As a result, aiming to reduce the running time of training and forecasting, we adopt an acceptable setting of these user-defined parameters based on our preliminary experimental results, shown as: $M = 1$, $k = 5$, $k_{soma} = 5$, $\theta_{soma} = 0.5$, $\eta = 0.05$, and $L_{max} = 1000$. Nevertheless, it is worth noticing that we have to be cautious about

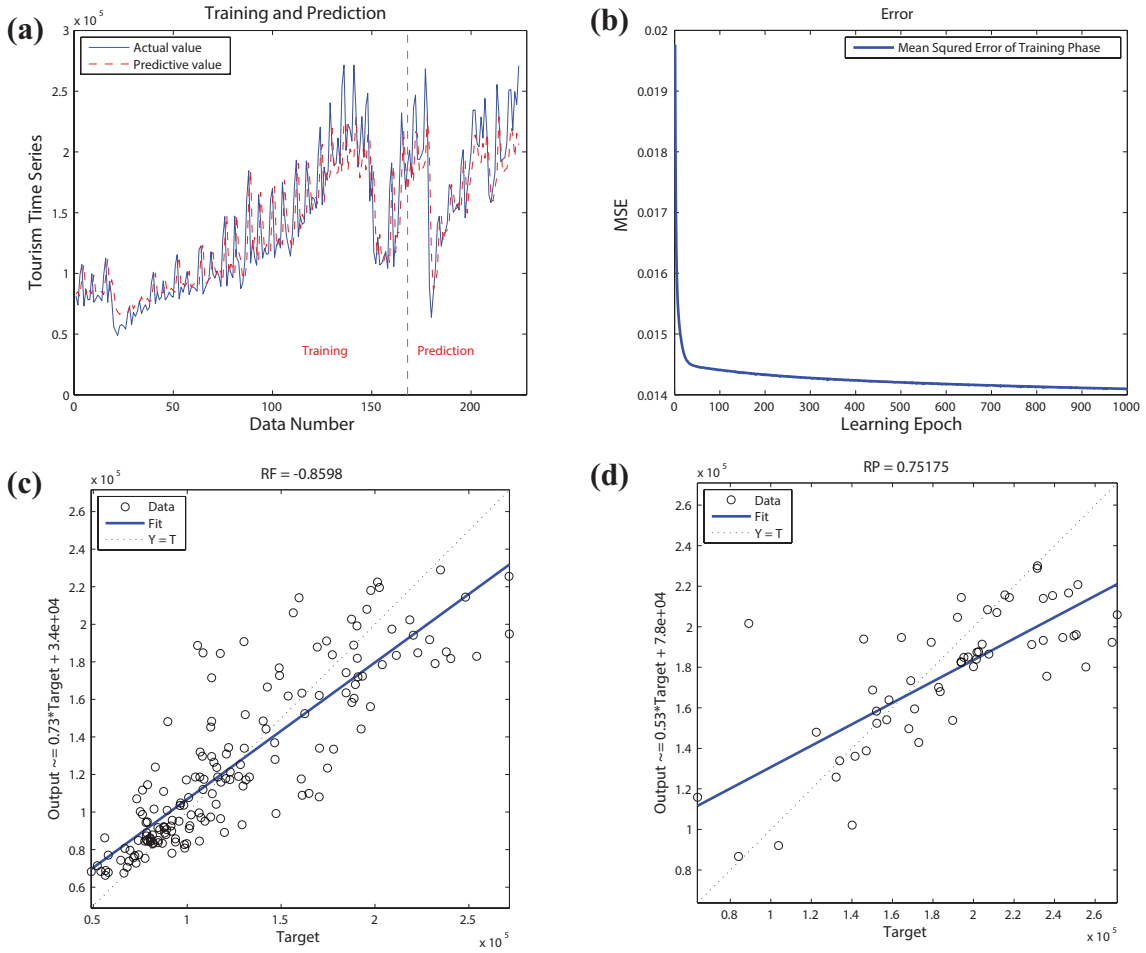


Figure 3.8: Performance of proposed method for the tourism time series of Korea to Japan: (a) training and prediction results, (b) convergence graph based on the BP-like learning, (c) the correlation coefficient of fitting, and (d) the correlation coefficient of prediction.

generalizing our conclusions here until a full factorial analysis is completed.

We use three assessments to evaluate the performance of SDNM, and compare SDNM with the traditional MLP network model [69], the Elman neural network [70], the ANFIS [71] and the SMN [47]. The three assessment criterions are calculated based on Eqs. (3.19) and (3.20).

- The mean square error (MSE) of the predictor for the normalized data is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(O - T)^2 \tag{3.19}$$

- The correlation coefficient of fitting (RF) and the correlation coefficient of prediction (RP) is calculated for the training phase and predication phase, respectively:

$$R = \frac{\sum_{i=1}^{n}(T-\overline{T})(O-\overline{O})}{\sqrt{\sum_{i=1}^{n}(T-\overline{T})^2 \sum_{i=1}^{n}(O-\overline{O})^2}} \tag{3.20}$$

where O is the vector of the output of the used predication model, T is the vector of the true values, and n is the number of data samples (168 in the training phase and 60 in the predication phase).

We implement all predication models for 25 independent runs and the average performance values are summarized in Tables 3.3 and 3.5 for the ten major source market data and six continents data, respectively. In MLP, Elman, ANFIS, and SMN, we adopt the reported values in the original reference for the user-defined parameters. From Tables 3.3 and 3.5, it is clear that all the MSE obtained by SDNM are less than $2.1E-2$, thereby it has the capacity to demonstrate the high accuracy in predicting. The RF and RP values are worse than those obtained using MLP, Elman, ANFIS, and SMN under the condition of using same data. The computational time consumed by SDNM is the least among the five compared models.

Moreover, we plot a typical running result for the tourism time series of Korea in Fig. 3.8, where (a) depicts the data fitting graphs of training and predication; (b) gives a convergence graph of the training phase; (c) illustrates the correlation coefficient of fitting; and (d) is the correlation coefficient of prediction. From this figure, we can find that both training and predication the outputs values obtained by SDNM are quite near the actual values, and a quick convergence is acquired, suggesting that the SDNM is somewhat easy to be trained. Relative high values of the correlation coefficients in training phase ($RF = 0.8598$) and predication phase ($RP = 0.75175$) can be obtained, verifying that the proposed model ban be utilized with great confidence.

All in all, from the experimental results it can be said that SNDM outperforms its competitor models in terms of predication accuracy and computational time.

3.5 Conclusions

In this study, we presented a short-term forecasting model based on a single dendritic neuron model (SDNM) for the tourism arrivals predication. First, chaotic properties of the tourism time series were using three classic indicators in Takens's theorem, including the time delay, the embedding dimension, and the maximum Lyapunov exponent. Then SDNM was used to perform the predication under the condition of the transformed technique of phase space. Experimental results can clearly show that the model does not only has high prediction accuracy but also has fitting effect. Performance comparisons demonstrated the superiority of SDNM.

The contributions of this study can be drawn into three conclusions. Theoretically the neural network models outperform the linear models in predicting nonlinear variables [10, 13, 24]. From the application perspective, SDNM based on PSR provides an effect alternative to learn the chaotic propensities of time series. In practice, the comparative experiment results might give some insights into the selection of neural models for decision makers.

This study opens the door to the following future research. First, more applications should be made on optimization, classification, and predication problems for SDNM to further verify its information processing capacity. Second, settings of the user-defined parameters need to be investigated systematically and some self-adaptive setting mechanisms should be developed. Last but not least, the hardware implementation of the approximated SDNM [45] can also be realized.

Table 3.3: Experimental results the monthly tourist arrivals from **ten major source markets** to Japan.

		MLP	Elamn	ANFIS	SMN	SDNM
China	*MSE*	4.1E-2	5.2E-2	2.0E-2	2.1E-2	1.9E-2
	RF	0.53	0.46	0.75	0.73	0.77
	RP	0.11	0.04	0.66	0.65	0.68
	Time	12.6	14.7	11.9	4.9	2.9
Korea	*MSE*	3.2E-2	2.5E-2	2.7E-2	1.8E-2	1.5E-2
	RF	0.71	0.78	0.76	0.80	0.82
	RP	0.34	0.43	0.55	0.69	0.73
	Time	12.3	14.5	11.6	4.5	2.7
H. K.	*MSE*	4.5E-2	5.0E-2	3.8E-2	2.9E-2	2.1E-2
	RF	0.64	0.44	0.57	0.71	0.80
	RP	0.17	0.12	0.22	0.45	0.71
	Time	12.8	14.6	11.8	4.8	2.8
Thail.	*MSE*	4.1E-2	3.7E-2	3.9E-2	2.6E-2	1.9E-2
	RF	0.60	0.69	0.77	0.78	0.83
	RP	0.25	0.48	0.63	0.69	0.75
	Time	12.9	14.5	11.7	4.9	2.8
Taiwan	*MSE*	3.4E-2	3.7E-2	1.9E-2	2.0E-2	1.3E-2
	RF	0.78	0.73	0.80	0.83	0.86
	RP	0.66	0.69	0.71	0.68	0.82
	Time	12.8	14.3	11.8	4.6	2.8
Sing.	*MSE*	2.1E-2	1.1E-2	7.8E-3	8.1E-3	6.3E-3
	RF	0.73	0.88	0.92	0.95	0.97
	RP	0.55	0.80	0.86	0.89	0.92
	Time	12.4	14.1	11.7	4.5	2.7
Austr.	*MSE*	2.8E-2	2.9E-2	1.8E-2	1.7E-2	1.5E-2
	RF	0.79	0.77	0.80	0.82	0.85
	RP	0.65	0.64	0.72	0.80	0.83
	Time	12.5	14.2	11.9	4.6	2.8
USA	*MSE*	3.1E-2	3.6E-2	1.9E-2	2.1E-2	1.9E-2
	RF	0.65	0.71	0.79	0.81	0.82
	RP	0.63	0.56	0.74	0.62	0.78
	Time	12.4	14.5	11.7	4.6	2.7
Canada	*MSE*	2.9E-2	3.1E-2	1.6E-2	1.9E-2	1.6E-2
	RF	0.80	0.73	0.83	0.81	0.85
	RP	0.71	0.58	0.71	0.64	0.81
	Time	12.5	14.6	11.8	4.5	2.8
UK	*MSE*	2.8E-2	2.9E-2	1.5E-2	1.8E-2	1.3E-2
	RF	0.76	0.83	0.88	0.78	0.90
	RP	0.69	0.67	0.84	0.69	0.84
	Time	12.6	14.7	11.9	4.5	2.8

Table 3.4: Results based on the $L_{16}(4^5)$ orthogonal array and factor assignment.

No.	M	k	k_{soma}	θ_{soma}	η	MSE $(\times 10^{-2})$	Time
1	1	1	1	0	0.005	3.63 ± 0.58	2.8
2	1	3	3	0.3	0.01	2.01 ± 0.45	2.8
3	1	5	5	0.5	0.05	1.57 ± 0.37	2.8
4	1	10	10	0.9	0.1	1.83 ± 0.46	2.8
5	3	1	3	0.5	0.1	1.61 ± 0.57	8.8
6	3	3	1	0.9	0.05	1.56 ± 0.73	8.8
7	3	5	10	0	0.01	2.33 ± 0.75	8.8
8	3	10	5	0.3	0.005	1.98 ± 0.81	8.8
9	5	1	5	0.9	0.01	3.45 ± 1.24	12.7
10	5	3	10	0.5	0.005	2.23 ± 0.95	12.7
11	5	5	1	0.3	0.1	2.35 ± 1.02	12.7
12	5	10	3	0	0.05	2.77 ± 1.45	12.7
13	10	1	10	0.3	0.05	3.05 ± 1.68	18.5
14	10	3	5	0	0.1	2.94 ± 1.43	18.5
15	10	5	3	0.9	0.005	2.37 ± 1.02	18.5
16	10	10	1	0.5	0.01	1.84 ± 0.53	18.5

Table 3.5: Experimental results the monthly tourist arrivals from **six continents** to Japan.

		MLP	Elamn	ANFIS	SMN	SDNM
Asia	MSE	5.2E-2	4.7E-2	2.9E-2	3.9E-2	2.0E-2
	RF	0.31	0.45	0.72	0.72	0.78
	RP	0.11	0.19	0.60	0.59	0.62
	Time	12.7	14.9	11.9	4.7	2.8
Europe	MSE	3.6E-2	3.1E-2	1.8E-2	2.1E-2	1.3E-2
	RF	0.79	0.75	0.83	0.76	0.88
	RP	0.43	0.51	0.76	0.53	0.80
	Time	12.8	14.8	11.8	4.8	2.7
Africa	MSE	3.1E-2	2.5E-2	1.7E-2	1.8E-2	1.3E-2
	RF	0.69	0.76	0.86	0.91	0.93
	RP	0.55	0.62	0.78	0.83	0.87
	Time	12.8	14.8	11.8	4.8	2.8
N. Ame.	MSE	2.5E-2	2.2E-2	1.9E-2	2.0E-2	1.7E-2
	RF	0.84	0.80	0.85	0.74	0.84
	RP	0.77	0.78	0.81	0.72	0.77
	Time	12.9	14.9	11.8	4.9	2.9
S. Ame.	MSE	3.1E-2	2.9E-2	2.1E-2	1.8E-2	1.6E-2
	RF	0.79	0.89	0.82	0.79	0.89
	RP	0.65	0.73	0.76	0.73	0.81
	Time	12.8	14.8	11.8	4.9	2.9
Oceania	MSE	3.3E-2	2.9E-2	2.3E-2	1.8E-2	1.7E-2
	RF	0.78	0.79	0.75	0.85	0.87
	RP	0.62	0.75	0.67	0.76	0.80
	Time	12.8	14.8	11.9	4.8	2.8

Chapter 4

Handling Multiobjectives with Adaptive Mutation based ε-Dominance Differential Evolution

4.1 Introduction

Differential evolution (DE) algorithm [72] is a novel technique that was originally thought to solve the problem of Chebyshev polynomial. It is a population based stochastic meta-heuristic for global optimization on continuous domains which related both with simplex methods and evolutionary algorithms. Due to its simplicity, robustness, and effectiveness, DE is successfully applied in solving optimization problems arising in various practical applications [73], such as data clustering, image processing, etc. DE outperforms many other evolutionary algorithms in terms of convergence speed and the accuracy of solutions. Its performance, however, is still quite dependent on the setting of control parameters such as the mutation factor [74] for complex real-world optimization problems, especially those with multiple objectives [75, 76].

In multiple objective problems, several objectives (or criteria) are, not unusually, stay in conflict with each other, thus requiring a set of non-dominated solutions, i.e., Pareto-optimal solutions to be the candidates for decision. The general goals of this requirement are the discovery of solutions as close to the Pareto-optimal as possible, and the distribution of solutions as diverse as possible in the obtained non-

dominated set. Many works have been reported to satisfying these two goals. Wang et al. [77] proposed a crowding entropy-based diversity measure to select the elite solutions into the elitist archive. Zhang et al. [78] utilized the direction information provided by archived inferior solutions to evolve the differential mutations. Gong et al. [79] introduced the ε-dominance and orthogonal design into DE to keep the diversity of the individuals along the trade-off surface. More recently, Chen et al. [80] proposed a cluster degree based individual selection method to maintain the diversity of non-dominated solutions. A hybrid opposition-based DE algorithm was proposed by combining with a multi-objective evolutionary gradient search [81]. Although these variants of multi-objective DE have demonstrated that DE is suitable for handling multiple objectives, rare work, however, is carried out to discuss the setting of control parameters involving the mutation factor in the multi-objective DE.

Based on the above consideration, in this work, we proposed an adaptive mutation operator into DE to avoid the premature convergence of non-dominated solutions. In the former searching phases, the setting of mutation scale factor F remains large enough to explore the search space sounding to the non-dominated solutions, thus maintaining the diversity of the distribution of Pareto set. Along with the lapse of evolution, F is gradually reduced to perform the exploitation around the promising search area, aiming to reserve good information and to avoid the destruction of the optimal solutions. Furthermore, as noticed by Zitzler et al. [82] that elitism helps in achieving better convergence of solutions in multi-objective evolutionary algorithm, an elitist scheme is adopted by maintaining an external archive of nondominated solutions obtained in the evolution process. Moreover, the ε-dominance strategy [83] which can provide a good compromise in terms of convergence near to the Pareto-optimal and the diversity of Pareto fronts is also used in the algorithm. It is expected that, with the utilization of elitist scheme and ε-dominance, the cardinality of Pareto-

optimal region can be reduced, and no two obtained solutions are located within relative small regions. To verify the performance of the proposed algorithm, five widely used benchmark multiple objective functions are utilized as the test suit. Experimental results indicate that the proposed adaptive mutation based multi-objective DE outperforms traditional multi-objective evolutionary algorithms in terms of the convergence and diversity of the Pareto fronts.

4.2 Brief Introduction to DE

The standard DE is essentially a kind of special genetic algorithm based on real parameter and greedy strategy for ensuring quality. The conventional DE algorithm can be divided into 4 phases: initialization vectors, mutation of the vectors, crossover or recombination, and selection. DE begins its search with a randomly initiated population for a global optimum limited in a D-dimensional real parameter space. We denote subsequent generations in DE by $G = \{0, 1, 2, \cdots, G_{max}\}$ and the i-th $(i = 1, 2, ..., NP)$ individual of the current population is denoted as $X_{i,G} = (x^1_{i,G},\ x^2_{i,G}, ...\ x^j_{i,G}, ...,\ x^D_{i,G})$. The initial population is randomly generated by:

$$x_{j,i,0} = x_{j,min} + rand_{i,j}[0, 1] * (x_{j,max} - x_{j,min}) \tag{4.1}$$

where $rand_{i,j}[0, 1]$ is a uniformly distributed random number in $[0, 1]$, $x_{j,min}$ and $x_{j,max}$ represents the boundary values of the search space. For each individual vector $X_{i,G}$ (target vector), differential evolution algorithm uses mutation operator to generate a new individual $V_{i,G}$ (variation vector), which is generated according to Eq. (2).

$$V_{i,G} = X_{r1,G} + F * (X_{r2,G} - X_{r3,G}) \tag{4.2}$$

where three individuals vectors $X_{r1,G}$, $X_{r2,G}$ and $X_{r3,G}$ are selected randomly from the current populations. $r1, r2, r3 \in \{1, 2, \cdots, NP\}$ are random indexes. F is a constant scale factor $\in [0, 2]$ which controls the amplification of the differential variation ($X_{r2,G}$ - $X_{r3,G}$). For increasing the potential diversity of the perturbed parameter vectors, a crossover operation is implemented after the mutation. The binomial crossover operation was shown in the following.

$$u_{i,G} = \begin{cases} v^j_{i,G}, & \text{if } rand_{i,j}[0,1] \leqslant C_r \ \ or \ \ j = j_{rand} \\ \\ X^j_{i,G}, & \text{otherwise} \end{cases} \tag{4.3}$$

where C_r is called the crossover rate. $rand_{i,j} \in [0, 1]$. After DE generates offspring through mutation and crossover operation, the one-to-one greedy selection operator is performed as:

$$u_{i,G+1} = \begin{cases} U^j_{i,G}, & \text{if} \quad f(U_{i,G}) \leqslant f(X_{i,G}) \\ \\ X^j_{i,G}, & \text{otherwise} \end{cases} \tag{4.4}$$

4.3 Design of multi-objective differential evolution algorithm

For solving multiple objective problems, the general requirements of the approximation of the Pareto optimal set are two-fold: (1) minimize the distance to the true pareto optimal fronts, and (2) the distribution of the obtained non-dominated solutions are located as diverse as possible [84]. The purpose of this research is aimed to address the above two requirements, and the processes of the proposed adaptive mutation based ε-dominance differential evolution (IDE) are summarized in Fig. 4.1.

To generate initial solutions evenly located over the whole decision space, the orthogonal experimental design method [85] is adopted in IDE. Refer to [86] for detailed

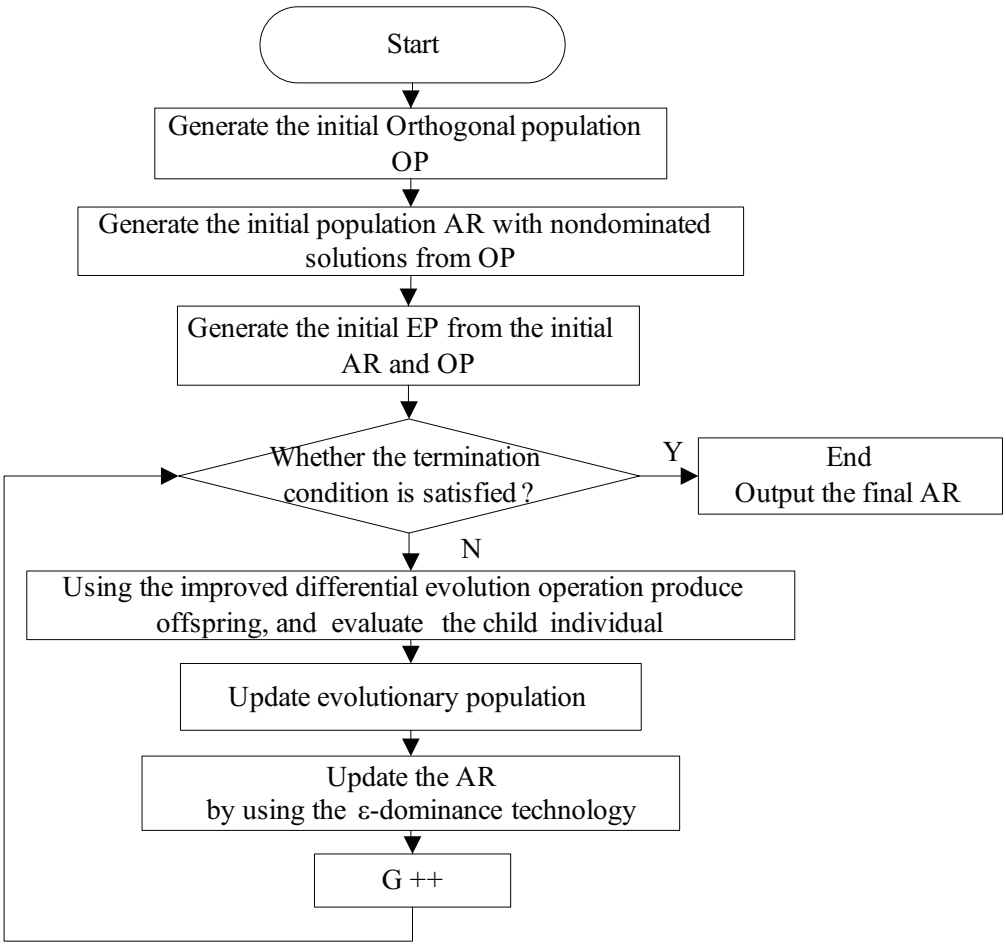


Figure 4.1: The general flow chart of the proposed adaptive mutation based multi-objective differential evolution (IDE).

description of the orthogonal experimental design in population-based evolutionary algorithm. After generating the orthogonal population (denoted as *OP*), an initial archive with the nondominated individuals extracted from *OP* through the traditional Pareto dominance method [87] is created. Then the initial evolutionary population (*EP*) which is responsible for finding new non-dominated solutions is generated from the initial archive and *OP*. If the size of initial archive is larger than *NP*, we select *NP* solutions from initialized randomly. In order to accelerate the algorithm convergence and make use of the archive individual to guide the evolution, we adopt a hybrid selection mechanism when selecting the target vector X_{r1} as shown in Eq. (2). At

the beginning phase of the evolution, and then randomly choose the parents in EP to generate the offspring. With the lapse of evolution, the elitist selection is used. The essential parent is selected from the solutions, and the other two parents are selected from the evolution population EP randomly.

In previously reported works [77–81], all those multi-objective DE algorithms set the scaling factor F as a constant in the whole process of evolution, which made the search appear precocious phenomenon frequently. It is very sensitive to set scaling factor F for traditional differential evolution algorithms. Experimental work in a variety of DE algorithms has provided strong evidence supporting the view that the performance of the algorithm is strongly depending on the setting of F values [88,89]. To be more specifically, if the F value is too large, the DE algorithm approximates for random search, thus the search efficiency and the accuracy of getting the global optimal solution are quite low. On the contrary, if the F value is too small, it can lose the diversity of population into the prematurity. To alleviate this problem, we propose an adaptive mutation operator that can determine the mutation rate adaptively according to the progress of the search of the algorithm, thus enabling the algorithm to possess greater mutation rates in the early search stages to maintain the individuals' diversity and to avoid precocious phenomena during the process. Later, the mutation operator was gradually reduced to reserve good information and avoid the destruction of the optimal solution, and meanwhile it increases the probability of searching to the optimal solutions.

To realize the above characteristic of the setting of F, an adaptive setting rule is designed as in Eqs. (6) and (7).

$$t = e^{1-\frac{G_m}{G_m+1-G}} \tag{4.5}$$

$$F = F_0 * 2^G \tag{4.6}$$

where F_0 is initial mutation operator. G_m denotes the maximum number of fitness evaluation. G indicates the current evolution number. At the beginning search phase of the algorithm, the adaptive mutation operator is carried out with a probability within [F_0 - $2F_0$], which is a relatively large value to maintain the individual diversity. Along with the lapse of evolution, the mutation operator is gradually reduced to reserve good information and expected to well balance the exploration and exploitation of the search.

In addition, as noticed by Zitzler et al. [90] that elitism helps in achieving better convergence in handling multiple objectives. Therefore, in this paper, the elitist scheme is adopted through maintaining an external archive AR of nondominated solutions found in evolutionary process. At the aim of achieving faster convergence, we adopted [91] ε-dominance mechanism to update archive population. At each generation, the newly generated non-dominated solution is compared with each other member which is already contained in the archive. The new individual can be saved in the archive only when it meets the requirements that no individuals within a ε distance exist. By doing so, we can ensure both convergence and diversity of the Pareto fronts within reasonable computational times.

4.4 Simulation and Analysis

Multi-objective optimization problem is also known as multi-criteria optimization problem [92]. In order to evaluate the effectiveness of the proposed IDE and make a comparison with other multi-objective evolutionary algorithms, five widely used benchmark problems [90] involving ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 are adopted as the test suit. All problems have two objective functions and all objective functions are to be minimized. The parameter settings of IDE are as follows: the maximum number of fitness evaluation $G_m = 5000$, the initial scaling factor value of

Table 4.1: Comparison of the convergence metric between IDE and MDE.

Problem	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
MDE	0.0028	0.00064	0.0038	0.0026	0.0008
IDE	0.00075	0.00084	0.0030	0.0020	0.00075

Table 4.2: Comparison of the diversity metric between IDE and MDE.

Problem	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
MDE	0.2536	0.38565	0.40025	0.3850	0.3571
IDE	0.2425	0.2896	0.39575	0.2709	0.2595

F_0=0.5, the crossover probability of $CR = 0.3$, $NP = 100$. For each problem, we run 50 times independently with different random seeds, then compared the performance of IDE with the one of the traditional multi-objective DE variants (MDE) [79]. In addition, we compared the results of IDE algorithm with NSGA-II [87], SPEA2 [93] and MOEO [94]. To assess the performance of the compared algorithms, the convergence metric λ and the diversity metric Δ are used [84]. The first convergence metric λ measures the distance of the obtained non-dominated sets Q and the true Pareto front approximation sets P^* as in Eq. (7).

$$\lambda = \frac{\sum_{i=1}^{|Q|} d_i}{\mid Q \mid} \tag{4.7}$$

where d_i is the Euclidean distance between the solution $i \in Q$ and P^*. It is clear that the lower the λ value, the better convergence of obtained solutions, suggesting that the obtained non-dominated sets are more closer to the true Pareto fronts.

The second diversity metric measures the extent of distribution among the obtained non-dominated sets Q. Δ is defined as in Eq. (8).

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{|Q|-1} \mid d_i - \bar{d} \mid}{d_f + d_l + (\mid Q \mid -1)\bar{d}} \tag{4.8}$$

where d_i measures the Euclidean distance of each point in Q to the point nearby.

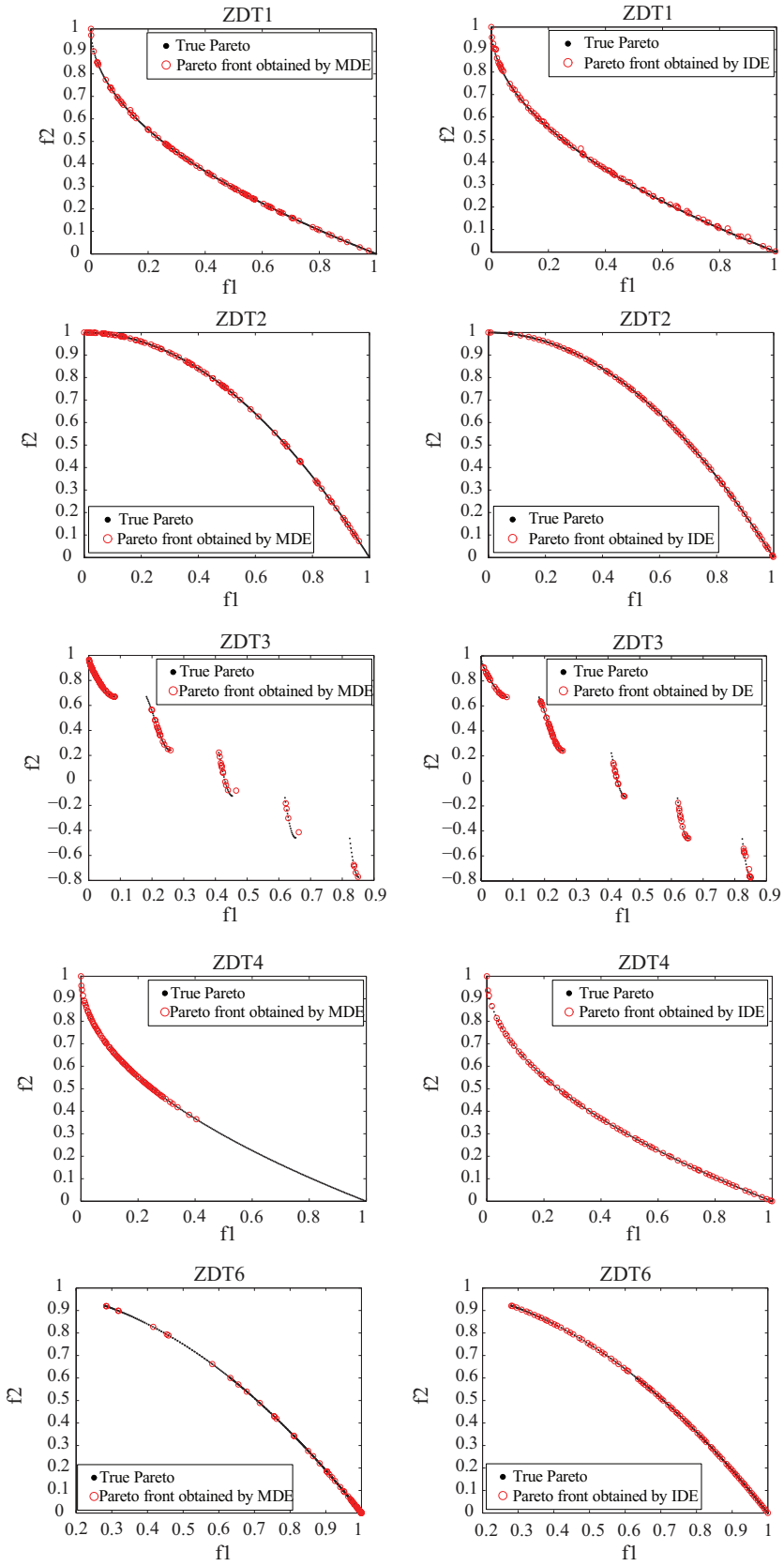


Figure 4.2: Pareto fronts obtained by IDE and its competitor algorithm MDE on ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 respectively.

Table 4.3: Comparison of the convergence metric during IDE, NSGA-II, SPEA2, and MOEO.

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGA-II	0.033482	0.072391	0.114500	0.513053	0.296564
SPEA2	0.023285	0.16762	0.018409	4.9271	0.23255
MOEO	0.001277	0.001355	0.004385	0.008145	0.000630
IDE	0.00075	0.00084	0.0030	0.0020	0.00075

Table 4.4: Comparison of the diversity metric during IDE, NSGA-II, SPEA2, and MOEO.

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
NSGA-II	0.390307	0.430776	0.738540	0.702612	0.668025
SPEA2	0.154723	0.33945	0.4691	0.8239	1.04422
MOEO	0.327140	0.285062	0.965236	0.275567	0.225468
IDE	0.2425	0.2896	0.39575	0.2709	0.2595

d_f and d_l denote the Euclidean distance between the extreme points in Q and P^*, respectively. Obviously, the lower the Δ value is, the better distribution of solutions possess.

Table 4.1 records the convergence metric λ obtained by IDE and the previous MDE algorithm [79]. The diversity metric Δ obtained by IDE and MDE are shown in Table 4.2. Table 4.3 shows the convergence metric obtained by IDE and three multi-objective evolutionary algorithms. Table 4.4 illustrates comparative results in terms of the diversity metric obtained by IDE and its competitors. From Table 4.1, we can find that IDE performs better results with respect to the convergence on all tested instances, except on ZDT2, which suggested that the incorporated adaptive mutation strategy indeed help the search finding better solutions. On the other hand, the comparative results in Table 4.2 show that IDE has capacity of finding a better spread of solutions than MDE on all problems except ZDT6. From Table 4.3, it is clear that IDE produces solutions significantly closer to the true Pareto fronts than NSGA-II, SPEA2, and MOEO on all tested functions. An exception is that MOEO can find slightly better solutions than IDE on ZDT6. With regards to the diversity of obtained non-dominated solutions, as shown in Table 4.4, an overall improvement

can be found on IDE that its non-dominated solutions located more evenly than those obtained by its competitor algorithms, verifying that the proposed adaptive mutation strategy together with the ε-dominance no doubt improve the performance of DE in terms of the diversity.

Furthermore, to further understand the performance of our improved algorithm more intuitively, Fig. 4.2 draws the Pareto fronts constructed by the obtained non-dominated solutions that obtained by IDE and MDE on all tested functions respectively. From this figure, it is clear that the Pareto fronts obtained by IDE is much better than those by MDE. The performance on ZDT6 is quite illuminating to further elaborate the search characteristics of the compared algorithms. Almost the same number of non-dominated solutions are obtained by both algorithms, and the average distance (measured by λ) to the true Pareto front is also within an acceptable tolerance (0.0008 vs 0.00075). Nevertheless, the distribution of the non-dominated solutions is quite different (0.3571 vs 0.2595). A significantly evenly distributed non-dominated solutions for ZDT6 are obtained by IDE, implying that IDE is capable of finding a well-distributed and near-complete set of non-dominated solutions when handling multiobjectives.

4.5 Conclusion

This paper proposed an adaptive mutation operator based on the multi-objective differential evolution algorithm. In the beginning of search phase, the algorithm has a relatively large value to maintain the individuals' diversity, and avoid the premature phenomenon of fast convergence. With the lapse of evolution, the mutation operator was gradually reduced to reserve good information and avoid the destruction to the optimal solution. Together with the ε-dominance strategy, we constructed the effective IDE to handling multiple objectives. We test IDE via five standard

multi-objective test functions and the performance comparison during MDE, NSGA-II, SPEA2 and MOEO. It can be concluded that IDE is superior to other algorithms on multiple problems, indicating that our approach has ability to obtain effective uniformly distributed and near-optimal Pareto sets.

Chapter 5

Improved GSA with chaotic local search

5.1 Introduction

Gravitational search algorithm (GSA) [95] is one of the newest heuristic optimization methods based on Newtonian laws of gravity and motion. It has shown remarkable search abilities in solving optimization problems [96] within high-dimensional search spaces. In GSA, a series of candidate solutions are kept as a group of objects. At each iteration, the objects update their solutions by moving stochastically. The objects with heavier masses have stronger attraction to other objects and move more slowly than the objects with lighter masses. By lapse of iterations, all other objects tend to move towards the heaviest object which corresponds to be the best solution for optimization problem. GSA is robust and can be simply used so it can be applied into many optimization problems [97]. However, GSA still has some drawback such as stick into local minima and slow convergence speed that reduce the solution quality.

To resolve the aforementioned problem, chaos, which is of randomicity, ergodicity and regularity was incorporated into GSA [98]. Chaos is a very common phenomenon of non-linear systems and has recently received many interests. In the field of optimal design, the ergodicity of chaos has been viewed as a optimization mechanism to avoid falling into the local search process. The chaotic state was introduced into the opti-

mization variables and did search using the chaos variables [99]. Meanwhile kinds of chaos optimization algorithm applying to solve the complex object for optimization problem were put forward [100, 101]. The search based on chaos has stronger exploration and exploitation capability and can enable the algorithm to effectively jump out of local extremum due to the inherent ergodicity of chaos. It has been demonstrated that the combination of GSA with chaotic system can alleviate the shortcomings of GSA and thus highlight the advantages of the usage of chaotic systems [98, 102].

There are two methods to combine GSA with chaos. One uses chaotic maps to generate chaotic sequences to substitute random sequences, while the other employs chaos to act as a local search approach. In our previous work, the logistic map was utilized to generate chaotic sequences and perform the local search [98]. In this study, other four different chaotic maps involving the piecewise linear chaotic map, the gauss map, the sinusoidal map, and the sinus map are utilized to combine with GSA. It is apparent that different chaotic maps possess distinct distribution characteristics. The objective of this work is, not only to find out which chaotic map most greatly improve the performance of GSA, but only to give some insights to the underlying reasons. To realize these, six commonly used benchmark optimization functions are chosen from the literature. The experimental results verify that all five incorporated chaotic maps can improve the performance of GSA in terms of the solution quality and convergence speed. In addition, the four newly incorporated chaotic maps exhibit better influence on improving the performance of GSA than the logistic map, suggesting that the hybrid searching dynamics of CGSA is significantly effected by the distribution characteristics of chaotic maps. Furthermore, simulation results also show that the performance of CGSA is tightly related to the search dynamics which results from the interaction between the incorporated chaotic map and the landscape of the solved problems.

The rest of the paper is divided as follows: Section II presents a brief description of the GSA. The five chaotic maps used in the chaotic local search procedure is introduced in Section III. In Section IV, the chaotic gravitational search algorithms using five different maps are proposed. Section V gives the experimental results of the five variants of CGSA on the six benchmark optimization functions. Finally, the conclusion can be drawn in the last.

5.2 Overview of GSA

GSA is a new stochastic algorithm introduced by Reshedi et al. [95]. It is a global search strategy that can handle efficiently arbitrary optimization problems. It is an efficient optimization algorithm with good search ability. it is based on the Newtonian laws of gravity. In GSA, the search agents can be considered as objects and the better their performance is their mass will be heavier. All these objects attract each other by a gravity force [103, 104] and this force will generate a acceleration of all objects towards the objects which have heavier mass. Hence, objects cooperate with each other directly according to the gravitational force. The agents with heavier masses will have slower velocity than the lighter ones, which makes sure that the algorithm can exploit to find the optima around a good solution. Supposing that there are N agents (objects) in the search space, and we define the position of the ith agent by:

$$X_i = (x_i^1, x_i^2 \ldots, x_i^d, \ldots, x_i^n) \quad i = 1, 2, \ldots, N \tag{5.1}$$

where x_i^d is the position of the ith agent in the dth dimension. n is the dimension of the search space.

At the tth iteration, force between search agents acting on the ith object from the

jth object is shown as follows [95]:

$$F_{ij}^{d}(t) = G(t)\frac{M_j(t)M_i(t)}{R_{ij}(t)+\varepsilon}(x_j^d(t) - x_i^d(t)) \tag{5.2}$$

where M_i and M_j are masses of search agents. $G(t)$ is the gravitational constant at interation t, ε is a very small constant to prevent the distance between two agents is too small and $R_{ij}(t)$ indicates the Euclidean distance between two agents i and j:

$$R_{ij}(t) = \| x_i(t), x_j(t) \|_2 \tag{5.3}$$

The gravitational constant $G(t)$ is generated when the iteration starts and it is reduced by the lapse of time to implement exlpitation. $G(t)$ is given by [105]:

$$G(t) = G_0 e^{-\alpha t/iter_{max}} \tag{5.4}$$

where G_0 is the initial value, α is a user-defined parameter, and $iter_{max}$ is the maximum number of iterations.

The total force acting on the ith agent is given by:

$$F_i^d(t) = \sum_{j=1, j\neq i}^{Kbest} rand_j F_{ij}^d(t) \tag{5.5}$$

where $Kbest$ is the K search agents with better fitness. It is a function of time that decreases linearly along with iteration time [95] and at the end of iterations there will be only 2% search agents have the gravitational force to attract others. $rand_j$ is a random number in [0,1]. And the acceleration $a_i^d(t)$ of the agent i at time t and in dth dimension is given by:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{5.6}$$

where $M_i(t)$ is a variable connected with the fitness and it can be calculated as follows:

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{5.7}$$

and

$$m_i = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{5.8}$$

where $best(t)$ is value of the global best agent obtained so far, $worst(t)$ is the value of the worst search agent, and $fit_i(t)$ represents the fitness of agent M_i by calculating the objective functions.

The new velocity of an agent is the sum of its current velocity with its acceleration. Thus the position and the velocity of the ith agent at tth iteration in the dth dimension is calculated as follows:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \tag{5.9}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{5.10}$$

where $rand_i$ is a uniform random variable generated in the interval [0,1], which in fact is an attempt of giving randomized characteristics to the search. The flow chart of GSA is given in the following.

Traditional Gravitational Search Algorithm
for all agents i (i=1,2,...,N) **do**
  initialize position x_i randomly in search space
end-for
while termination criteria not satisfied **do**
  for all agent i **do**
    compute overall force $F_i^d(t)$ according to Eqs.(2)-(5)
    compute acceleration $a_i^d(t)$ according to Eq.(6)
    compute current velocity according to Eq.(9)
    compute current position according to Eq.(10)
  end-for
end-while

The main features of GSA are listed as follows:

(1) The object with heavier mass owns a stronger attractive force and moves slower than the lighter agent.

(2) Gravitational constant decreases with time to make the search have better accuracy.

(3) The acceleration of an agent is decided by the total force which is inversely proportional to the distance between two agents.

(4) The next position of agent only depends on its current velocity and current position.

(5) GSA is a less-memory algorithm, and only requires a small memory capability of hardware.

5.3 Chaotic maps

Recently, chaos has attracted much attention because of its dynamic properties. It is widely used in pattern recognition and optimization theory [106]. In this section five chaotic maps are introduced.

5.3.1 Logistic map

The logistic map is a very famous chaotic map and it is applied in many researches about chaos. The map became commonly used after a seminal paper proposed by the biologist Robert May [107]. Its mathematical expression is given by Eq. (11).

$$x_{k+1} = ax_k(1 - x_k) \quad k = 1, 2, \ldots, N \tag{5.11}$$

where x_k is the kth chaotic number at the kth iteration, and a is usually set to 4. The initial number $x_0 \in [0, 1]$ and $x_0 \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$. When the logistic map is implemented into GSA, the hybrid algorithm is called CGSA1.

5.3.2 Piecewise linear chaotic map

Piecewise linear chaotic map (PWLCM) is more and more famous because of its simplicity and dynamical behavior. [108]. The PWLCM can be simply defined in Eq. (12):

$$x_{k+1} = \begin{cases} x_k/p & x_k \in (0, p) \\ (1 - x_k)(1 - p) & x_k \in [p, 1) \end{cases} \tag{5.12}$$

In this thesis, p is set to 0.7. And the GSA combined with PWLCM is labeled as CGSA2.

5.3.3 Gauss map

The Gauss map can be defined by [102, 109]

$$x_{k+1} = \begin{cases} 0 & x_k = 0 \\ (\mu/x_k)\mathrm{mod}(1) & otherwise \end{cases} \tag{5.13}$$

where μ is set to 1. When the gauss map is combined with GSA, the hybrid algorithm is labeled as CGSA3.

5.3.4 Sinusoidal map

The following equation defines the Sinusoidal map [107]

$$x_{k+1} = ax_k^2 sin(\pi x_k) \tag{5.14}$$

For $a = 2.3$ and $x_0 = 0.7$, it has the following simplified form

$$x_{k+1} = sin(\pi x_k) \tag{5.15}$$

When the sinusoidal map is combined with GSA, the hybrid algorithm is labeled as CGSA4.

5.3.5 Sinus map

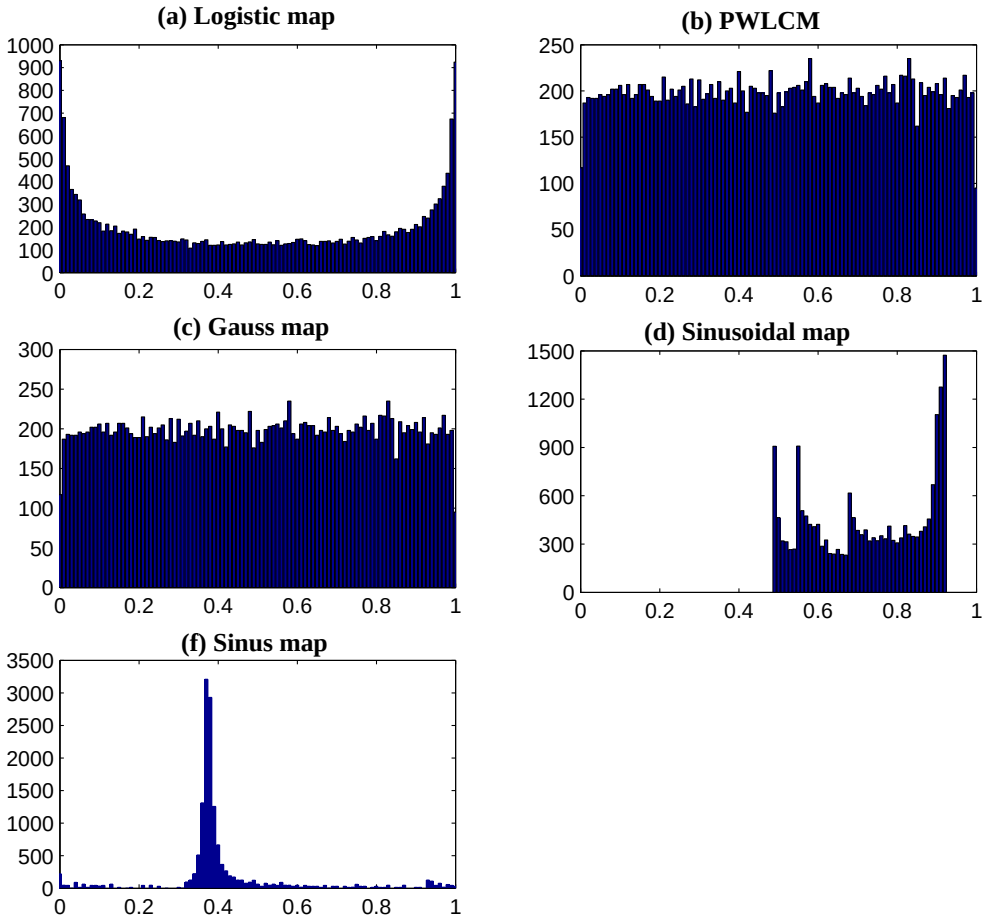
Sinus map is defined as

$$x_{k+1} = 2.3(x_k)^{2sin(\pi x_k)} \tag{5.16}$$

When the sinus map is combined with GSA, the hybrid algorithm is labeled as CGSA5.

To illustrate the details of chaos, the distributions of x for all the five chaotic maps are given in Fig.5.1. The dynamic ranges of the five chaotic maps are summarized as follows: $[0, 1]$ for the Logistic, PWLCM, and Gauss maps, $[0, 0.92]$ for Sinusoidal map, and $[0, +\infty]$ for Sinus map. It is worth pointing out that: 1) for all the values of x, we take two digits after the decimal point for the convenience of illustration. 2) the distribution of x in PWLCM and gauss map are flatter than the other three

Figure 5.1: The distribution of x under certain system parameters in 20000 iterations when $x_0 = 0.74$.



maps, which suggests that the probabilities of x visiting the values in [0,1] is nearly the same. 3) only the values between $[0, 1]$ are utilized in the chaotic local search. Although Xiang et al. [110] gave an argument that flat distribution of x performed better than rough distributions when it was applied in chaotic search, they only gave the simulation results between PWLCM and the logistic map. It is reasonable that the performance of chaotic search is not only related to the distribution of chaos, but also to the landscape of the optimization function. More evidences can be found in Section V.

5.4 Chaotic gravitational search algorithm

Chaos has many dynamical properties like ergodicity, iteration-based searching algorithms is chaos optimization algorithms (COA) were presented [100, 101]. It is easier for COA to jump out of the local optimum than conventional stochastic optimization algorithms. The chaotic system repeats through all the states of the phase space by its ergodicity, based on the movement rule of its own from an initial state. It can traversal for many times near the current optimal solution due to the advantage of this property of the chaotic system. Compared with the random local search, chaotic local search can express better search ability in escaping from the local optimum and improve the quality of solutions. The flowchart of CLS is given below:

Chaotic Local Search Algorithm
step 1. Set the parameters of a chaotic system and the number of chaotic search L
step 2. According to the chaotic system, get a chaotic sequence whose length is N
step 3. Choose the best individual v_c in the current population
step 4. Recorded chaotic search initial counter as 0
step 5. **while** $(t < L)$
step 6. Superimpose an item of the chaotic sequence on v_c in any dimension to form a new individual that is marked as v_n
step 7. Calculation the fitness value of the new individual v_n
step 8. Compute current velocity according to Eq. (9)
step 9. **for** the optimization function f
step 10. **if** $f(v_c) > f(v_n)$
step 11. $v_c \leftarrow v_n$
step 12. **end-if**
step 13. $t = t + 1$
step 14. **end-while**

Noted that the search space of X_g is builded in a hypercube whose center is X_g in semidiameter r, where $r = \rho \times r$. The constant ρ is set to 0.978.

The improved gravitational search algorithm is combined with chaotic local search. It should be noticed that the local search is only applied to the current global best agent X_g obtained from the GSA. Compared with carrying out local search on all agents, it is expected that this scheme can not only save computational times, but also produce competitive good solutions. The procedure of CGSA is described in the following.

Chaotic Gravitational Search Algorithm
for all agents i (i=1,2,,N)**do**
initialize position x_i randomly in search space
end-for
while termination criteria not satisfied **do**
for all agent i **do**
Compute overall force ${F_i}^d(t)$ according to Eqs. (2)-(5)
Compute acceleration ${a_i}^d(t)$ according to Eq. (6)
Update velocity according to Eq. (9)
Update position according to Eq. (10)
end-for
Find out the global best agent X_g
Implement chaotic local search approach (CLS)
Decrease chaotic local search radius using $r = \rho \times r$
end-while

5.5 Numerical simulation

5.5.1 Experimental setup

For evaluating the performance of the proposed algorithm, six benchmark optimization problems in Table 5.1 are used, where functions $f1$-$f3$ are unimodal functions, while functions $f4$-$f6$ are multimodal functions with plenty of local minima and the number of the local minima in these functions increases exponentially with the dimension of the function. The population size of all constructed algorithms is set to be 50. The maximum iteration number is set to 1000. In order to eliminate stochastic

Table 5.1: The function name, definition, dimension, feasible interval of variants, and the known global minimum of six benchmark function.

Function name	Definition	Dim	Interval	Global minimum
Sphere	$f1(X) = \sum_{i=1}^{n} x_i^2$	30	[-100,100]	0.0
Schwefel 2.22	$f2(X) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0.0
Rosenbrock	$f3(X) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0.0
Schwefel 2.26	$f4(X) = -\sum_{i=1}^{n} \sin(\sqrt{(\lvert x_i \rvert)})$	30	[-500,500]	-418.9829D
Ackley	$f5(X) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})$ $-\exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	30	[-32,32]	0.0
Griewank	$f6(X) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[-600,600]	0.0

discrepancy and give the statical analysis, each algorithm is repeated 30 times. The constants ε, α and G_0 are set to 1000, 1.0E-100 and 100 respectively. The experiments are conducted in Microsoft Visual Studio 2010 on a personal PC.

5.5.2 Results and discussions

We firstly compared the performance during GSA, CGSA1, CGSA2, CGSA3, CGSA4, and CGSA5. Table 5.2 to Table 5.7 recorded the minimum fitness, maximum fitness and average fitness for each algorithm on the benchmark functions respectively. From these tables, it is clear that all chaotic GSAs perform better than GSA, suggesting that chaotic search as a local search approach is able to enhance the global search capacity of the algorithm and prevent the search to stick on a local solution. Moreover the average fitness of the best-so-far solutions found by CGSA3, CGSA4 perform better than CGSA1 for all the six functions, which indicates that the sinusoidal map and gauss map possess the better searching performance than the logistic map used in [98]. Thus, it is evident that the searching dynamics of GSA is definitely effected by the distribution characteristics of chaos, and meanwhile the famous logistic map is might not the best choice for the utilization for many optimization problems.

In order to analysis the final best-so-far solution in details, a box-and whisker diagram is used in Fig. 5.2. The vertical axis indicates the fitness values of the final

solutions and the horizontal axis represents the six algorithms. From Fig. 5.2, it is apparent that CGSA2, CGSA3 and CGSA4 generate better solutions than CGSA1 in terms of not only the maximum, average, and minimum values, but also the lower quartile, median, and upper quartile values of the final best-so-far solutions for $f2$-$f4$ and $f6$. CGSA5 outperforms CGSA1 on $f2$, $f3$, $f4$ and $f6$. In particular, CGSA5 produces significant better solutions than the other algorithms on $f4$. The reason seems to be distinct distribution characteristics of the sinus map where most of the chaotic values are located around 0.4. To sum up, it can be concluded that: 1) hybridization of GSA with chaos is demonstrated to be an essential aspect of the high performance; 2) the four newly incorporated chaotic maps generally exhibit better influence on improving the performance of GSA than the logistic map; and 3) however, there is no specific chaotic maps can enable GSA to achieve the best solution for all optimization problems, suggesting that the performance of hybrid CGSAs are related not only to the search capacity of the algorithm, but also to the landscape of the solved problems.

To give some insights to how the chaotic local search on the search dynamics of GSA, the convergence trendline figures of function $f2$, $f3$, $f4$, and $f6$ obtained by the six algorithms are given in Fig. 5.3. In this figure, the horizontal axis represents the iteration and the vertical axis denotes the average fitness best-so-far solutions in logarithmic scales. The convergence graphs of the last 100 iteration are embedded aiming to show the differences more clearly. It is difficult to distinguish the convergence

Table 5.2: Statistical results of different methods for Sphere function ($f1$).

Method	Minimum fitness	Maximum fitness	Average fitness
GSA	1.21E-17	3.25E-17	2.08E-17
CGSA-1	1.38E-17	3.11E-17	2.11E-17
CGSA-2	8.18E-18	3.50E-17	1.99E-17
CGSA-3	1.11E-17	3.69E-17	2.01E-17
CGSA-4	1.10E-17	4.05E-17	1.98E-17
CGSA-5	1.38E-17	3.65E-17	2.19E-17

graphs for the six algorithms on $f1$ since the algorithm has quite quick convergence speed mainly manipulated by the GSA rather than the chaotic local search. The search behaviors of algorithms on multimodal functions $f4$ and $f6$ are quite illuminating. CGSA3, CGSA4 and CGSA5 performs much faster convergence speed than the other algorithms on multimodal functions, suggesting that the gauss map, the sinusoidal map and the sinus map might be more suitable for helping algorithms to jump out of the local solutions.

Furthermore, we define the ratio of best-so-far solutions found by the five chaotic maps to those found by GSA verse the iteration. We assume AF_{GSA}, AF_{CGSA1}, AF_{CGSA2}, AF_{CGSA3}, AF_{CGSA4}, and AF_{CGSA5} represent the average fitness of best-so-far solution found by GSA, CGSA1, CGSA2, CGSA3, CGSA4, and CGSA5, respectively. The ratio is defined as follows:

$$Ra = \frac{AF_{candidate}}{AF_{GSA}} \tag{5.17}$$

where the candidate is obtained from one of the CGSAs. Fig. 5.4 depicts the ratios of algorithms verse the iteration where the values of the solutions found by GSA are set as the basis, thus forming a horizontal line in the figure. For Fig. 5.4(a), (b) and (d), the values above this line indicate worse solutions found by the algorithm than those by GSA, while the values below the line denote better ones. For Fig. 5.4(c), the inverse cases happen since the values of solutions are negative numbers. From Fig.

Table 5.3: Statistical results of different methods for Schwefel function ($f2$).

Method	Minimum fitness	Maximum fitness	Average fitness
GSA	1.44E-8	3.11E-8	2.28E-8
CGSA-1	1.56E-8	3.04E-8	2.21E-8
CGSA-2	1.48E-8	3.30E-8	2.10E-8
CGSA-3	1.40E-8	3.18E-8	2.11E-8
CGSA-4	1.54E-8	3.13E-8	2.06E-8
CGSA-5	1.38E-8	3.05E-8	2.03E-8

Table 5.4: Statistical results of different methods for Rosenbrock function ($f3$).

Method	Minimum fitness	Maximum fitness	Average fitness
GSA	25.70	152.14	35.19
CGSA-1	25.44	85.49	27.62
CGSA-2	24.80	136.43	33.29
CGSA-3	25.07	27.06	25.42
CGSA-4	25.17	25.58	25.43
CGSA-5	23.73	82.17	29.75

Table 5.5: Statistical results of different methods for Schwefel 2.26 function ($f4$).

Method	Minimum fitness	Maximum fitness	Average fitness
GSA	-3617.23	-2178.52	-2844.65
CGSA-1	-4288.88	-2321.88	-3110.29
CGSA-2	-7693.55	-4158.85	-5250.43
CGSA-3	-7001.99	-3645.29	-5050.60
CGSA-4	-7180.01	-3448.26	-4887.43
CGSA-5	-12561.4	-12123.8	-12383.54

5.4, it is clear that chaotic GSAs significantly outperform GSA on $f3$, $f4$ and $f6$. On the other hand, chaotic GSAs still has capacity of jumping out of local minima on the latter search phases which can be observed from the subfigure of Fig. 5.4(a), although they only produce slightly better solutions than GSA.

5.6 Conclusion

In this paper, improved gravitational search algorithms (CGSA) using five different chaotic maps were proposed to testify optimization problems. These chaotic maps were utilized to carry out the chaotic local search which is inserted into GSA. The architecture of such resultant hybrid algorithm is mingled with chaotic local search and GSA. Experimental results indicated that the chaotic search can effectively improve the current solution found by GSA, thus improving the convergence speed, and further obtaining a higher probability of jumping out of the local optima.

Moreover, the distribution characteristics of the five chaotic maps were also observed. Results suggested that the four newly introduced chaotic maps in this paper

Table 5.6: Statistical results of different methods for Ackley function ($f5$).

Method	Minimum fitness	Maximum fitness	Average fitness
GSA	2.64E-9	4.42E-9	3.40E-9
CGSA-1	2.56E-9	4.70E-9	3.49E-9
CGSA-2	2.63E-9	4.91E-9	3.39E-9
CGSA-3	2.52E-9	4.45E-9	3.42E-9
CGSA-4	2.32E-9	4.49E-9	3.42E-9
CGSA-5	2.90E-9	4.73E-9	3.41E-9

Table 5.7: Statistical results of different methods for Griewank function ($f6$).

Method	Minimum fitness	Maximum fitness	Average fitness
GSA	1.37	12.52	4.28
CGSA-1	1.25	4.50	2.17
CGSA-2	1.01E-14	4.41E-2	3.60E-3
CGSA-3	1.60E-14	7.31E-2	1.02E-2
CGSA-4	1.02E-14	7.5E-2	7.17E-3
CGSA-5	3.62E-2	0.88	0.38

generally exhibit better influence on improving the performance of GSA than the logistic map. Nevertheless, there is no specific chaotic maps can enable GSA to achieve the best solution for all optimization problems, suggesting that the performance of hybrid CGSAs are related not only to the search capacity of the algorithm, but also to the landscape of the solved problems. In the future, we plan to adaptively use multiple chaotic maps simultaneously in the chaotic search to construct a more powerful CGSA and analyze the search dynamics of the algorithm.

Figure 5.2: Statistical values of the final best-so-far solution obtained by the six algorithms.

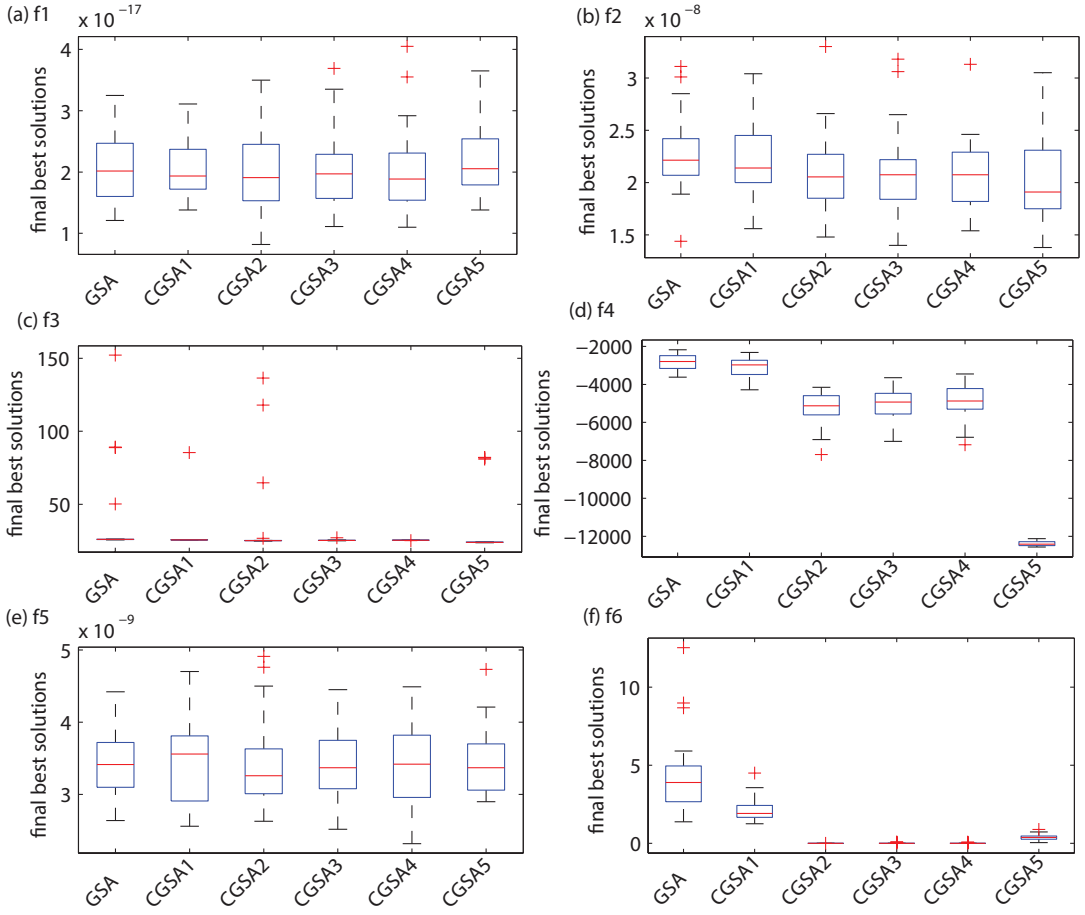


Figure 5.3: The average fitness trendlines of the best-so far solution found by the six algorithms.

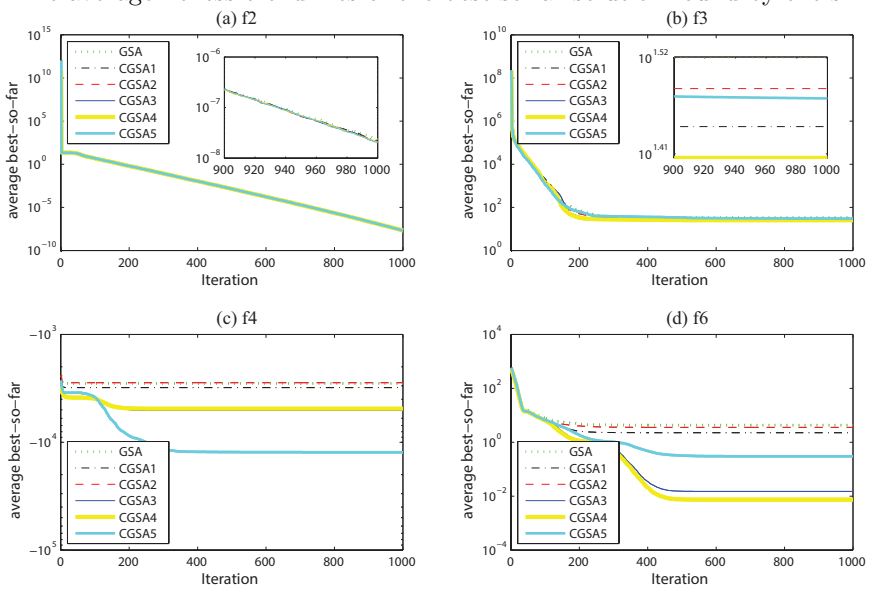
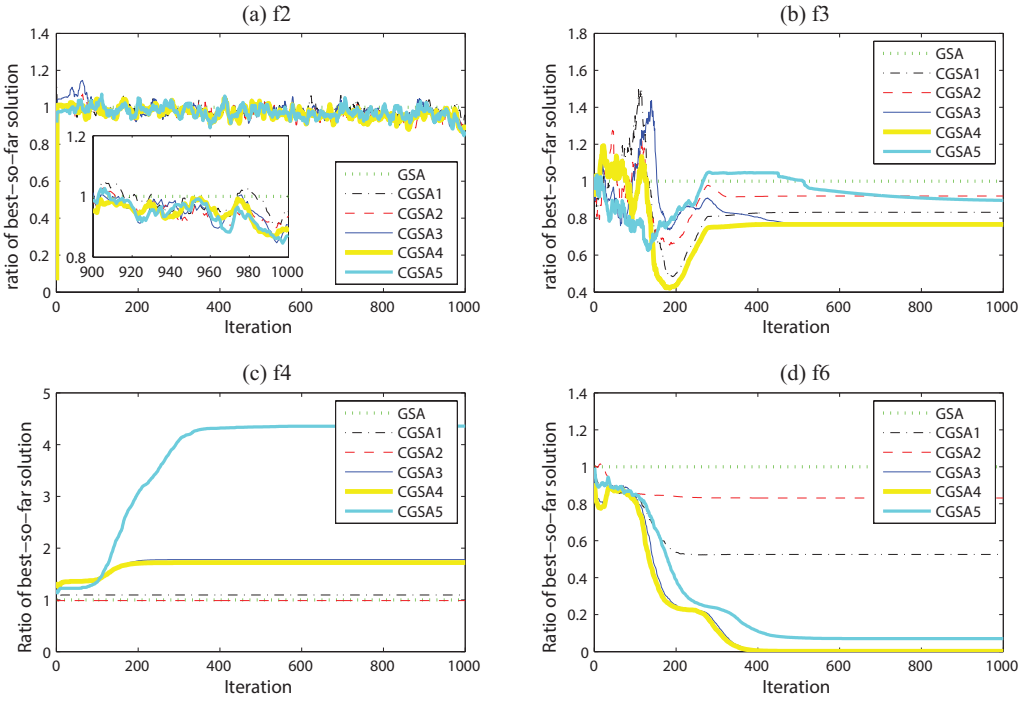


Figure 5.4: The ratio of best-so-far solutions found by the six algorithms.



Chapter 6

General Conclusions and Remarks

In this thesis, we investigate both artificial neural networks and evolutionary algorithms, together with their application in solving various problems. Artificial neural networks(ANNs) is the popular and promising area of artificial intelligence research. it has many learning methods and a variety of network architectures and it can be connected with different computational capabilities to produce neural networks. ANN is a purely computational model based on the human brain's organizational structure. In the past few years, the research of ANNs has made great progress, and successfully solved many modern computer practical problems in the fields of automatic control, intelligent robot, biology, economics, and pattern recognition and prediction estimation. Basically, ANNs always show good intelligence. Prediction is one of the main applications of ANN. We know that ANN's traditional prediction method has excellent classification and pattern recognition ability. Some special features make ANN a better predictive tool. Distinctive features also make predictions valuable and attractive. However, due to the high volatility, irregular motions and non-stationarity of the travel time series, traditional methods often suffer from prediction accuracy problems. In this study, with the rapid development of international tourism, it has become a growing industries with very fast speed in this world. Therefore, the prediction of tourism demand has been a challenge to the international tourism market. A new single dendritic neuron model (SDNM) is proposed to perform tourism demand

forecasting. First, we use phase space reconstruction to analyze the characteristics of tourism and reconstruct the time series into appropriate phase space points. The maximum Lyapunov exponent is then used to identify the chaotic properties of the time series used to determine the prediction limit. Finally, we use SDNM for short-term forecasting. The experimental results of monthly foreign tourists arriving in Japan show that the proposed SDNM model is more efficient and accurate than other neural networks including multilayer perceptrons, neuro-fuzzy inference systems, Elman networks and haploid neurons. On the other hand, multi-objective processing using -Doher differential evolution based on adaptive mutation will be described. Differential evolution (DE) is a well-known and robust population-based stochastic real parameter optimization algorithm in continuous space. DE has recently been shown to be superior to several well-known stochastic optimization methods in solving multi-objective problems. However, its performance is still limited in finding a uniform distribution and approaching the optimal Pareto front. To mitigate this limitation and reduce the computational cost, an adaptive mutation operator is introduced to avoid premature convergence by adaptively adjusting the mutation scale factor F and using the -dominance strategy to update the archives that store non-dominated solutions. Experiments based on five widely used multi-objective functions were performed. The simulation results demonstrate the effectiveness of our proposed approach in solving the Pareto frontier convergence and diversity aspects.

In the future, we plan to use proposed evolutionary algorithms, especially the chaotic mechanisms embedded evolutionary algorithms, to train the artificial neural network to further improve its performance. Following aspects are planed to be carried out in the near future:

- First of all, this research opens the door of the investigation of the hybridization of evolutionary algorithms and the dendritic neural computation models. Inno-

vative ideas should be thought regarding how to effectively use merits of both approach to achieve a better efficient computational hybrid model.

- Second, advanced evolutionary algorithms, such the particle swarm optimization, artificial immune algorithms, gravitational search algorithms, should also be considered to be used as the learning algorithm for dendritic neural model.

- Moreover, the architecture of the dendritic neural model should be modified along with the learning of the algorithms, aiming to provide a more flexible answer when use it to solving complex real-world problems, especially in Big Data or dynamic environments.

- Last but not least, detailed analysis of user-defined parameters should be done to extract some general guidelines when designing related neural computational models.

Bibliography

[1] Http://www.jnto.go.jp/.

[2] Http://www.tourism.jp/.

[3] H. Qu and H. Q. Zhang, "Projecting international tourist arrivals in east asia and the pacific to the year 2005," *Journal of Travel Research*, vol. 35, no. 1, pp. 27–34, 1996.

[4] C. Goh and R. Law, "Modeling and forecasting tourism demand for arrivals with stochastic nonstationary seasonality and intervention," *Tourism Management*, vol. 23, no. 5, pp. 499–510, 2002.

[5] R. Law, "Initially testing an improved extrapolative hotel room occupancy rate forecasting technique," *Journal of Travel & Tourism Marketing*, vol. 16, no. 2-3, pp. 71–77, 2004.

[6] C. Burger, M. Dohnal, M. Kathrada, and R. Law, "A practitioners guide to time-series methods for tourism demand forecastinga case study of durban, south africa," *Tourism Management*, vol. 22, no. 4, pp. 403–409, 2001.

[7] F.-L. Chu, "Forecasting tourism demand: a cubic polynomial approach," *Tourism Management*, vol. 25, no. 2, pp. 209–218, 2004.

[8] J. H. Kim and T. Ngo, "Modelling and forecasting monthly airline passenger flows among three major australian cities," *Tourism Economics*, vol. 7, no. 4, pp. 397–412, 2001.

[9] H. Song and G. Li, "Tourism demand modelling and forecastinga review of recent research," *Tourism Management*, vol. 29, no. 2, pp. 203–220, 2008.

[10] E. Olmedo, "Comparison of near neighbour and neural network in travel forecasting," *Journal of Forecasting*, vol. 35, pp. 217–223, 2016.

[11] F.-L. Chu, "A piecewise linear approach to modeling and forecasting demand for macau tourism," *Tourism Management*, vol. 32, no. 6, pp. 1414–1420, 2011.

[12] K.-H. Huarng, T. H.-K. Yu, and F. Sole Parellada, "An innovative regime switching model to forecast taiwan tourism demand," *The Service Industries Journal*, vol. 31, no. 10, pp. 1603–1612, 2011.

[13] X. Hong, R. J. Mitchell, S. Chen, C. J. Harris, K. Li, and G. W. Irwin, "Model selection approaches for non-linear system identification: a review," *International Journal of Systems Science*, vol. 39, no. 10, pp. 925–946, 2008.

[14] U. Thissen, R. Van Brakel, A. De Weijer, W. Melssen, and L. Buydens, "Using support vector machines for time series prediction," *Chemometrics and Intelligent Laboratory Systems*, vol. 69, no. 1, pp. 35–49, 2003.

[15] P.-F. Pai, H. Wei-Chiang, C. Ping-Teng, and C. Chen-Tung, "The application of support vector machines to forecast tourist arrivals in barbados: An empirical study," *International Journal of Management*, vol. 23, no. 2, p. 375, 2006.

[16] W.-C. Hong, Y. Dong, L.-Y. Chen, and S.-Y. Wei, "Svr with hybrid chaotic genetic algorithms for tourism demand forecasting," *Applied Soft Computing*, vol. 11, no. 2, pp. 1881–1890, 2011.

[17] C.-H. Wang, "Predicting tourism demand using fuzzy time series and hybrid grey theory," *Tourism management*, vol. 25, no. 3, pp. 367–374, 2004.

[18] C. Goh and R. Law, "Incorporating the rough sets theory into travel demand analysis," *Tourism Management*, vol. 24, no. 5, pp. 511–517, 2003.

[19] C. Goh, R. Law, and H. M. Mok, "Analyzing and forecasting tourism demand: A rough sets approach," *Journal of Travel Research*, vol. 46, no. 3, pp. 327–338, 2008.

[20] M. Alvarez-Diaz, J. Mateu-Sbert, and J. Rossello-Nadal, "Forecasting tourist arrivals to balearic islands using genetic programming," *International Journal of Computational Economics and Econometrics*, vol. 1, no. 1, pp. 64–75, 2009.

[21] R. Law and N. Au, "A neural network model to forecast japanese demand for travel to hong kong," *Tourism Management*, vol. 20, no. 1, pp. 89–97, 1999.

[22] R. Law, "Back-propagation learning in improving the accuracy of neural network-based tourism demand forecasting," *Tourism Management*, vol. 21, no. 4, pp. 331–340, 2000.

[23] V. Cho, "A comparison of three different approaches to tourist arrival forecasting," *Tourism Management*, vol. 24, no. 3, pp. 323–330, 2003.

[24] S. C. Kon and L. W. Turner, "Neural network forecasting of tourism demand," *Tourism Economics*, vol. 11, no. 3, pp. 301–328, 2005.

[25] A. Palmer, J. J. Montano, and A. Sesé, "Designing an artificial neural network for forecasting tourism time series," *Tourism Management*, vol. 27, no. 5, pp. 781–790, 2006.

[26] C.-F. Chen, M.-C. Lai, and C.-C. Yeh, "Forecasting tourism demand based on empirical mode decomposition and neural network," *Knowledge-Based Systems*, vol. 26, pp. 281–287, 2012.

[27] O. Claveria, E. Monte, and S. Torra, "Tourism demand forecasting with neural network models: different ways of treating information," *International Journal of Tourism Research*, vol. 17, no. 5, pp. 492–500, 2015.

[28] L. Wang, Y. Zeng, and T. Chen, "Back propagation neural network with adaptive differential evolution algorithm for time series forecasting," *Expert Systems with Applications*, vol. 42, no. 2, pp. 855–863, 2015.

[29] K.-Y. Chen and C.-H. Wang, "Support vector regression with genetic algorithms in forecasting tourism demand," *Tourism Management*, vol. 28, no. 1, pp. 215–226, 2007.

[30] M. Khashei, S. R. Hejazi, and M. Bijari, "A new hybrid artificial neural networks and fuzzy regression model for time series forecasting," *Fuzzy Sets and Systems*, vol. 159, no. 7, pp. 769–786, 2008.

[31] K.-Y. Chen, "Combining linear and nonlinear model in forecasting tourism demand," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10 368–10 376, 2011.

[32] P.-F. Pai, K.-C. Hung, and K.-P. Lin, "Tourism demand forecasting using novel hybrid system," *Expert Systems with Applications*, vol. 41, no. 8, pp. 3691–3702, 2014.

[33] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[34] M. Ardalani-Farsa and S. Zolfaghari, "Chaotic time series prediction with residual analysis method using hybrid elman–narx neural networks," *Neurocomputing*, vol. 73, no. 13, pp. 2540–2553, 2010.

[35] V. Cho, "A study on the temporal dynamics of tourism demand in the asia pacific region," *International Journal of Tourism Research*, vol. 11, no. 5, pp. 465–485, 2009.

[36] J. P. Teixeira and P. O. Fernandes, "Tourism time series forecast-different ann architectures with time index input," *Procedia Technology*, vol. 5, pp. 445–454, 2012.

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

[38] R. Sitte and J. Sitte, "Analysis of the predictive ability of time delay neural networks applied to the s&p 500 time series," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 30, no. 4, pp. 568–572, 2000.

[39] X. Lin, Z. Yang, and Y. Song, "Short-term stock price prediction based on echo state networks," *Expert systems with applications*, vol. 36, no. 3, pp. 7313–7317, 2009.

[40] R. Chandra, "Competition and collaboration in cooperative coevolution of elman recurrent neural networks for time-series prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3123–3136, 2015.

[41] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[42] Y. Todo, H. Tamura, K. Yamashita, and Z. Tang, "Unsupervised learnable neuron model with nonlinear interaction on dendrites," *Neural Networks*, vol. 60, pp. 96–103, 2014.

[43] A. Destexhe and E. Marder, "Plasticity in single neuron and circuit computations," *Nature*, vol. 431, no. 7010, pp. 789–795, 2004.

[44] Z. Sha, L. Hu, Y. Todo, J. Ji, S. C. Gao, and Z. Tang, "A breast cancer classifier using a neuron model with dendritic nonlinearity," *IEICE Trans. Inf. & Syst.*, vol. E98-D, no. 7, pp. 1365–1376, 2015.

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

[46] F. Takens, *Detecting strange attractors in turbulence*. Springer, 1981.

[47] R. Yadav, P. Kalra, and J. John, "Time series prediction with single multiplicative neuron model," *Applied Soft Computing*, vol. 7, pp. 1157–1163, 2007.

[48] L. Zhao and Y. Yang, "Pso-based single multiplicative neuron model for time series prediction," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2805–2812, 2009.

[49] C. Zhang, W. Wu, and Y. Xiong, "Convergence analysis of batch gradient algorithm for three classes of sigma-pi neural networks," *Neural Processing Letters*, vol. 26, no. 3, pp. 177–189, 2007.

[50] 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, 2011, article 5.

[51] C. Koch, T. Poggio, and V. Torre, "Nonlinear interactions in a dendritic tree: localization, timing, and role in information processing," *Proceedings of the National Academy of Sciences*, vol. 80, no. 9, pp. 2799–2802, 1983.

[52] N. Brunel, V. Hakim, and M. J. Richardson, "Single neuron dynamics and computation," *Current opinion in neurobiology*, vol. 25, pp. 149–155, 2014.

[53] L. Abbott and W. G. Regehr, "Synaptic computation," *Nature*, vol. 431, no. 7010, pp. 796–803, 2004.

[54] A. Destexhe and E. Marder, "Plasticity in single neuron and circuit computations," *Nature*, vol. 431, no. 7010, pp. 789–795, 2004.

[55] Y.-N. Jan and L. Y. Jan, "Branching out: mechanisms of dendritic arborization," *Nature Reviews Neuroscience*, vol. 11, no. 5, pp. 316–328, 2010.

[56] B. W. Mel and C. Koch, "Sigma-pi learning: On radial basis functions and cortical associative learning." in *NIPS*, 1989, pp. 474–481.

[57] M. P. Jadi, B. F. Behabadi, A. Poleg-Polsky, J. Schiller, and B. W. Mel, "An augmented two-layer model captures nonlinear analog spatial integration effects in pyramidal neuron dendrites," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 782–798, 2014.

[58] 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.

[59] C. Koch and I. Segev, "The role of single neurons in information processing," *Nature Neuroscience*, vol. 3, pp. 1171–1177, 2000.

[60] T. Jiang, D. Wang, J. Ji, Y. Todo, and S. C. Gao, "Single dendritic neuron with nonlinear computation capacity: A case study on xor problem," in *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*. IEEE, 2015, pp. 20–24.

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

[62] P. Grassberger and I. Procaccia, "Estimation of the kolmogorov entropy from a chaotic signal," *Physical Review A*, vol. 28, no. 4, pp. 2591–2593, 1983.

[63] A. M. Fraser and H. L. Swinney, "Independent coordinates for strange attractors from mutual information," *Physical Review A*, vol. 33, no. 2, pp. 1134–1140, 1986.

[64] 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.

[65] P. Shang, X. Na, and S. Kamae, "Chaotic analysis of time series in the sediment transport phenomenon," *Chaos, Solitons & Fractals*, vol. 41, no. 1, pp. 368–379, 2009.

[66] H. D. I. Abarbanel, *Analysis of observed chaotic data*. Springer, 1996.

[67] G. Taguchi, R. Jugulum, and S. Taguchi, *Computer-based robust engineering: essentials for DFSS*. ASQ Quality Press, 2004.

[68] J. F. Khaw, B. Lim, and L. E. Lim, "Optimal design of neural networks using the taguchi method," *Neurocomputing*, vol. 7, no. 3, pp. 225–245, 1995.

[69] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning internal representations by error propagation*. Institute for Cognitive Science, University of California, San Diego, 1985.

[70] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[71] C. Vairappan, H. Tamura, S. C. Gao, and Z. Tang, "Batch type local search-based adaptive neuro-fuzzy inference system (anfis) with self-feedbacks for time-series prediction," *Neurocomputing*, vol. 72, pp. 1870–1877, 2009.

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

[73] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, no. 99, pp. 1–28, 2010.

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

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

[76] L. V. Santana-Quintero and C. A. C. Coello, "An algorithm based on differential evolution for multi-objective problems," *International Journal of Computational Intelligence Research*, vol. 1, no. 1, pp. 151–169, 2005.

[77] Y.-N. Wang, L.-H. Wu, and X.-F. Yuan, "Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure," *Soft Computing*, vol. 14, no. 3, pp. 193–209, 2010.

[78] J. Zhang and A. C. Sanderson, "Self-adaptive multi-objective differential evolution with direction information provided by archived inferior solutions," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 2801–2810.

[79] W. Gong and Z. Cai, "An improved multiobjective differential evoluton based on pareto-adaptive epsilon-dominance and orthogonal design," *European Journal of Operational Research*, vol. 198, no. 2, pp. 576–601, 2009.

[80] B. Chen, Y. Lin, W. Zeng, D. Zhang, and Y.-W. Si, "Modified differential evolution algorithm using a new diversity maintenance strategy for multi-objective optimization problems," *Applied Intelligence*, pp. 1–25, 2015.

[81] J. K. Chong and K. C. Tan, "An opposition-based self-adaptive hybridized differential evolution algorithm for multi-objective optimization (osade)," in *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems*. Springer, 2015, pp. 447–461.

[82] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[83] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evolutionary computation*, vol. 10, no. 3, pp. 263–282, 2002.

[84] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[85] K. Fang and C. Ma, "Orthogonal and uniform experimental design," *Beijing: Science press*, 2001.

[86] Y. W. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 41–53, 2001.

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

[88] S. Dasgupta, S. Das, A. Biswas, and A. Abraham, "On stability and convergence of the population-dynamics in differential evolution," *AI Communications*, vol. 22, no. 1, pp. 1–20, 2009.

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

[90] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[91] A. Hernández-Díaz, L. Santana-Quintero, C. Coello Coello, and J. Molina, "Pareto-adaptive ε-dominance," *Evolutionary Computation*, vol. 15, no. 4, pp. 493–517, 2007.

[92] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[93] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," in *Proc. Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, 2001, pp. 95–100.

[94] M.-R. Chen and Y.-Z. Lu, "A novel elitist multiobjective optimization algorithm: Multiobjective extremal optimization," *European Journal of Operational Research*, vol. 188, no. 3, pp. 637–651, 2008.

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

[96] P. K. Roy, "Solution of unit commitment problem using gravitational search algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 53, pp. 85–94, 2013.

[97] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Bgsa: binary gravitational search algorithm," *Natural Computing*, vol. 9, no. 3, pp. 727–745, 2010.

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

[99] L. Bing and J. Weisun, "Chaos optimization method and its application," *Control Theory and Applications*, vol. 14, no. 4, pp. 613–615, 1997.

[100] J. Yang, J. Z. Zhou, W. Wu, F. Liu, C. Zhu, and G. Cao, "A chaos algorithm based on progressive optimality and tabu search algorithm," in *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, vol. 5. IEEE, 2005, pp. 2977–2981.

[101] H. Xu, Y. Zhu, T. Zhang, and Z. Wang, "Application of mutative scale chaos optimization algorithm in power plant units economic dispatch," *Journal of Harbin Institute of Technology*, vol. 32, no. 4, pp. 55–58, 2000.

[102] M. Bucolo, R. Caponetto, L. Fortuna, M. Frasca, and A. Rizzo, "Does chaos work better than noise?" *IEEE Circuits and Systems Magazine*, vol. 2, no. 3, pp. 4–19, 2002.

[103] R. Resnick, D. Halliday, and J. Walker, *Fundamentals of physics*. John Wiley, 1988.

[104] P. Schroeder, "Gravity from the ground up," *Proceedings of the NPA*, vol. 7, pp. 498–503, 2010.

[105] R. Mansouri, F. Nasseri, and M. Khorrami, "Effective time variation of g in a model universe with variable space dimension," *Physics Letters A*, vol. 259, no. 3, pp. 194–200, 1999.

[106] S. Talatahari, B. Farahmand 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.

[107] R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, no. 5560, pp. 459–467, 1976.

[108] A. Baranovsky and D. Daems, "Design of one-dimensional chaotic maps with prescribed statistical properties," *International Journal of Bifurcation and Chaos*, vol. 5, no. 06, pp. 1585–1598, 1995.

[109] 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.

[110] T. Xiang, X. Liao, and K. 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.